

# DE10-Nano

## QT CONTROL PANEL



Control Panel

LED    Button    G-Sensor

FPGA LED

<input type="checkbox"/> LED0	<input type="checkbox"/> LED4
<input type="checkbox"/> LED1	<input type="checkbox"/> LED5
<input type="checkbox"/> LED2	<input type="checkbox"/> LED6
<input type="checkbox"/> LED3	<input type="checkbox"/> LED7

HPS LED

<input type="checkbox"/> LED0
-------------------------------

**Light All**

**Unlight All**

**terasic**  
www.terasic.com

# CONTENTS

<b>CHAPTER 1</b>	<b>OVERVIEW.....</b>	<b>1</b>
1.1 SYSTEM BLOCK DIAGRAM .....	1	
1.2 LEARNING TOPICSS .....	2	
1.3 SYSTEM REQUIREMENTS.....	3	
1.4 DEVELOPMENT FLOW .....	3	
1.5 SET UP THE CONTROL PANEL DEMO .....	3	
<b>CHAPTER 2</b>	<b>LINUX X64 INSTALLATION.....</b>	<b>10</b>
2.1 SYSTEM REQUIREMENTS.....	10	
2.2 INSTALL VMWARE WORKSTATION PLAYER.....	11	
2.3 LAUNCH VMWARE.....	16	
2.4 INSTALL LINUX UBUNTU DESKTOP .....	16	
2.5 UPGRADE LINUX X64 SOFTWARE PACKAGE.....	32	
2.6 ENABLE VMWARE SHARE FOLDER .....	35	
<b>CHAPTER 3</b>	<b>QT CREATOR INSTALLATION.....</b>	<b>46</b>
3.1 INSTALL TOOLCHAIN FOR LINUX X64.....	46	
3.2 DOWNLOAD AND INSTALL QT INSTALLER .....	49	
3.3 LAUNCH QT CREATOR AND CHECK CONFIGURE .....	62	
3.4 HELLO PROGRAM.....	68	
<b>CHAPTER 4</b>	<b>INTEL SOC TOOLCHAIN INSTALLATION.....</b>	<b>76</b>
4.1 DOWNLOAD AND INSTALL TOOLCHAIN .....	76	
4.2 SETUP TOOLCHAIN PATH .....	78	
<b>CHAPTER 5</b>	<b>QT LIBRARY INSTALLATION FOR INTEL SOC.....</b>	<b>80</b>
5.1 COPY PREBUILT QT LIBRARY FROM SYSTEM CD .....	80	
5.2 INSTALL PREBUILT QT LIBRARY .....	81	
<b>CHAPTER 6</b>	<b>QT APP FOR INTEL SOC .....</b>	<b>83</b>
6.1 SET UP "BUILD & RUN" IN QT CREATOR .....	83	
6.2 CROSS-COMPILE THE HELLO PROJECT.....	88	
6.3 EXECUTE HELLO PROGRAM.....	92	
<b>CHAPTER 7</b>	<b>CONTROL PANEL QUARTUS PROJECT.....</b>	<b>97</b>
7.1 BUILD QUARTUS PROJECT OF CONTROL PANEL .....	97	

7.2 TEST FPGA CONFIGURATION FILE .....	98
7.3 MORE ON THE QUARTUS PROJECT.....	99
<b>CHAPTER 8      <i>CONTROL PANEL QT PROJECT</i>.....</b>	<b>102</b>
8.1 INSTALL INTEL SOC EDS ON LINUX X64 .....	102
8.2 COPY CONTROL PANEL QT PROJECT .....	110
8.3 BUILD CONTROL PANEL QT PROJECT.....	111
8.4 EXECUTE CONTROL PANEL PROGRAM .....	117
8.5 MORE ON THE CONTROL PANEL QT PROJECT.....	118

# Chapter 1

## Overview

The Intel Cyclone V SoC Development Kit (DE10-Nano), a robust platform that is built around Intel System-on-Chip (SoC) FPGA which combines the dual-core Cortex-A9 embedded cores with industry leading programmable logic for ultimate design flexibility. Intel's SoC FPGA integrates Hard Processor System (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone. Other than having the Intel SoC FPGA, DE10-Nano FPGA development board is equipped with hardware, such as DDR3 memory and display capabilities, Ethernet networking, and much more that promise many exciting applications.

To demonstrate the power of the SoC FPGA on the DE10-Nano FPGA development board, we have created a "Control Panel" program, board utility software that uses ARM to access various peripherals that belong both to the Hard Processor System (HPS) and the FPGA. This demo is particularly useful for anyone who is interested in the SoC FPGA structure and development of future SoC applications.

This scope of this tutorial is to help users understand and learn how to build the board utility "Control Panel" on the DE10-Nano FPGA development board in a step-by-step fashion. To boost user confidence while following the instructions, we provide screenshots of steps needed to be followed. This helps users debug if they encounter any problems during the learning process. Users should have basic concepts toward building their own applications once they have finished working with the whole tutorial. Also, users should find this tutorial self-explanatory and self-contained as we hope that lots of creative projects can be inspired as a result of this.

### 1.1 System Block Diagram

**Figure 1-1** shows the block diagram of the Control Panel where user can see the Control Panel program is running on Linux on the left-hand side. The program GUI is built based on Qt library and it can access the FPGA resources through the AXI bus. The Linux Frame Buffer Display hardware is implemented based on Altera VIP suite and HPS DDR3 SDRAM memory is used as frame buffer.



Block Diagram of DE10-Nano  
Control Panel based on QT

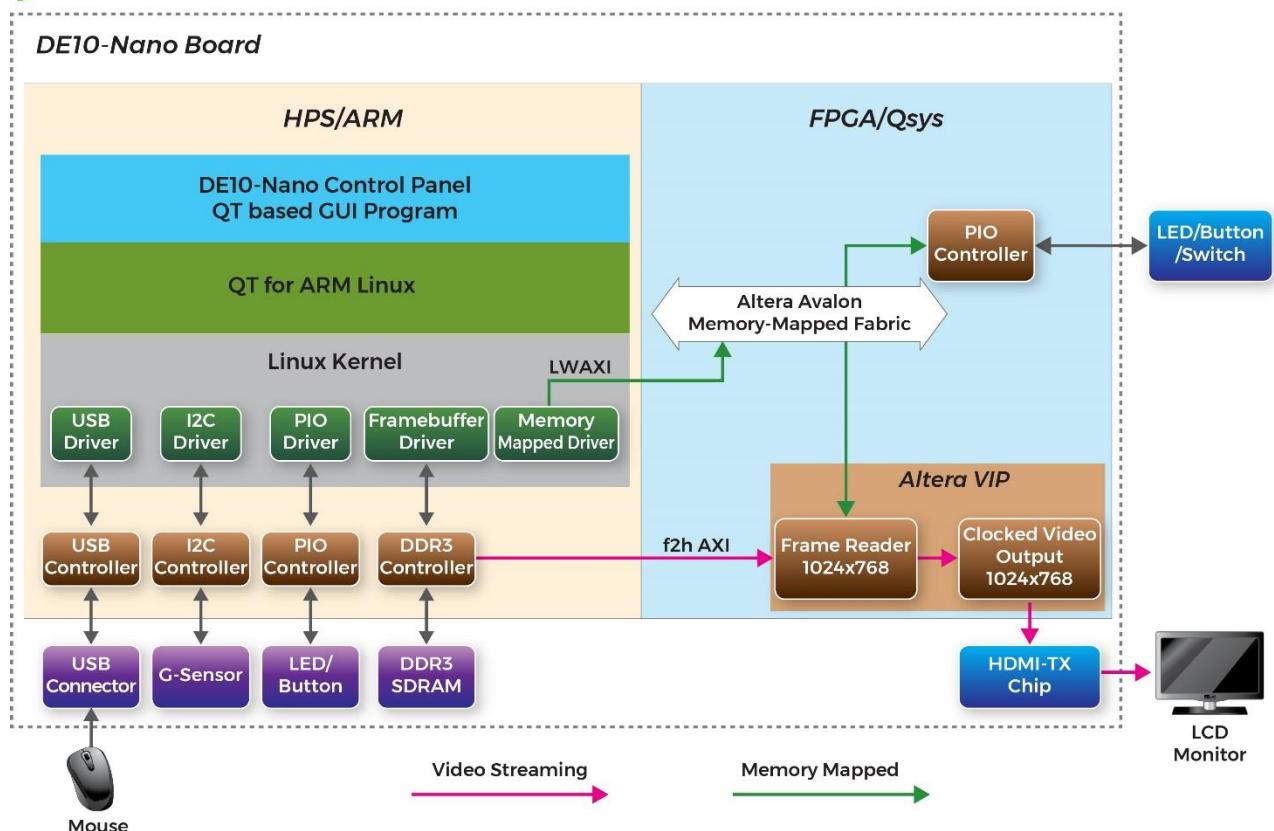


Figure 1-1 Block Diagram of the Control Panel

## 1.2 Learning Topics

To give users an outline what they will learn in this manual, here we provide a list of topics that will be covered and explained:

- Install a virtual machine VMware on 64-bit Windows Host
- Install a Linux x64 under virtual machine VMware
- Install Qt Creator on Linux x64
- Create and build a Qt hello program for Linux x64
- Install Linaro ARM cross compiler toolchain for Intel SoC ARM
- Install prebuild Qt Library for Intel SoC ARM
- Cross-compile to build Qt Application for Intel SoC ARM
- Create and build a Quartus Project for Intel SoC FPGA
- Launch Qt Application on Intel SoC FPGA board

## **1.3 System Requirements**

Before starting this tutorial, please note that the following items are required to complete the Control Panel project:

- A x64 PC
- DE10-Nano FPGA board
- Linux Ubuntu 16.04 x64 Installer
- Options if Windows Host is preferred
  - MS Windows Installer
  - VMware Workstation Player Installer
- Qt 5.7.0 for Linux 64-bit Installer
- Qt 5.5.1 open source
- Quartus 16.0 or later installer

## **1.4 Development Flow**

Our Control Panel program GUI is built using the Qt library. Considering that most users might not be familiar with the Qt library, we believe it would be most helpful to first list all the steps required to complete the control panel project in the following:

1. Install Linux x64 on VMware running under Windows Host
2. Install Qt Creator 5.7.0 on Linux x64
3. Install Intel SoC Toolchain on Linux x64
4. Install Qt 5.5.1 Library for Intel SoC on Linux x64
5. Build Qt App for Intel SoC FPGA Board on Linux x64
6. Build Quartus Project on Windows Host
7. Execute Qt App on Intel SoC FPGA Board

## **1.5 Set up the Control Panel Demo**

This section describes how to set up the Control Panel program on the DE10-Nano Development Kit. There are five major steps involved. Please carefully follow the instructions below. If you already have the MicroSD card for LXDE Desktop, skip first two steps.

1. Download microSD card writer utility – **Win32 Disk Imager**

2. Download the compressed LXDE Desktop image file. Then, decompress the compressed image file, and write the file into a 4GB microSD card with an ImageWriter Utility
3. Insert the microSD card into DE10-Nano FPGA Development Board, connect a USB keyboard and mouse to the USB host port, connect a LCD monitor to the HDMI port, then power on the board to boot Linux System.
4. Boot LXDE Desktop.
5. Double click "ControlPanel" icon on LXDE Desktop to launch the Control Panel Utility. If a confirm dialog appears, please click the "Execute" button.

## ■ Download Win32 Disk Imager

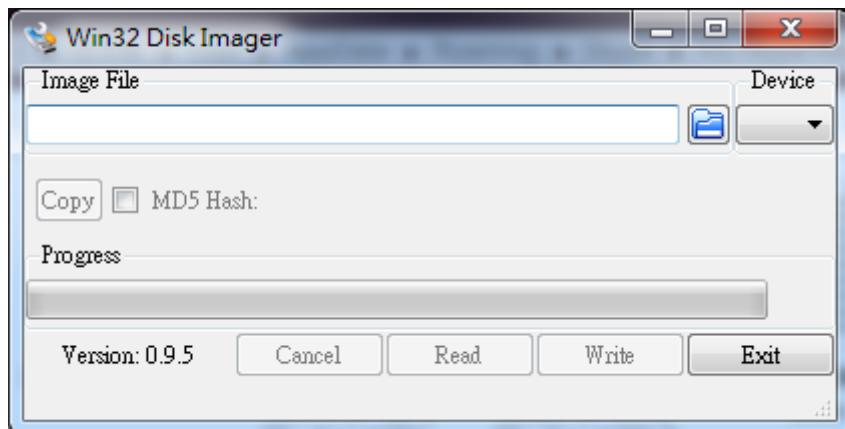
To write the microSD card image into a microSD card, a writing tool is required. In this tutorial, **Win32 Disk Imager** utility is used. This utility is a shareware, users can download it from the web: (<http://sourceforge.net/projects/win32diskimager/>)

On the download page as shown in **Figure 1-2**, click "Download" to start the download process. The downloaded filename is "Win32DiskImager-0.9.5-install.exe".



**Figure 1-2 Download Web Page of Win32 Disk Imager**

Install the Win32DiskImage by executing the installer and execute Win32DiskImage.exe to launch the tool as shown in **Figure 1-3**.



**Figure 1-3 Win32 Disk Imager**

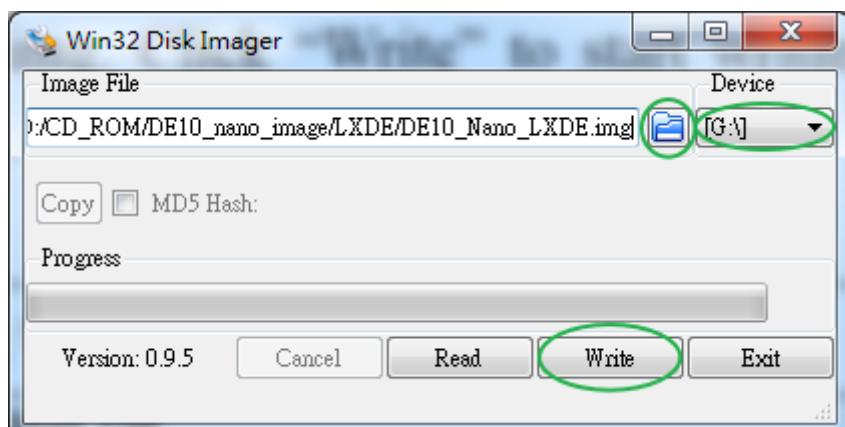
## ■ Create Linux Booting microSD card

The "LXDE Desktop" microSD card image is available from the website:

<http://de10-nano.terasic.com/cd>

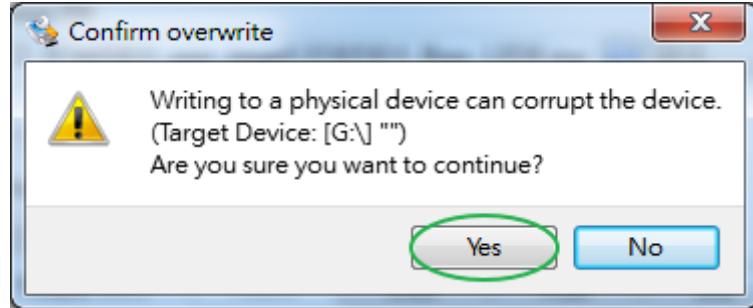
Please download the microSD card image "Linux LXDE Desktop" under the Linux BSP section. The filename is DE10\_Nano\_LXDE.zip. The archive needs to be decompressed to extract the image named DE10\_Nano\_LXDE.img.

Then, insert a 4GB microSD card into the host Windows. Launch Win32 Disk Imager, select the inserted microSD card in the Device list box and click on the Folder icon to select the image file DE10\_Nano\_LXDE.img. Click "Write" to start writing the image file into the microSD card as shown in **Figure 1-4**.



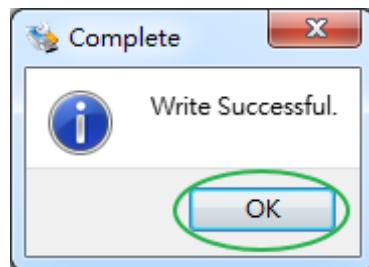
**Figure 1-4 Write microSD card with a Given Image File**

Before writing, a "Confirm overwrite" dialog appears as shown in **Figure 1-5**. Make sure the device is the microSD card, and click "Yes" to close the window.



**Figure 1-5 Confirm to Continue**

When the writing process is complete, a "Complete" dialog appears as shown in **Figure 1-6**. Click "OK" to close the window.

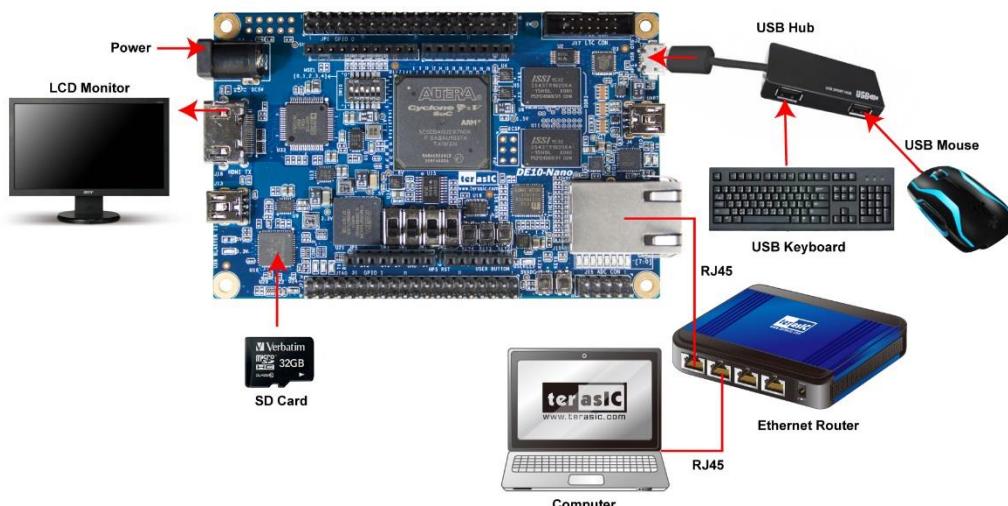


**Figure 1-6 Write SD-card Complete**

## ■ Set up Hardware and Boot Linux

Connect the following items to the DE10-Nano development board as shown in **Figure 1-7**.

- USB mouse and keyboard
- LCD monitor
- Linux LXDE microSD Card
- Ethernet (option, only needed on chapter 6.3 and chapter 8.4)
- DC Power for DE10-Nano



**Figure 1-7 DE10-Nano Development Board Setup**

Please also make sure that MSEL[4:0]=01010 located on the back of the DE10-Nano FPGA development board as shown in

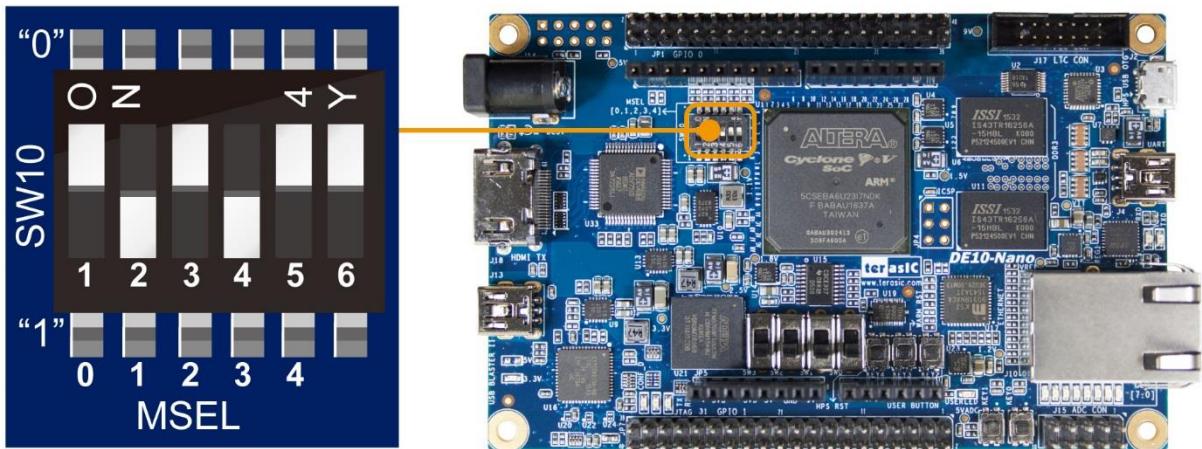


Figure 1-8.

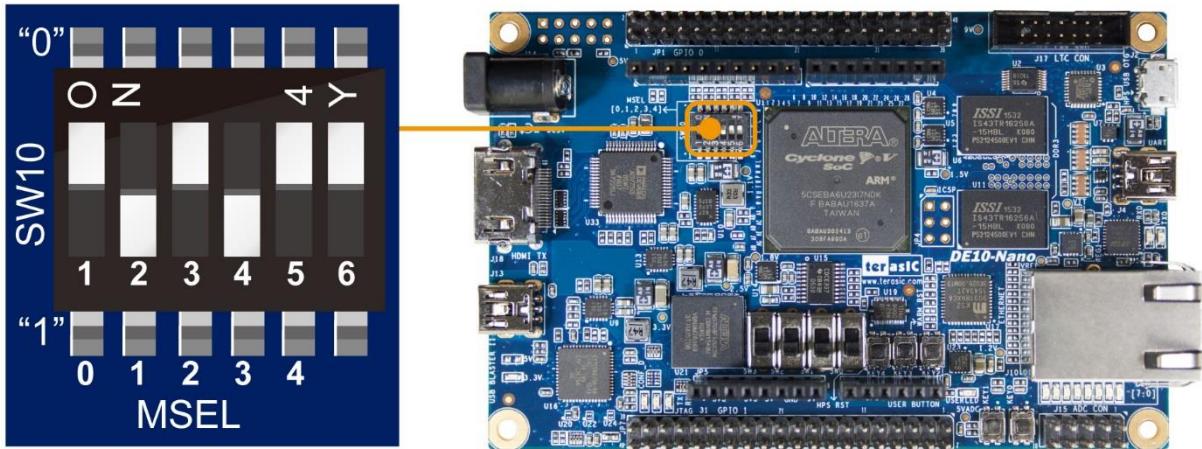
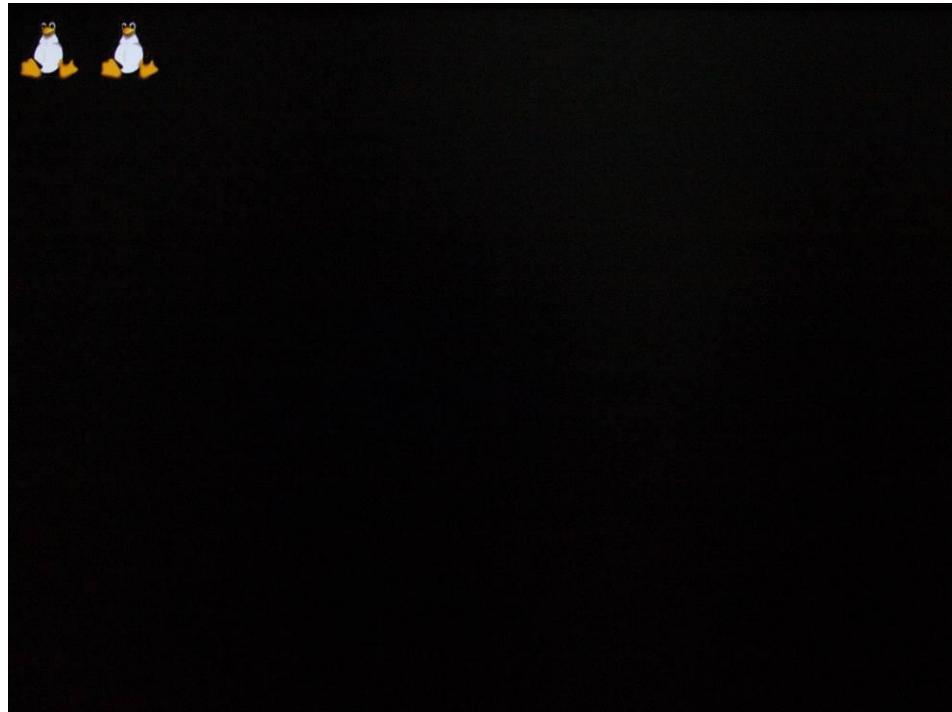


Figure 1-8 MSEL[4:0] on the DE10-Nano FPGA Development Board

After connecting the above peripherals, power on the DE10-Nano board. The LCD monitor will now display Penguin Linux Logo as shown in **Figure 1-9**.



**Figure 1-9 Linux Booting Using Root Account**

When the Linux boot process is complete, LXDE Desktop will appear as shown in **Figure 1-10**.



**Figure 1-10 Launch the Control Panel**

## ■ **Launch Control Panel**

After successful boot, the Control Panel execution file "ControlPanel" is pre-built in the desktop. Please double click "ControlPanel" icon and click "Execute" button to launch the Control Panel as

shown in **Figure 1-11**. Note that Qt library has been preinstalled in the microSD card image therefore users can simply launch the Control Panel without installing the Qt library.

The Control Panel window will appear on the LCD monitor as shown **Figure 1-11**. Now, you can use the USB mouse to operate the Control Panel.



**Figure 1-11 Screenshot of DE10-Nano Control Panel**

# Chapter 2

## *Linux x64 Installation*

The Control Panel program runs on Linux in the Intel SoC ARM as shown in the block diagram in Chapter 1. In Chapter 2, we begin by installing Linux x64 as a first step toward completing the Control Panel project. The Linux x64 system can be installed on a x64 PC directly or on a virtual machine running on 64-bit Microsoft Windows. This chapter also describes how to install an Ubuntu OS, a Linux x64 system, on a virtual machine VMware Workstation Player running on Microsoft Windows. The VMware Workstation Player is first installed on a Windows host, and subsequently Ubuntu Linux is installed on the VMware Workstation Player with an Easy Install mode.

The Control Panel is a GUI-based program and Qt Creator is a convenient tool to help create such GUI-based programs. Regarding the Linux installation, users might ask if it is possible to install Cygwin Linux in Microsoft Windows instead of installing Linux x64 on VMware Workstation Player running on Microsoft Windows. Note that Qt Creator, software meant for GUI-based programs, can't be successfully installed on Cygwin in Microsoft Windows.

Running Linux x64 on VMware Workstation Player in Microsoft Windows is an easier way when users wish to create a GUI application, such as oscilloscopes or logic analyzers, without the need to have a Desktop Linux. Users can simply follow our GUI application development method explained in this tutorial for more user-based GUI applications.

### **2.1 System Requirements**

Before starting the Linux installation on a virtual machine, please make sure and check if these following items are in place:

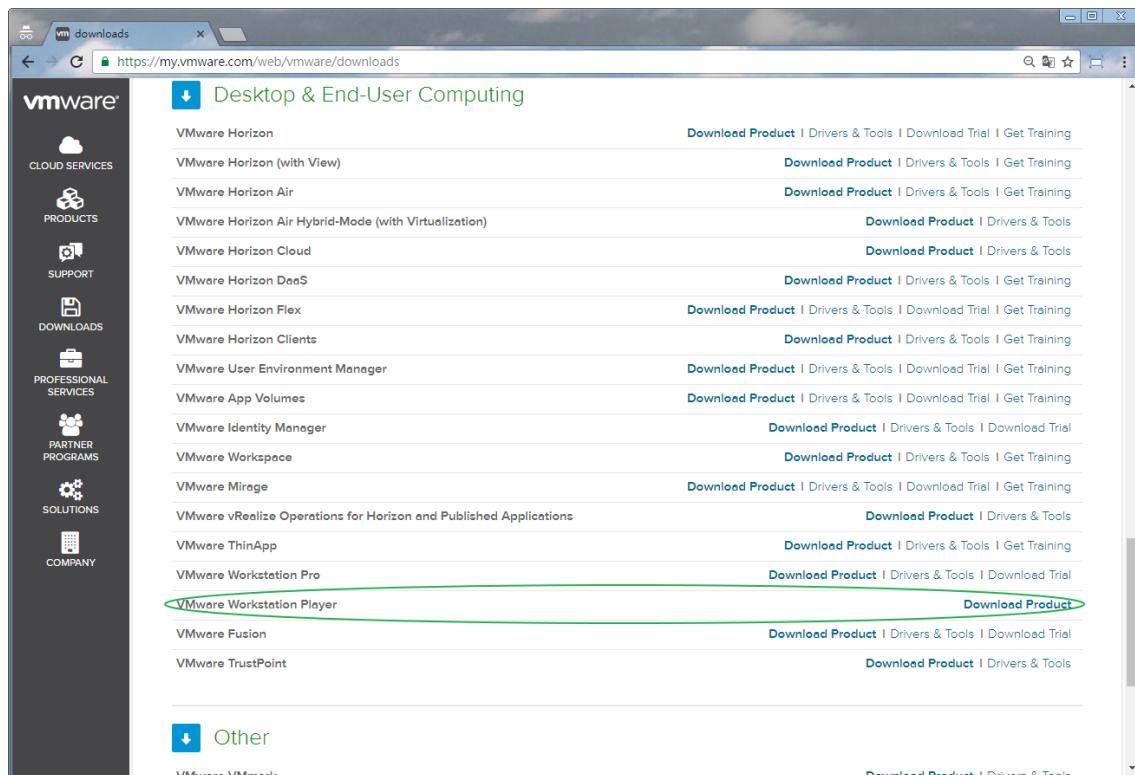
- a PC with 64-bits Microsoft Windows installed
- VMware Workstation Player installer (shareware)
- Linux Ubuntu 16.04 x64 .iso image file

## 2.2 Install VMware Workstation Player

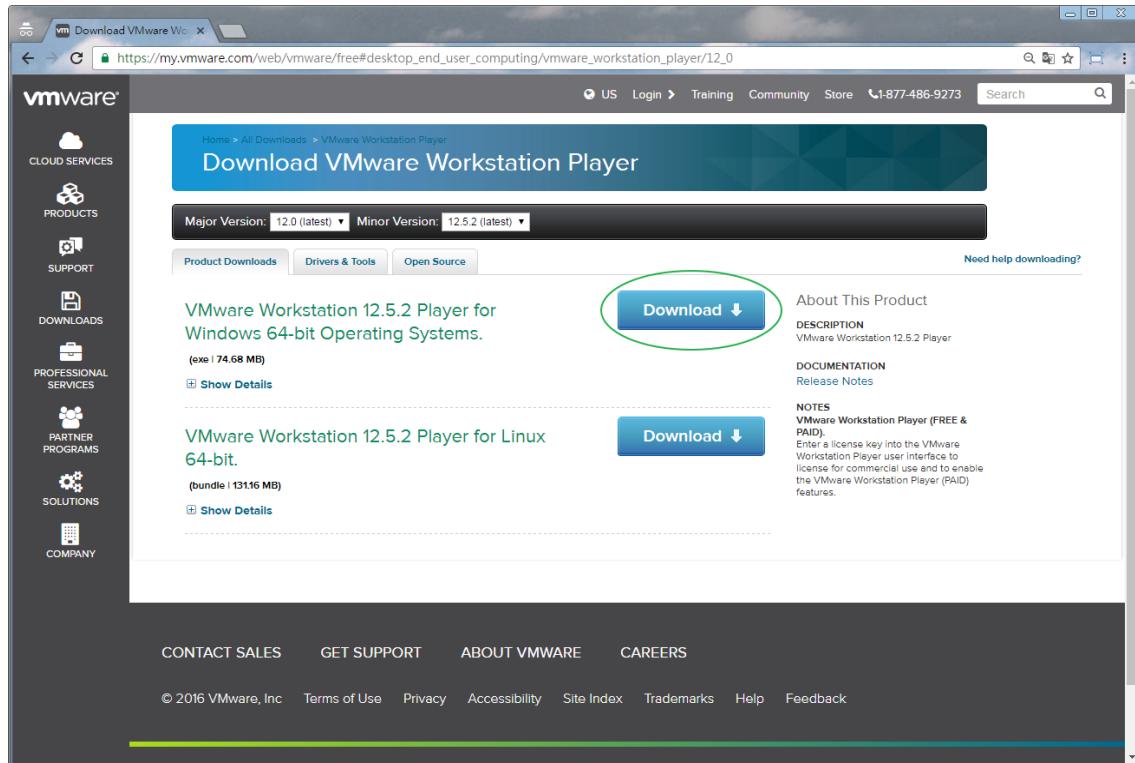
Now, we proceed to install the virtual machine under Microsoft Windows. This section shows (1) where to download VMware Workstation Player and (2) how to install it under Microsoft Windows.

### ■ Download VMware Workstation Player Installer

Go to VMware download web page: <https://my.vmware.com/web/vmware/downloads>, find the VMware Workstation Player item and click "Download Product" as shown in **Figure 2-1**. In the VMware Workstation Player download page, click the "Downloads" button to download VMware Workstation Player, as shown **Figure 2-2**.



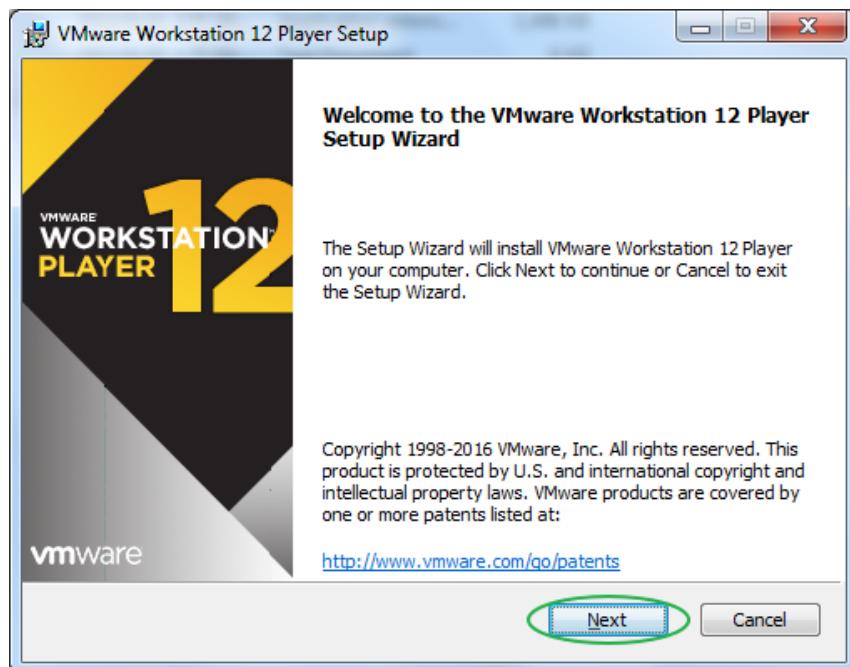
**Figure 2-1 Download Web Page of VMware**



**Figure 2-2 VMware Workstation Player Download Web Page**

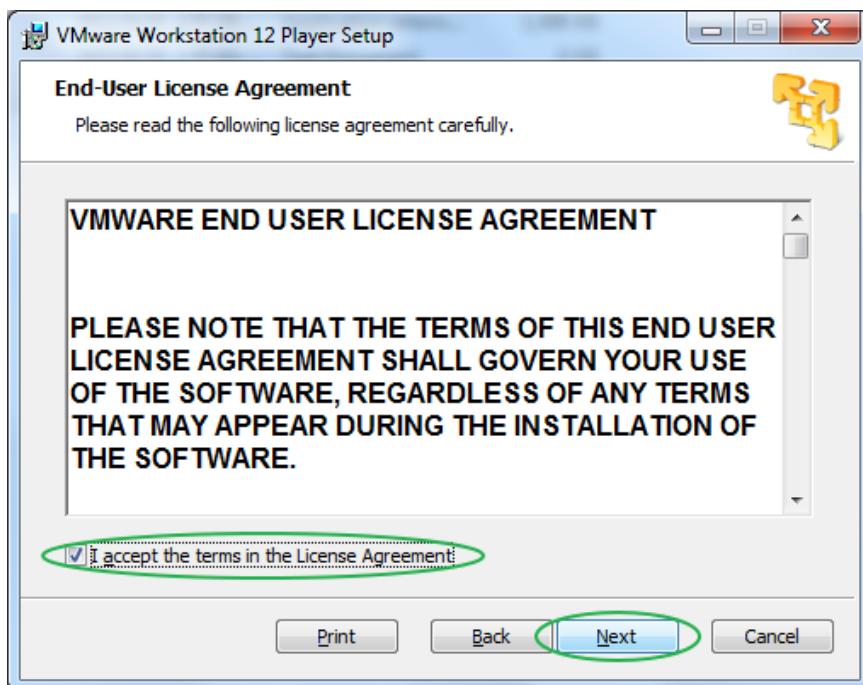
## ■ Install VMware

Under the Windows host, execute the downloaded installer "VMware-player-12.5.2-4638234.exe" to start the setup process. **Figure 2-3** shows the screenshot of the installer. Click "Next >" button to go to the next step.



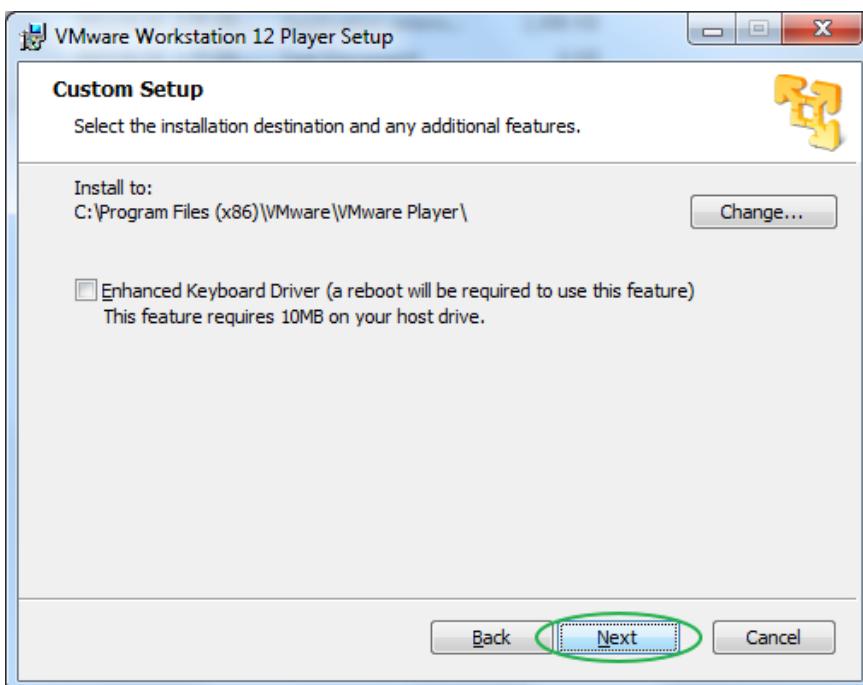
**Figure 2-3 Welcome Dialog**

In the **End-User License Agreement** dialog as shown in [Figure 2-4](#), check "I accept the terms in the license agreement" and click "Next >" button to go to the next step.



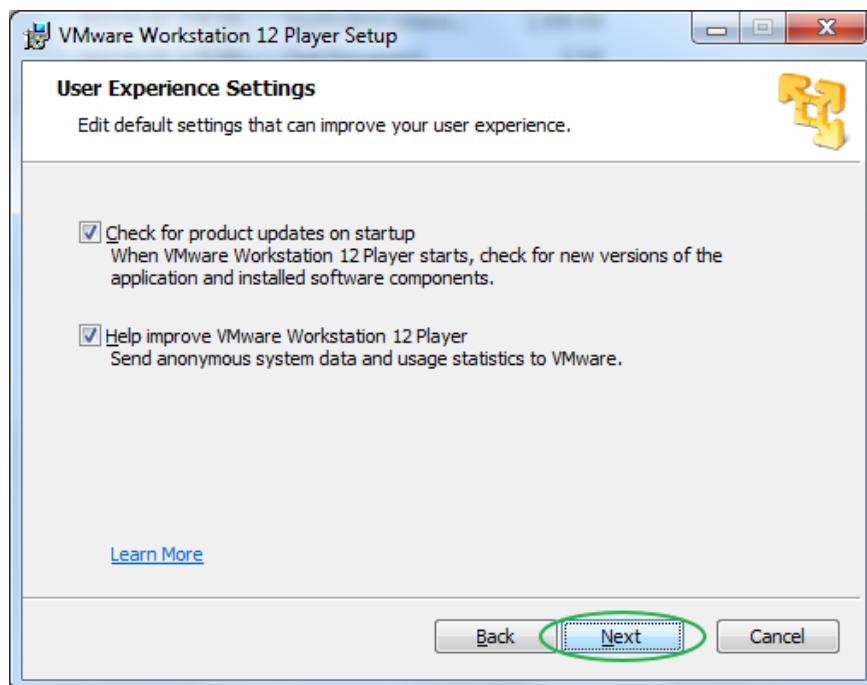
**Figure 2-4 End-User License Agreement Dialog**

In the **Custom Setup** dialog as shown in [Figure 2-5](#), keep the default destination or specify desired installation folder, then click "Next >" button to go to the next step.



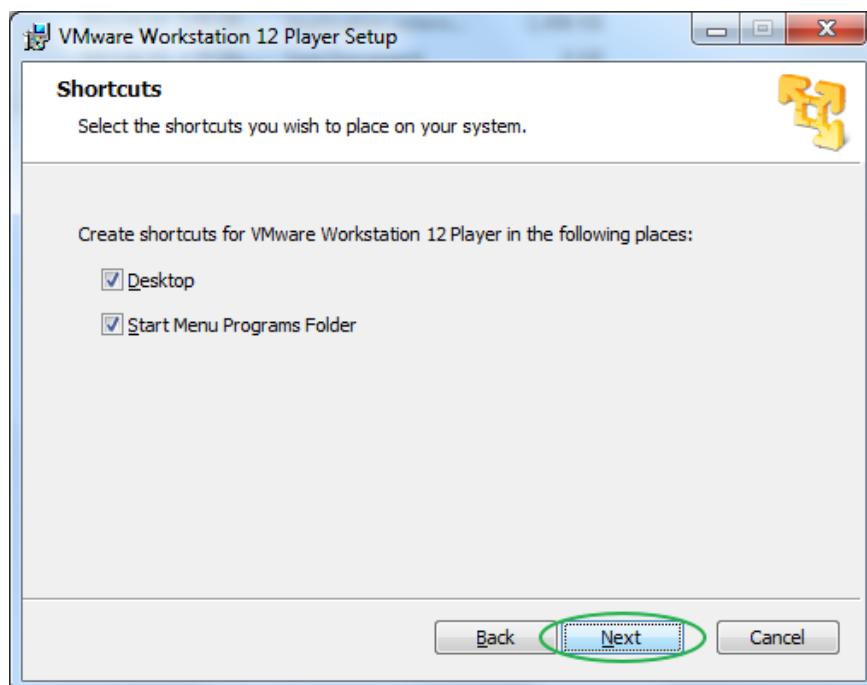
**Figure 2-5 Custom Setup Dialog**

In the **User Experience Settings** dialog as shown in **Figure 2-6**, check on the "Check for product updates on startup" and click "Next" to continue.



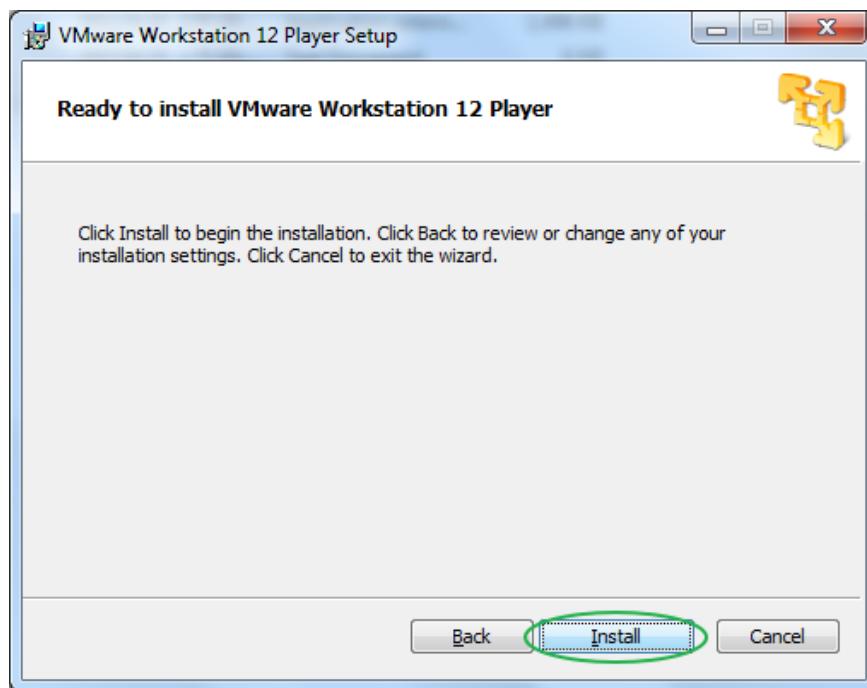
**Figure 2-6 User Experience Settings Dialog**

In the **Shortcuts** dialog as shown in **Figure 2-7**, keep the default setting, then click "Next >" button to go to the next step



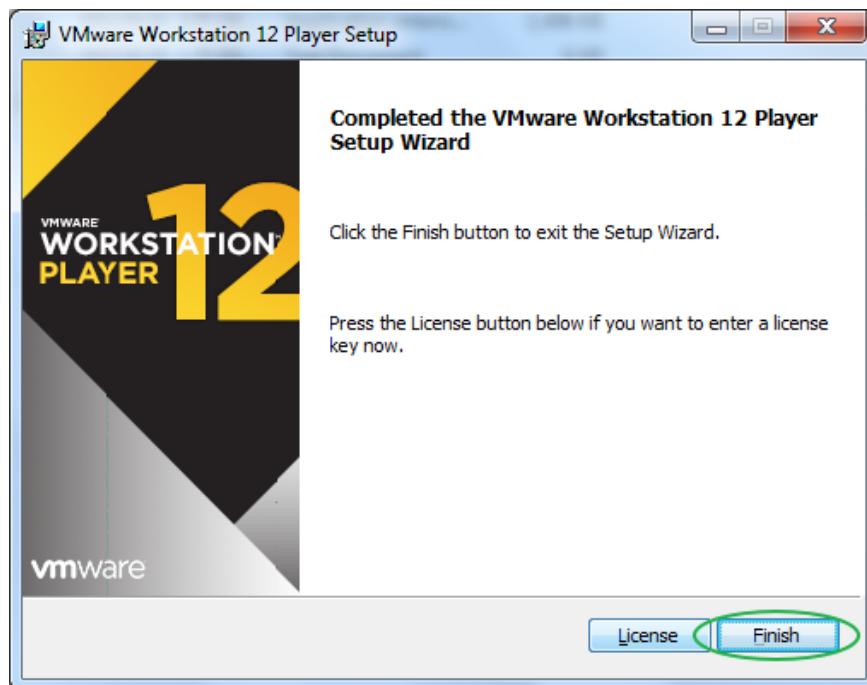
**Figure 2-7 Shortcuts Dialog**

In the **Ready to install VMware Workstation 12 Player** dialog as shown in [Figure 2-8](#), click "Next >" button to go to the next step



**Figure 2-8 Ready to install VMware Workstation 12 Player Dialog**

When the installation is complete, **Completed the VMware Workstation 12 Player Setup Wizard** dialog appears as shown in [Figure 2-9](#). Click "Finished" button to finish the setup process.



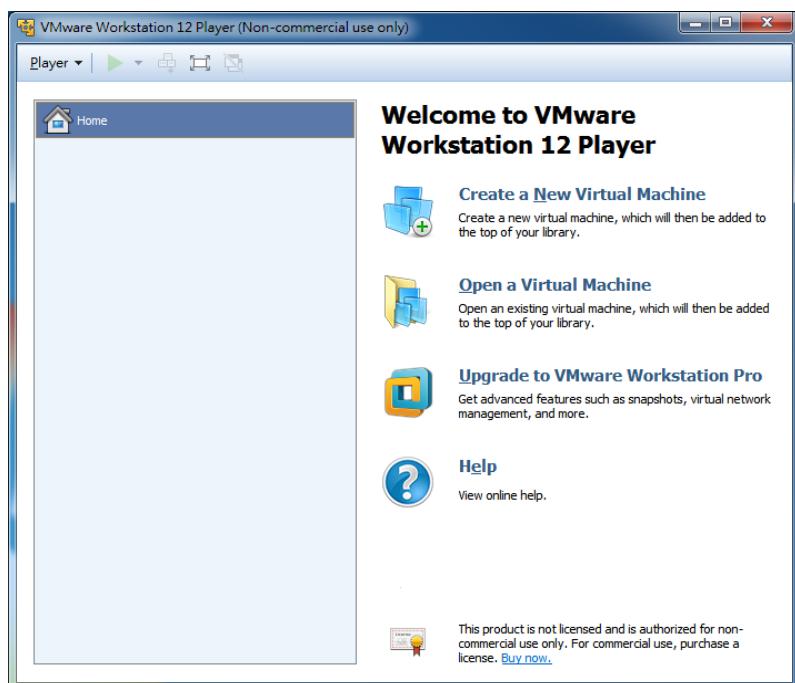
**Figure 2-9 Completed the VMware Workstation 12 Player Setup Wizard Dialog**

## 2.3 Launch VMware

Once VMware Workstation Player is completely installed, a program shortcut icon will appear on the desktop in Windows, as shown in **Figure 2-10**. Double-click the shortcut icon to launch the VMware Workstation Player. **Figure 2-11** shows the main window of VMware Workstation Player.



**Figure 2-10 VMware Workstation Player Shortcut on Windows Desktop**



**Figure 2-11 Main Window of VMware Workstation Player**

## 2.4 Install Linux Ubuntu Desktop

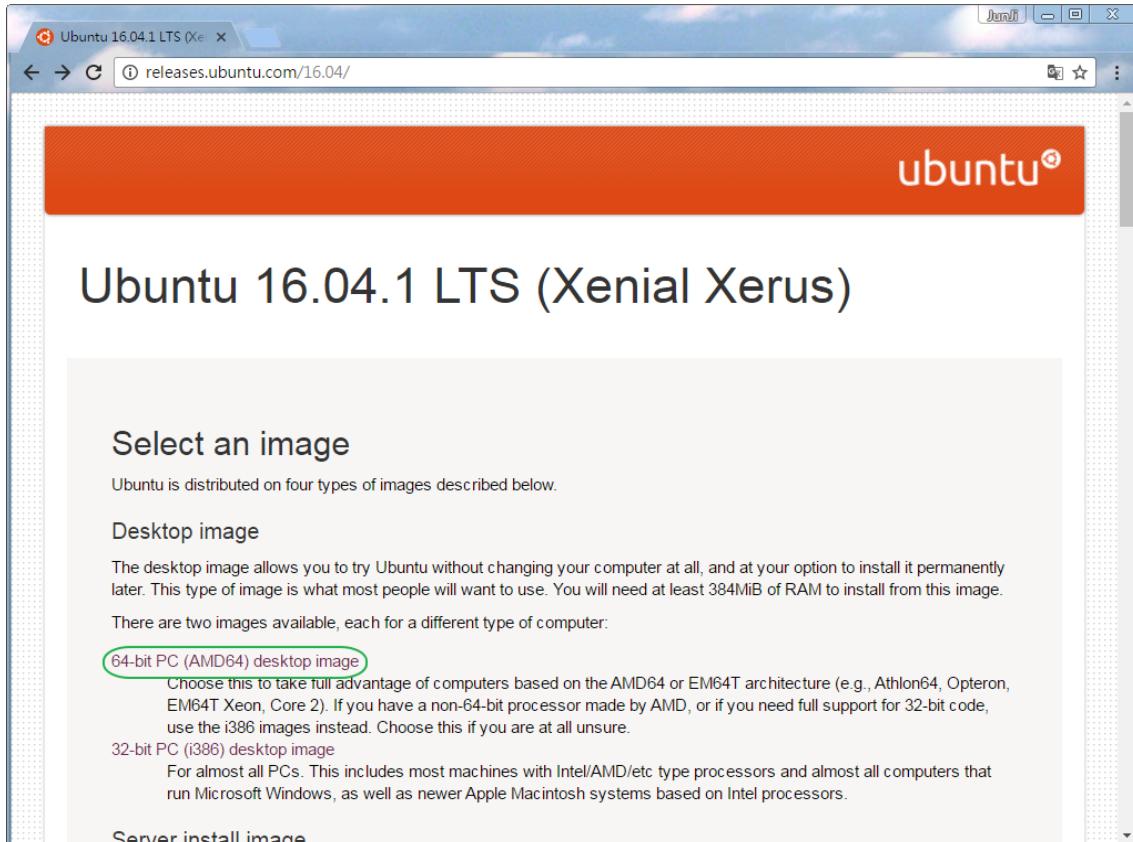
Now we have downloaded and installed the VMware Workstation Player, we are ready to download Ubuntu image file and install it on the VMware Workstation Player. Note that the "Easy Install" mode is automatically applied when VMware Workstation Player detects if the installed OS is Ubuntu 16.04.

### ■ Download Ubuntu Linux Image

The Ubuntu Desktop can be downloaded from the web link provided below, as shown in **Figure 2-12**

<http://releases.ubuntu.com/16.04/>

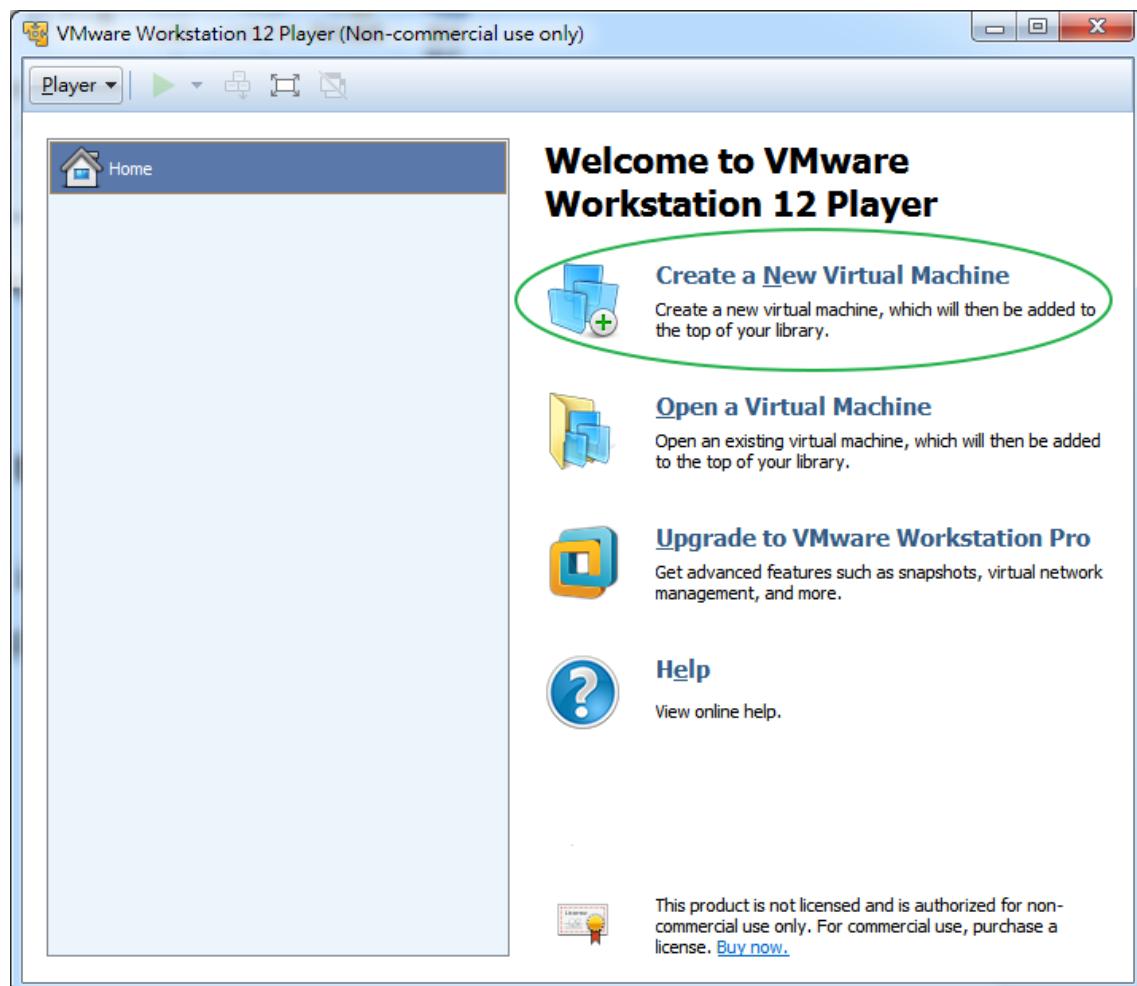
On the download page, click "64-bit PC (AMD64) desktop image" link to start the download process. The downloaded image filename is "**ubuntu-16.04.1-desktop-amd64.iso**".



**Figure 2-12 Download Page of Ubuntu Desktop**

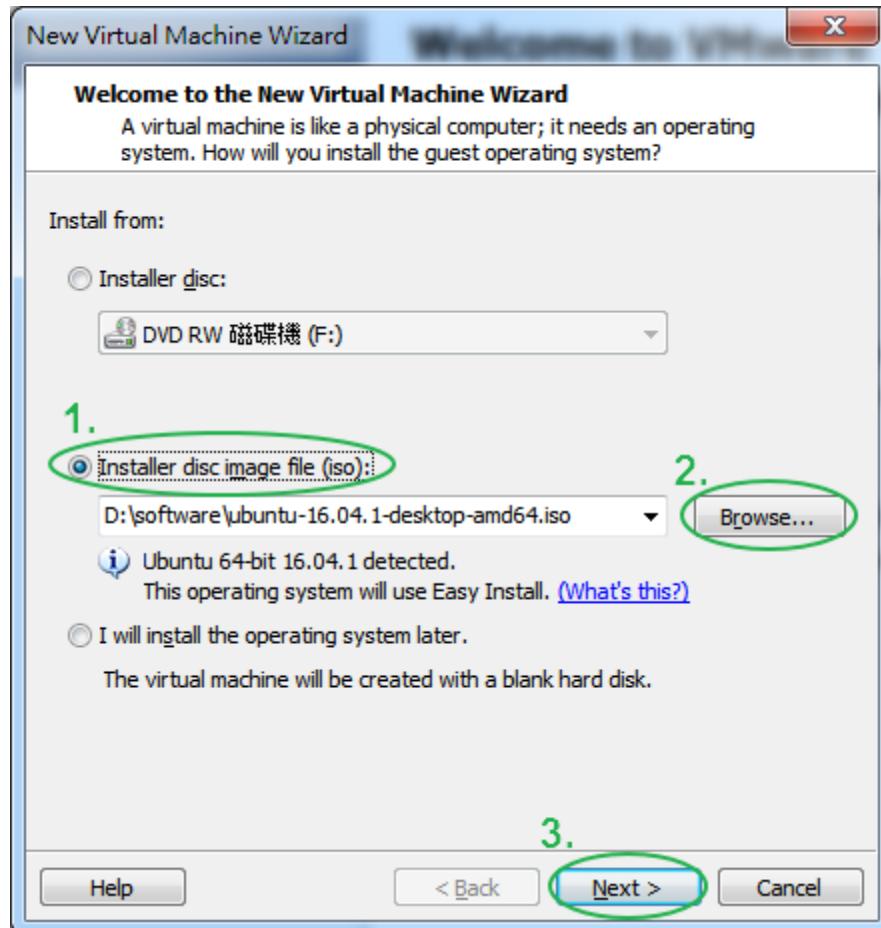
## ■ Create a Ubuntu Virtual Machine on the VMware Workstation Player

Launch the VMware Workstation Player as shown in **Figure 2-13**. Click the "Create a New Virtual Machine" icon to create a virtual machine accordingly.



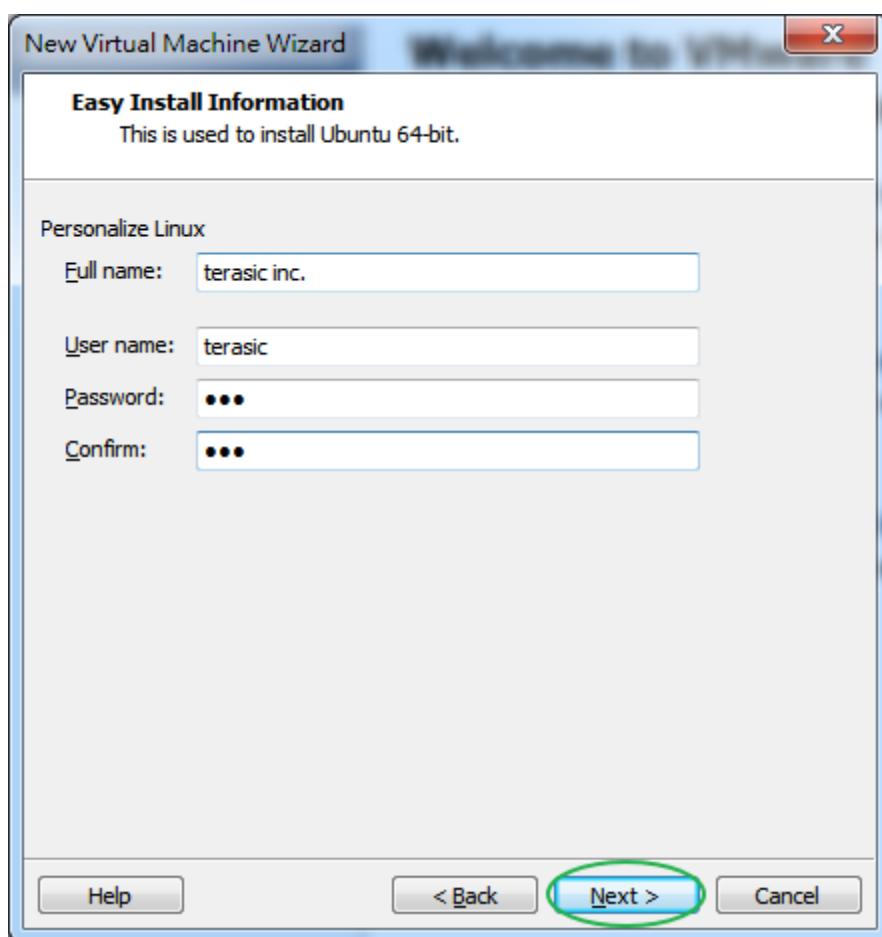
**Figure 2-13** VMware Workstation Player Main Window

When the **Welcome to the New Virtual Machine Wizard** dialog appears as shown in **Figure 2-14**, select the "installer disk image file (iso):" radio button, and click "Browse" button to specify the location of the Ubuntu Linux image file "**ubuntu-16.04.1-desktop-amd64.iso**" which has been downloaded in the previous step. Note that the "Easy Install" mode is enabled for Ubuntu 16.04.1. Click "Next >" to go to the next step.



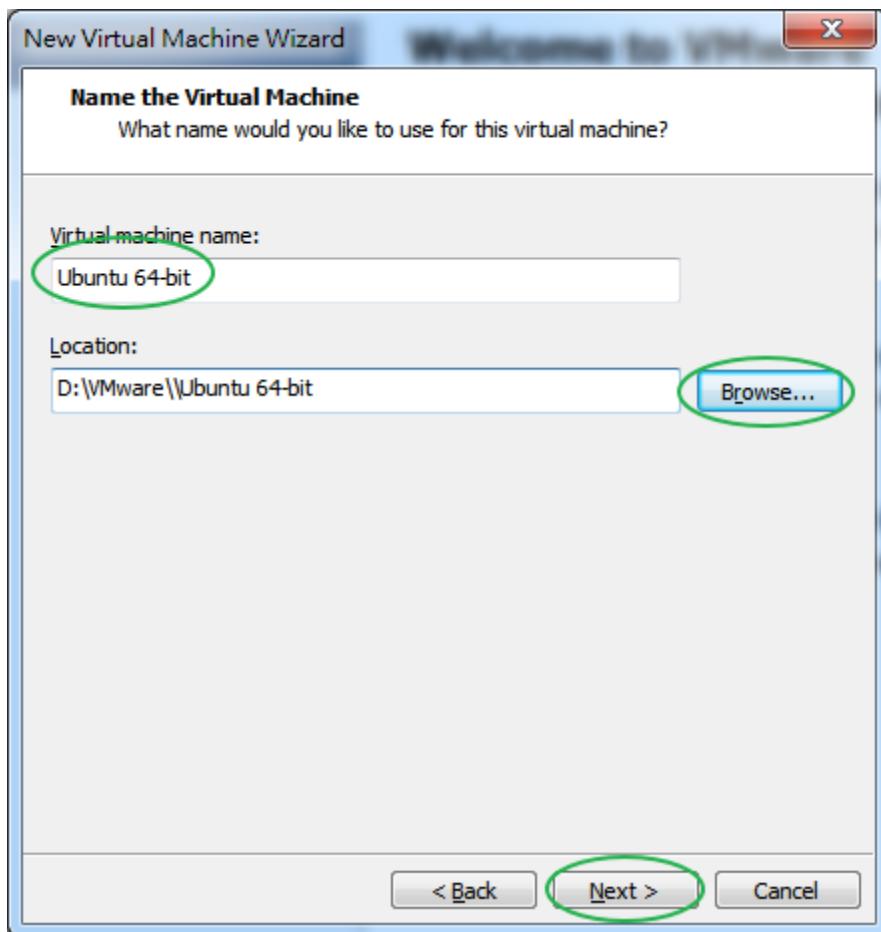
**Figure 2-14 Install Dialog**

When the **Easy Install Information** dialog appears as shown in **Figure 2-15**, please specify your username and password. In this tutorial, the user name "terasic" and the password "123" are used. Click "Next >" button to go to the next step. **Please remember your username and password, as they will be frequently used throughout the tutorial.**



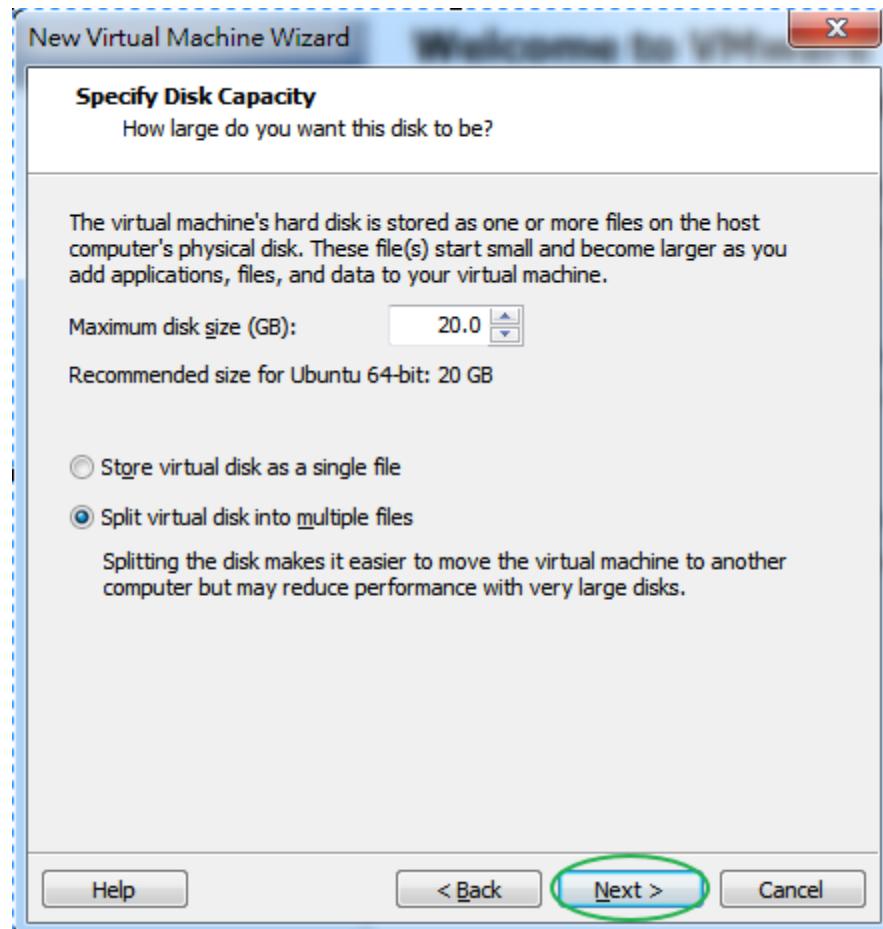
**Figure 2-15 Easy Install Information Dialog**

When the **Name the Virtual Machine** dialog appears as shown in **Figure 2-16**, you can change the virtual machine name as you wish. In this example, a machine name "Ubuntu 64-bit" is used. Click the "Browse" button to specify a folder location for the virtual disk of the virtual machine. Finally, click "Next >" button to go to the next step.



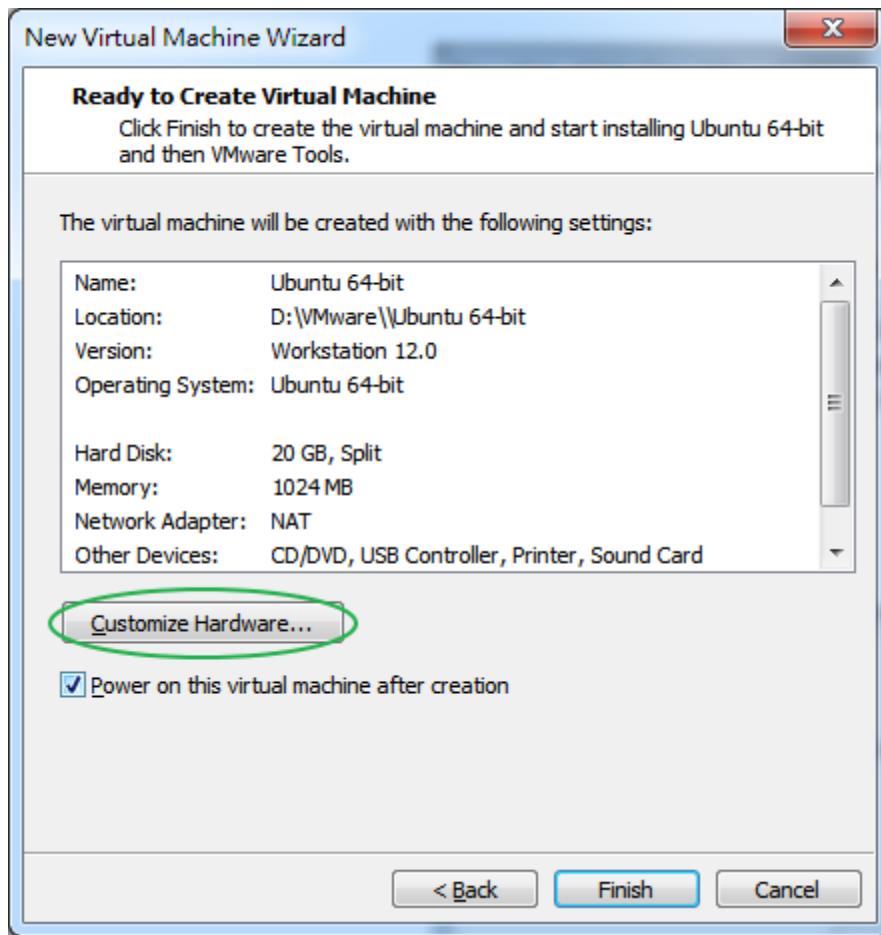
**Figure 2-16 Name the Virtual Machine Dialog of the New Virtual Machine Wizard**

When the **Specify Disk Capacity** dialog appears as shown in **Figure 2-17**, please specify the maximum disk size and click "Next >" button to go to the next step.



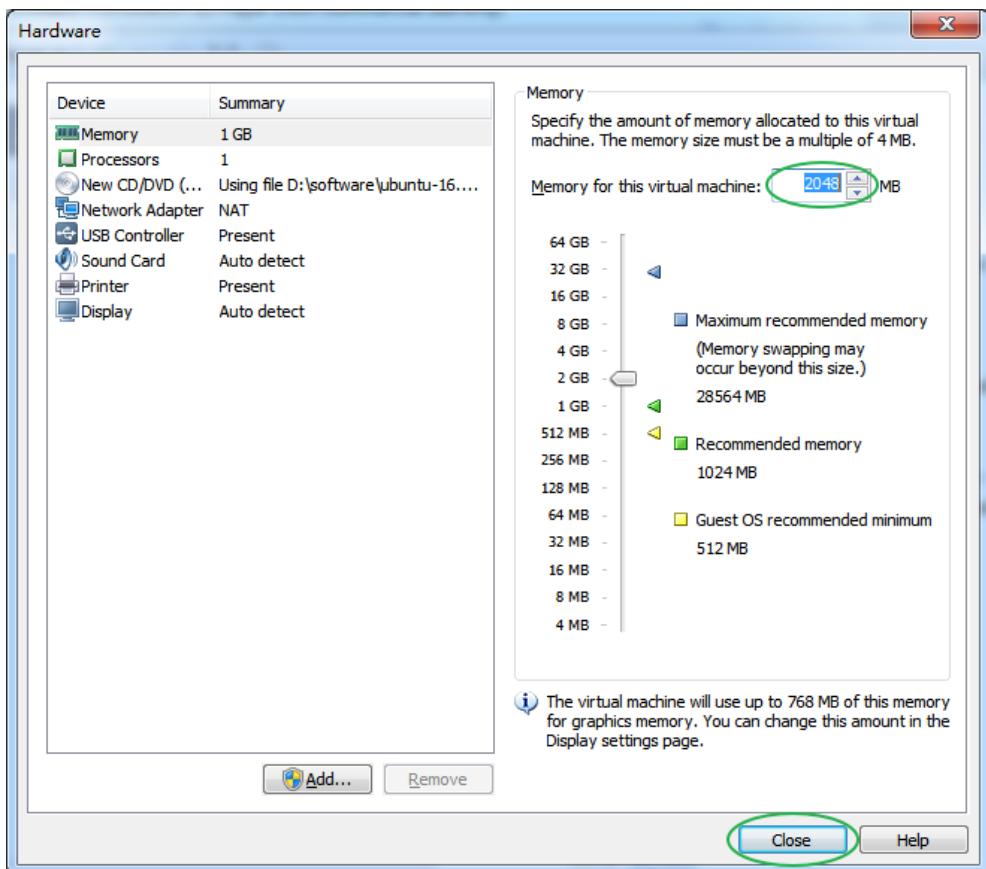
**Figure 2-17 Specify Disk Capacity Dialog**

When the **Ready to Create Virtual Machine** dialog appears as shown in **Figure 2-18**. Click the "Customize Hardware ..." button to modify memory allocated to the virtual machine.



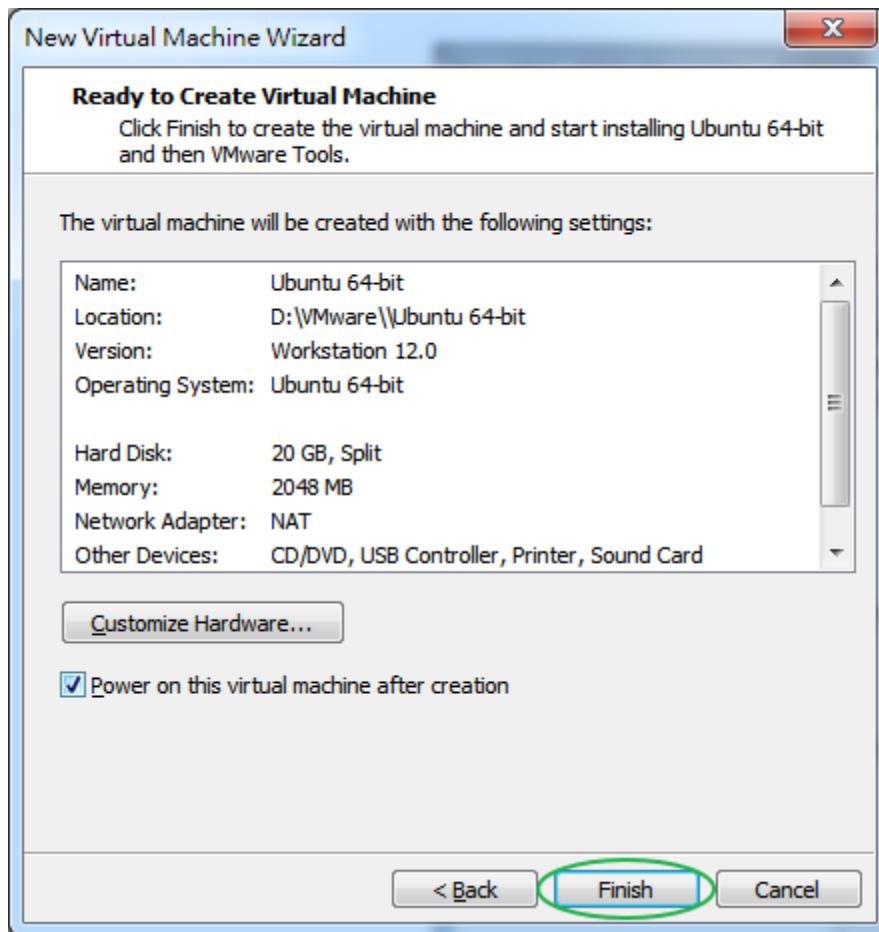
**Figure 2-18 Ready to Create Virtual Machine Dialog**

The **Hardware** dialog appears as shown in **Figure 2-19**. Modify memory for this virtual machine to 2048MB or more. Then, click "Close " button to return to the **Ready to Create Virtual Machine** dialog.



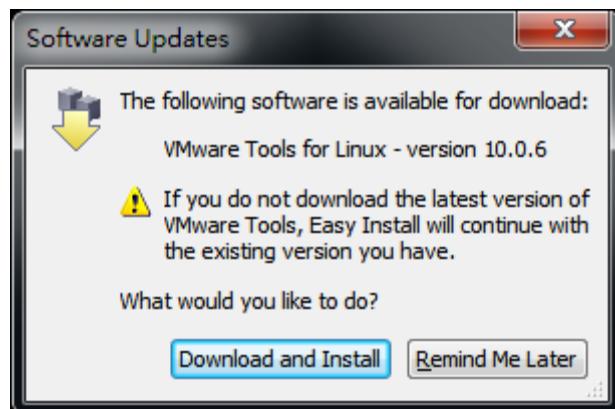
**Figure 2-19 Hardware Dialog**

Finally, the **Ready to Create Virtual Machine** dialog appears as shown in [Figure 2-20](#). Click the "Finish" button to start installation progress.



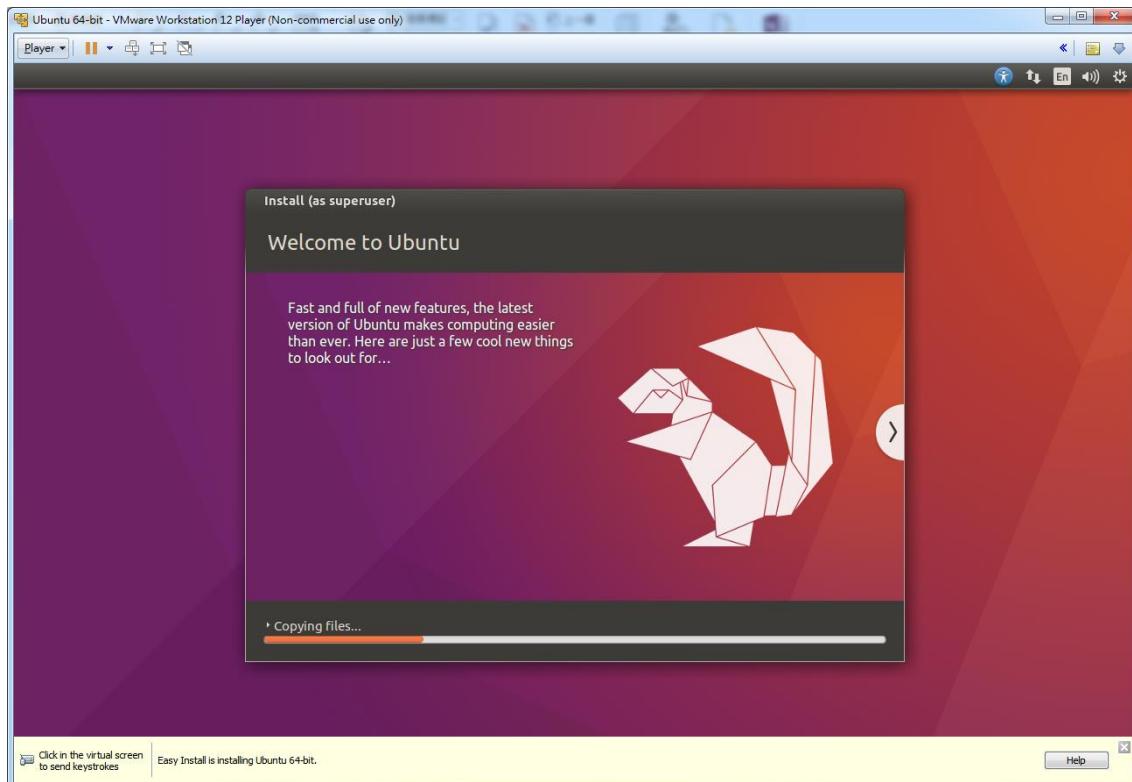
**Figure 2-20 Ready to Create Virtual Machine Dialog - Modify Memory**

While installing, the **Software Updates** dialog appears as shown in [Figure 2-21](#). Click the "Download and Install" button to proceed.



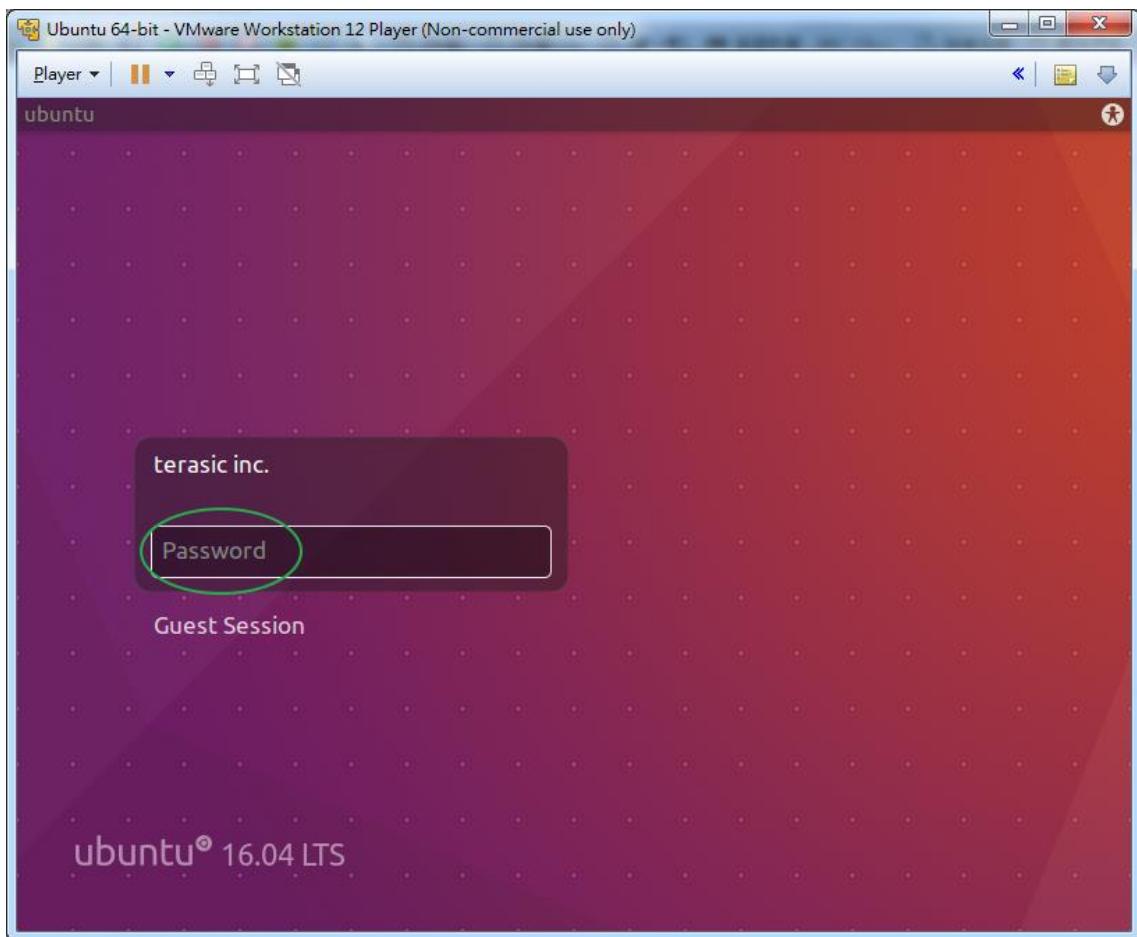
**Figure 2-21 Software Updates Dialog**

**Figure 2-22** shows a screenshot of the installation progress.



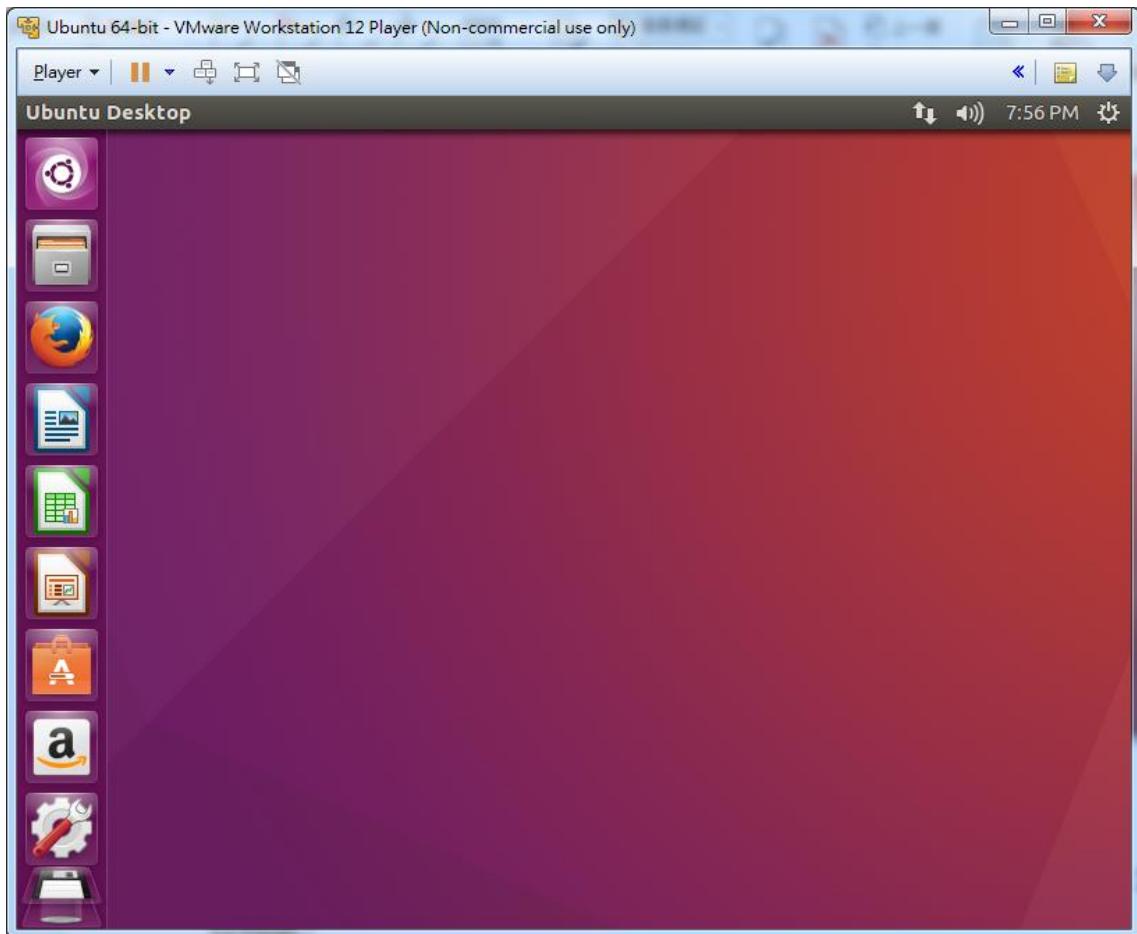
**Figure 2-22 Installation Progress**

When the installation is completed, Ubuntu login dialog appears as shown in **Figure 2-23**. In the Password edit box, key in the password specified in the previous step (in this tutorial, password is "123"), and press "ENTER" on the keyboard.



**Figure 2-23 Ubuntu Login**

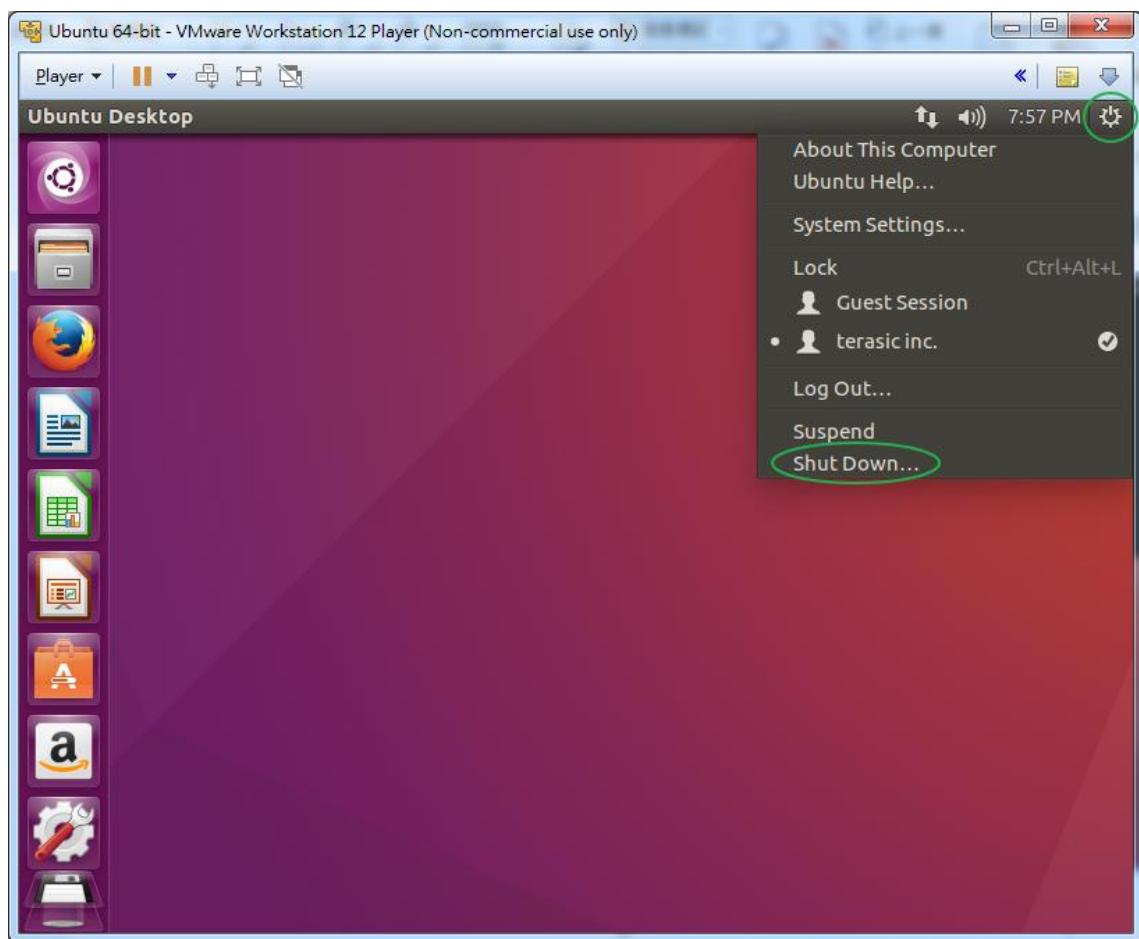
After successful login, Ubuntu Desktop appears as shown in **Figure 2-24**.



**Figure 2-24 Ubuntu Desktop**

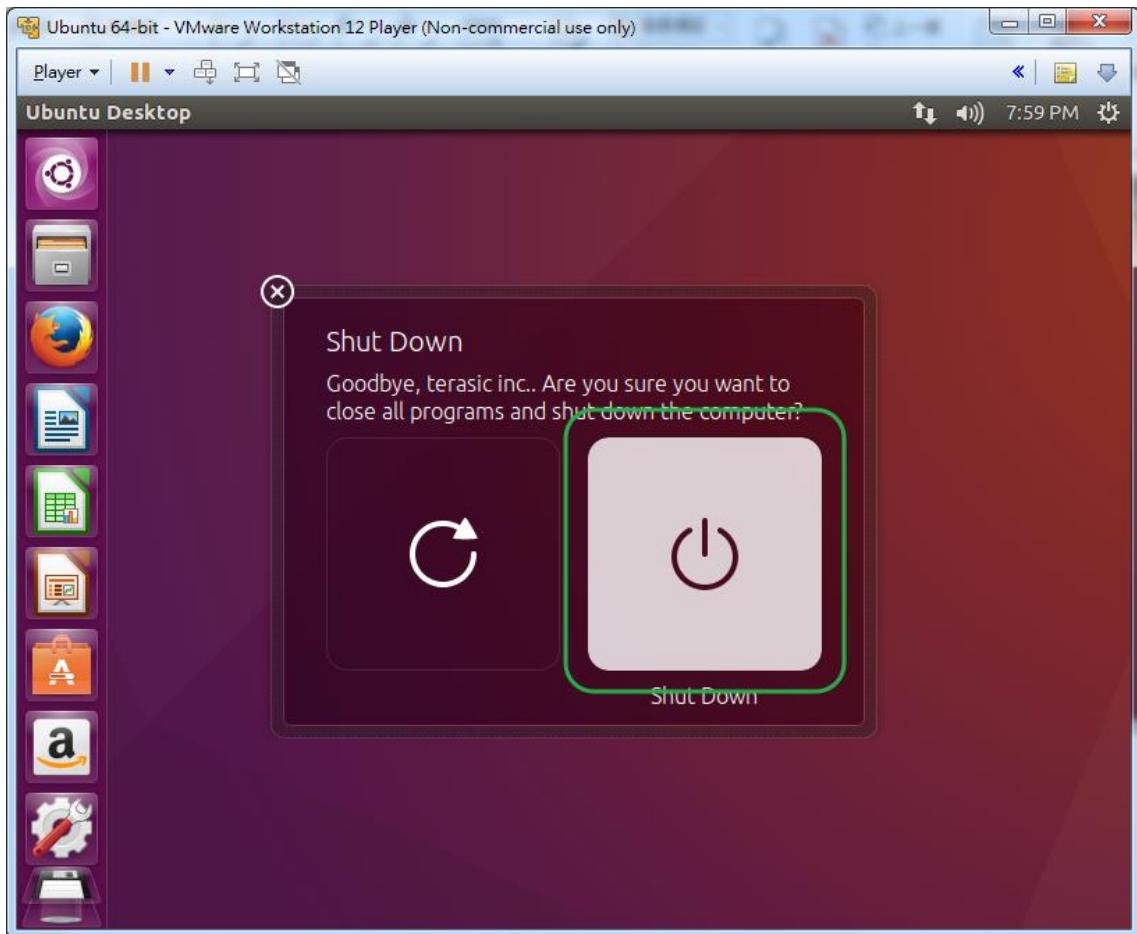
## ■ Shut Down Linux System

Click the Power icon and select the "Shut Down..." as shown in **Figure 2-25**.



**Figure 2-25 Shut Down Ubuntu Desktop - 1**

Then, select "Shut Down" as shown in **Figure 2-26** to turn off the Linux system.

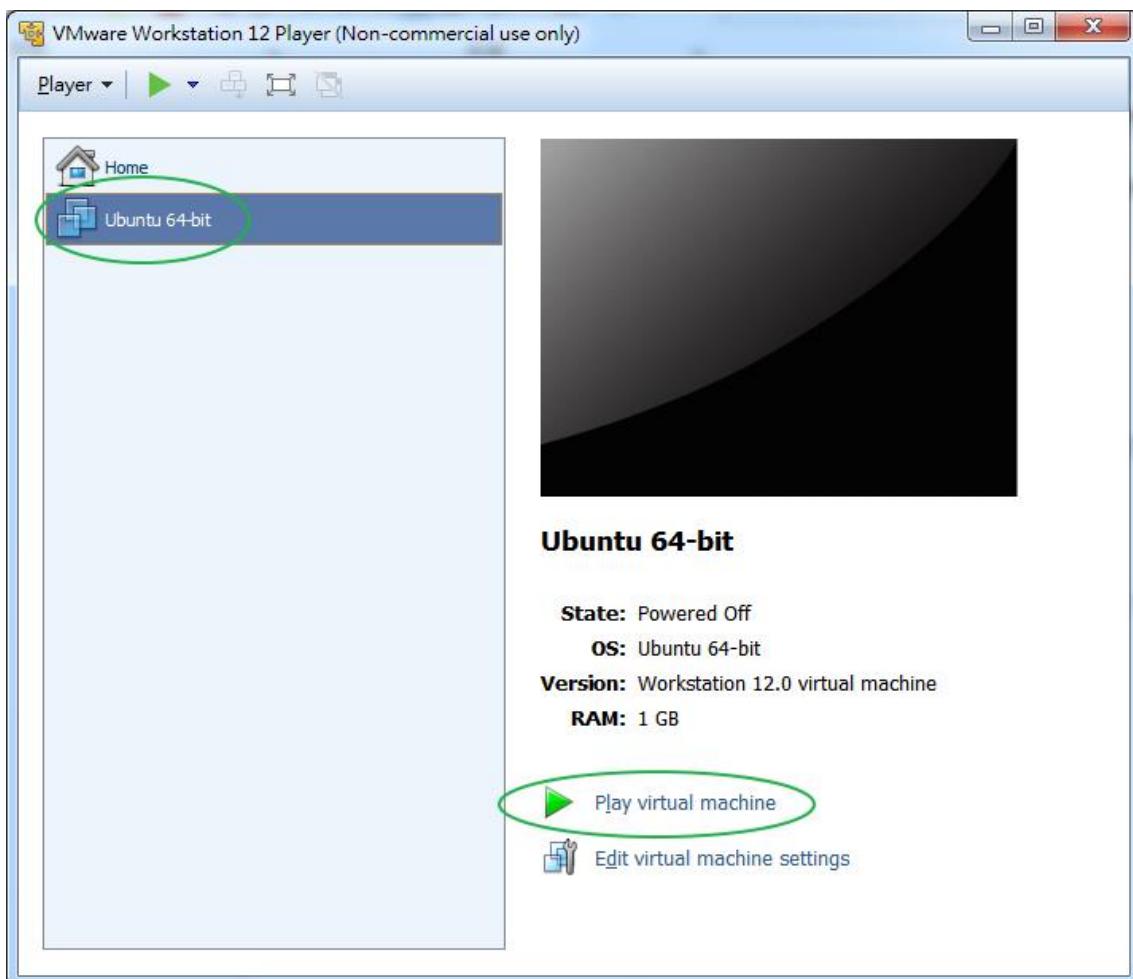


**Figure 2-26 Shut Down Ubuntu Desktop - 2**

## ■ Restart Ubuntu

To restart Ubuntu after it has been shut down, here is the step-by-step instruction:

1. Launch VMWare Workstation Player
2. Select the "Ubuntu 64-bit" item and click "Play virtual machine" button to start Ubuntu as shown in **Figure 2-27**.



**Figure 2-27 Start Ubuntu**

## 2.5 Upgrade Linux x64 Software Package

After Linux x64 has been completely installed, we also need to upgrade the system to make sure the system is the most up-to-date.

Please follow carefully with the installation procedures in this section. First, press "CTRL+ALT+T" on keyboard to launch a terminal as show in **Figure 2-28**.

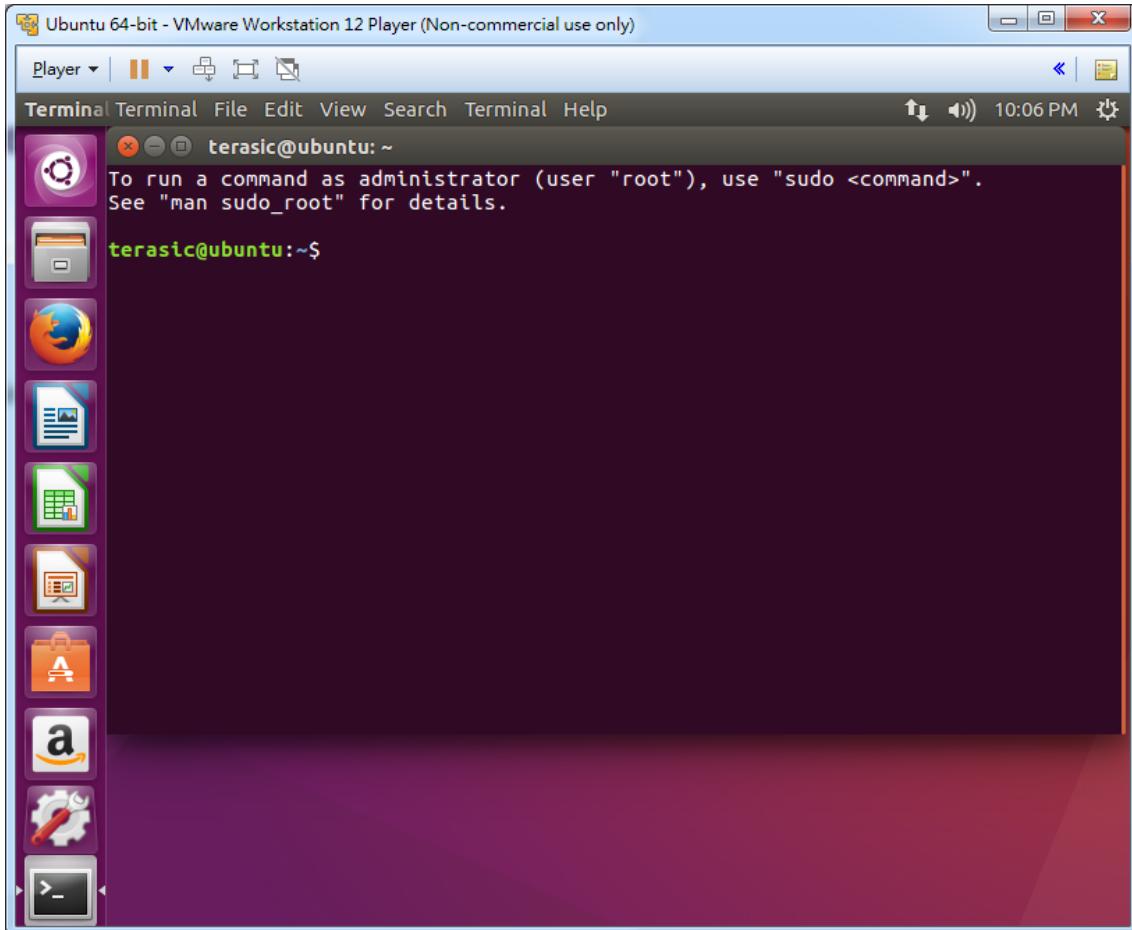
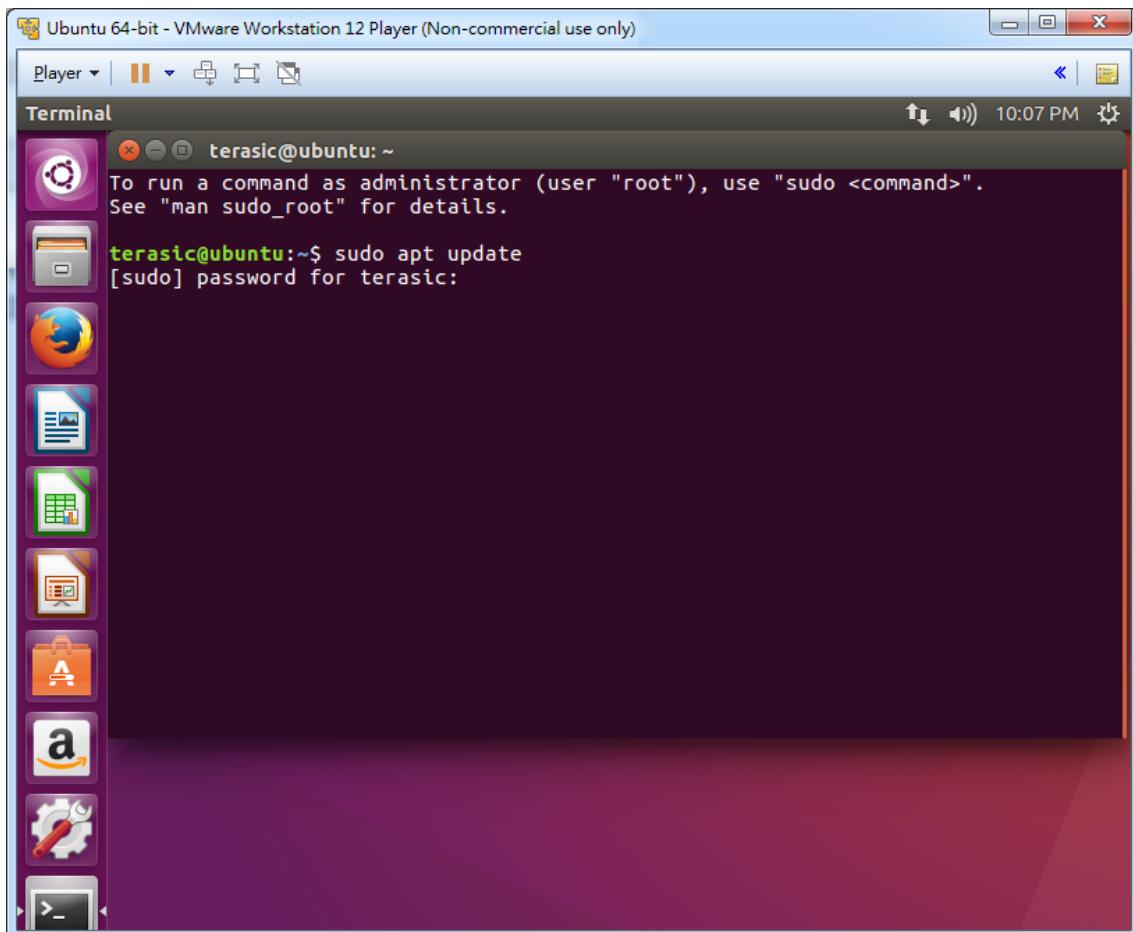


Figure 2-28 Linux Terminal

In the terminal, type the following command and press ENTER.

```
$ sudo apt update
```

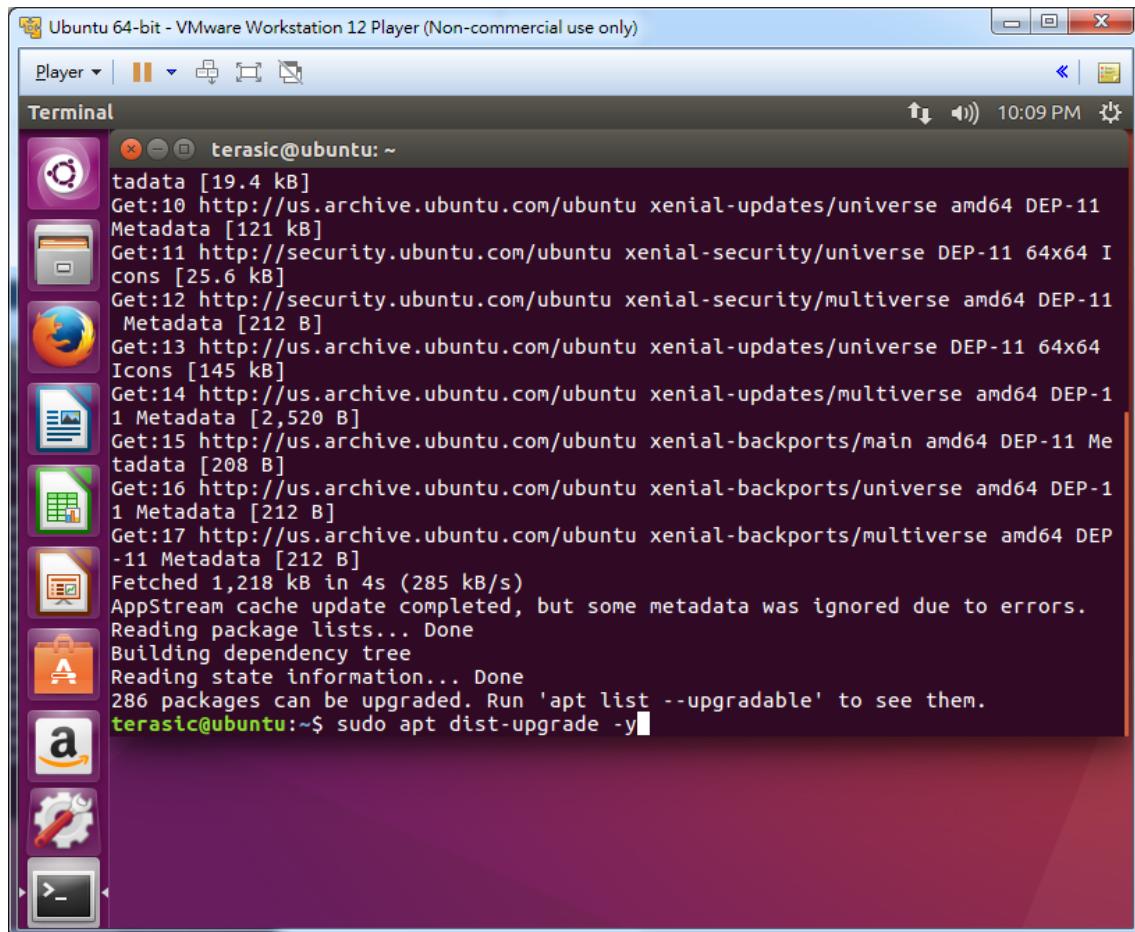
The system will prompt users to input a password. Please input your password (in this tutorial, the password is "123") and press ENTER as shown in **Figure 2-29**.



**Figure 2-29 Input password for terasic**

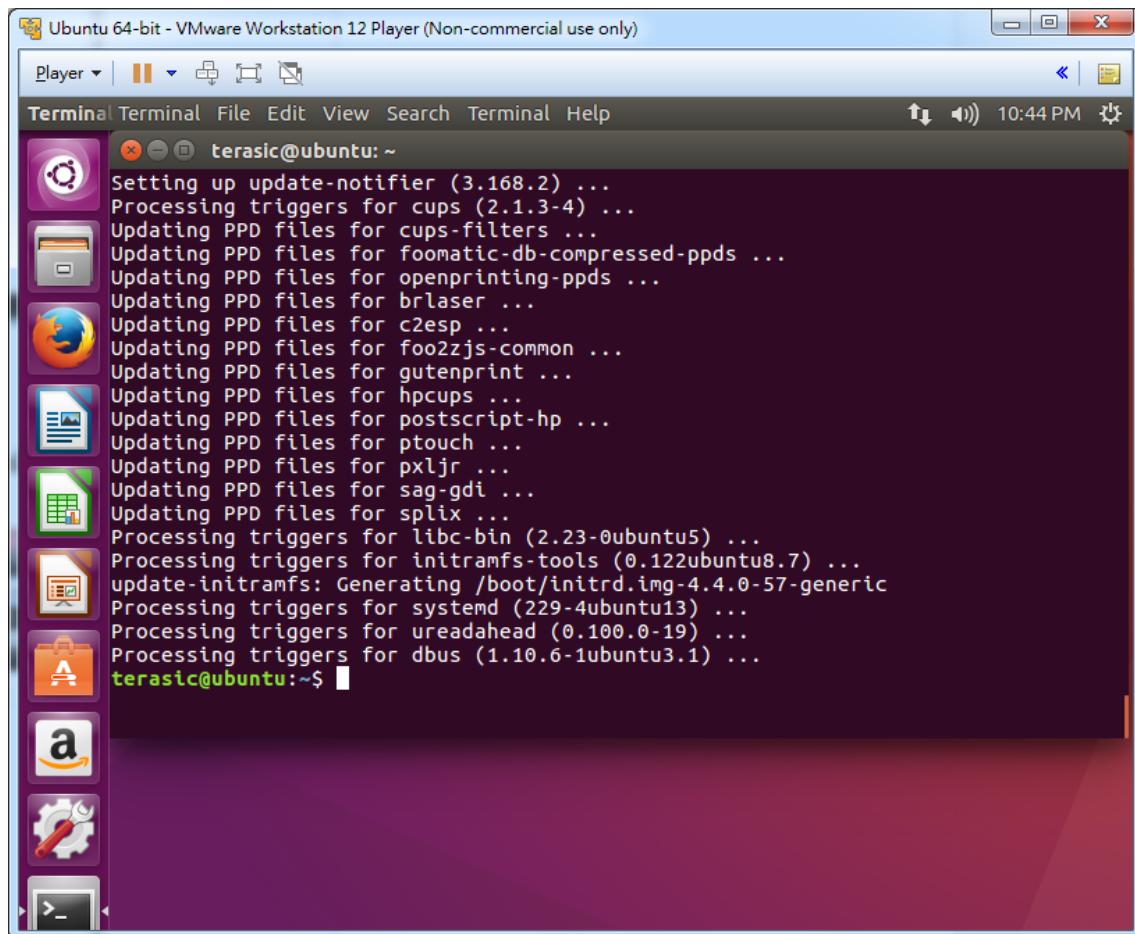
After finishing typing in "sudo apt update" and the password, now type in the following command and press ENTER, as shown in **Figure 2-30**. The upgrade process will take about 10~15 minutes.

```
$ sudo apt dist-upgrade -y
```



**Figure 2-30 Typing in the Command "sudo apt dist-upgrade -y"**

**Figure 2-31** shows the screenshot after the upgrade process has been complete. Type "exit" to close the terminal.



The screenshot shows a terminal window titled "Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)". The window contains a list of upgrade logs:

```
terasic@ubuntu: ~
Setting up update-notifier (3.168.2) ...
Processing triggers for cups (2.1.3-4) ...
Updating PPD files for cups-filters ...
Updating PPD files for foomatic-db-compressed-ppds ...
Updating PPD files for openprinting-ppds ...
Updating PPD files for brlaser ...
Updating PPD files for c2esp ...
Updating PPD files for foo2js-common ...
Updating PPD files for gutenprint ...
Updating PPD files for hpcups ...
Updating PPD files for postscript-hp ...
Updating PPD files for ptouch ...
Updating PPD files for pzljr ...
Updating PPD files for sag-gdi ...
Updating PPD files for splix ...
Processing triggers for libc-bin (2.23-0ubuntu5) ...
Processing triggers for initramfs-tools (0.122ubuntu8.7) ...
update-initramfs: Generating /boot/initrd.img-4.4.0-57-generic
Processing triggers for systemd (229-4ubuntu13) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for dbus (1.10.6-1ubuntu3.1) ...
terasic@ubuntu:~$
```

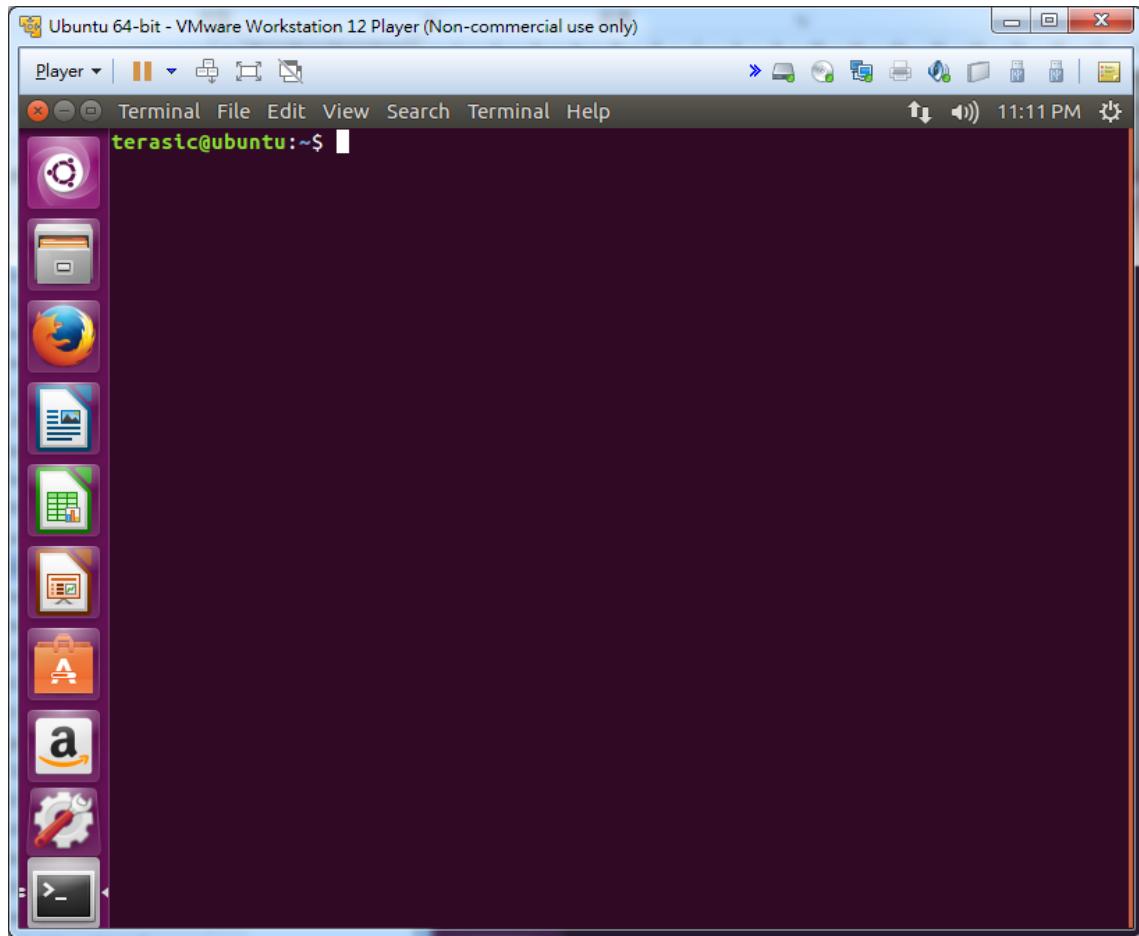
**Figure 2-31 Upgrade Process Complete**

## 2.6 Enable VMware Share Folder

After Linux system has been upgraded, we also need to copy files from Windows host by using VMware Workstation Player's **Shared Folder** features. Note, a software patch is applied to fix known bugs.

### ■ Fix and Enable VMware Workstation Player's Shared Folder features

Please follow carefully with the installation procedures in this section. First, press "CTRL+ALT+T" on keyboard to launch a terminal as show in **Figure 2-32**.



**Figure 2-32 Linux Terminal**

In the terminal, execute the following commands to download and install VMware Tools Patches.

```
$ wget https://codeload.github.com/rasa/vmware-tools-patches/zip/master \
-O vmware-tools-patches.zip
$ unzip vmware-tools-patches.zip
$ cd vmware-tools-patches-master/
$ sudo ./patched-open-vm-tools.sh
```

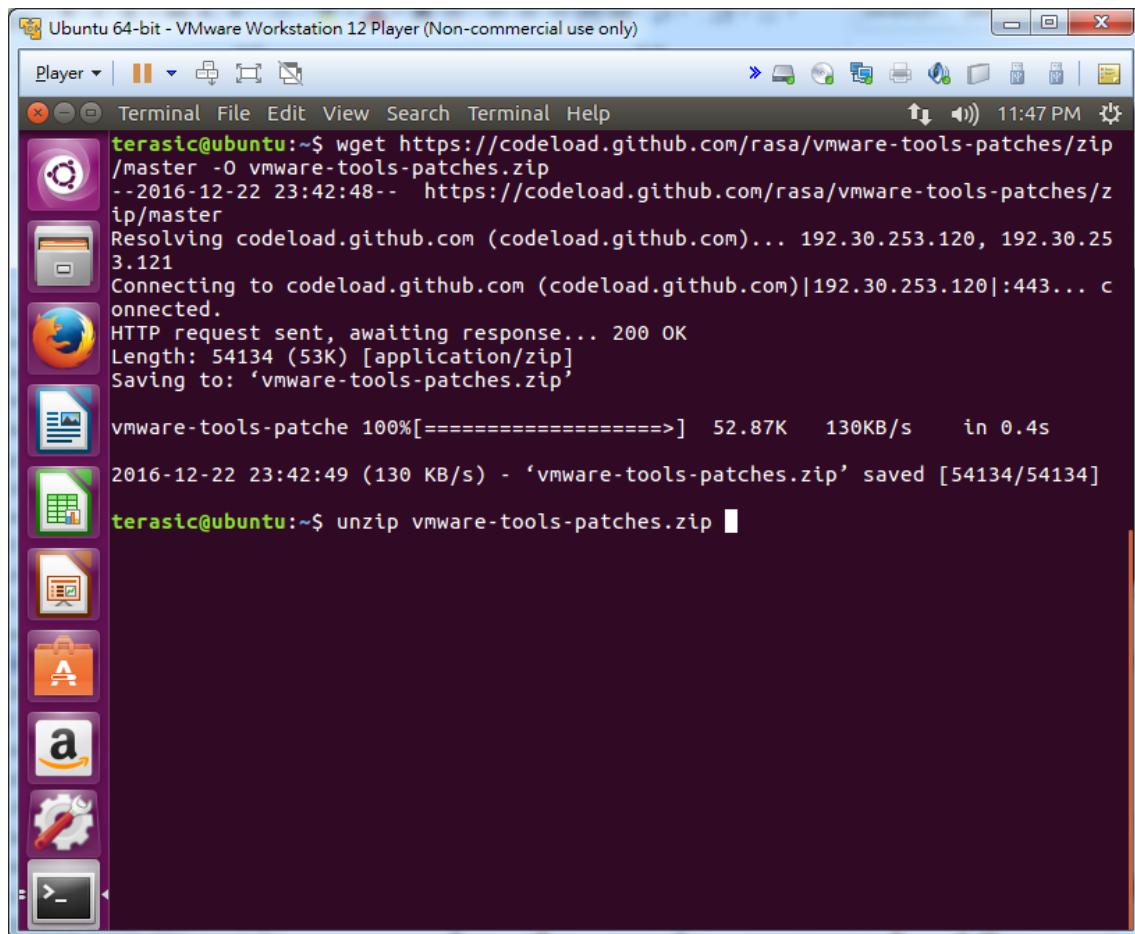
Figure 2-33 shows the screenshot of patches download finish after typing in the wget command.

```
terasic@ubuntu:~$ wget https://codeload.github.com/rasa/vmware-tools-patches/zip /master -O vmware-tools-patches.zip
--2016-12-22 23:42:48-- https://codeload.github.com/rasa/vmware-tools-patches/zip/master
Resolving codeload.github.com (codeload.github.com)... 192.30.253.120, 192.30.253.121
Connecting to codeload.github.com (codeload.github.com)|192.30.253.120|:443...
HTTP request sent, awaiting response... 200 OK
Length: 54134 (53K) [application/zip]
Saving to: 'vmware-tools-patches.zip'

vmware-tools-patches.zip 100%[=====] 52.87K 130KB/s in 0.4s
2016-12-22 23:42:49 (130 KB/s) - 'vmware-tools-patches.zip' saved [54134/54134]
terasic@ubuntu:~$
```

Figure 2-33 VMware Tools Patches Download Finished

The downloaded file is in the compressed format, so we need to decompress it before we can use it. **Figure 4-2** shows the screenshot after patches download has been finished and has been extracted by the **unzip** command. After extraction, the patches is located in the folder " ~/vmware-tools-patches-master"

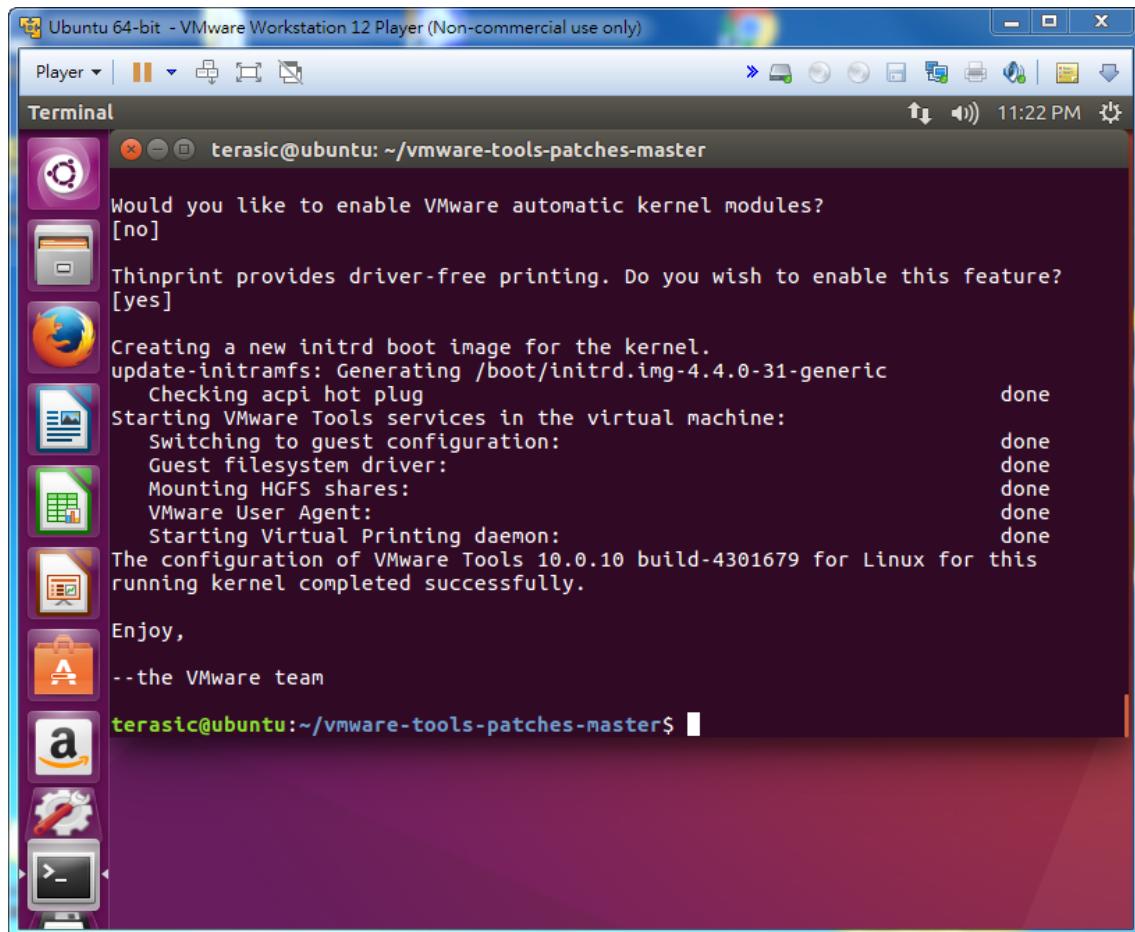


```
terasic@ubuntu:~$ wget https://codeload.github.com/rasa/vmware-tools-patches/zip/master -O vmware-tools-patches.zip
--2016-12-22 23:42:48-- https://codeload.github.com/rasa/vmware-tools-patches/zip/master
Resolving codeload.github.com (codeload.github.com)... 192.30.253.120, 192.30.253.121
Connecting to codeload.github.com (codeload.github.com)|192.30.253.120|:443...
HTTP request sent, awaiting response... 200 OK
Length: 54134 (53K) [application/zip]
Saving to: 'vmware-tools-patches.zip'

vmware-tools-patches 100%[=====] 52.87K 130KB/s in 0.4s
2016-12-22 23:42:49 (130 KB/s) - 'vmware-tools-patches.zip' saved [54134/54134]
terasic@ubuntu:~$ unzip vmware-tools-patches.zip
```

**Figure 2-34 VMware Tools Patches Extracted by the **unzip** Command**

In order to fix VMware Tools bug, installing patches is required. **Figure 2-35** shows the screenshot of installing the patches. The system will prompt users to input a password. Please input your password (in this tutorial, the password is "123") and press ENTER.



```
terasic@ubuntu: ~/vmware-tools-patches-master
Would you like to enable VMware automatic kernel modules?
[no]

Thinprint provides driver-free printing. Do you wish to enable this feature?
[yes]

Creating a new initrd boot image for the kernel.
update-initramfs: Generating /boot/initrd.img-4.4.0-31-generic
    Checking acpi hot plug                                         done
Starting VMware Tools services in the virtual machine:
    Switching to guest configuration:                                done
    Guest filesystem driver:                                         done
    Mounting HGFS shares:                                           done
    VMware User Agent:                                              done
    Starting Virtual Printing daemon:                               done
The configuration of VMware Tools 10.0.10 build-4301679 for Linux for this
running kernel completed successfully.

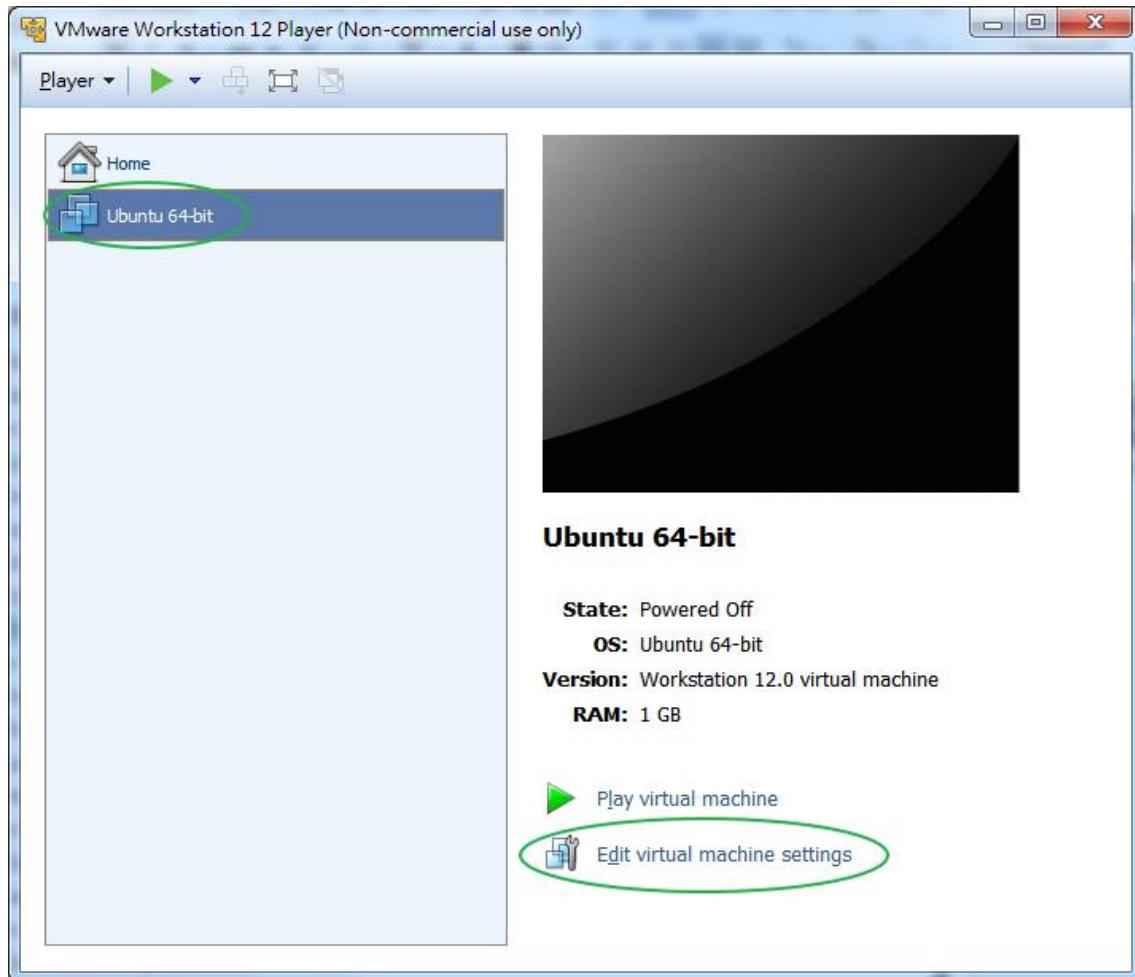
Enjoy,
--the VMware team

terasic@ubuntu:~/vmware-tools-patches-master$
```

**Figure 2-35 VMware Tools Patches Installation**

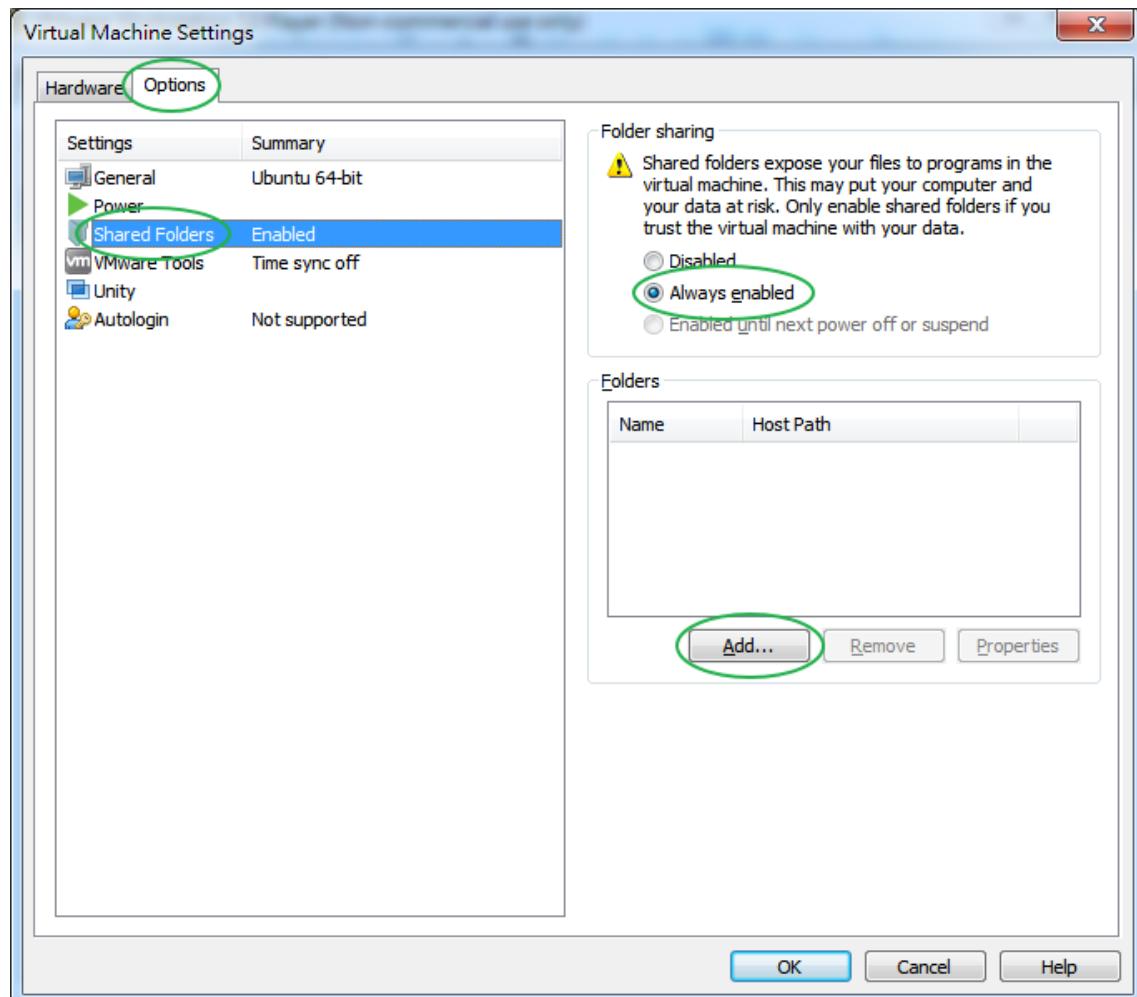
## ■ Shared Folder Setup

In order to modify virtual machine setting. First, shut down Linux system. Launch the VMware Workstation Player, select the "Ubuntu 64-bit", and click "Edit virtual machine settings", as shown in **Figure 2-36**.



**Figure 2-36 Launch Edit Virtual Machine Settings**

When the **Virtual Machine Settings** dialog appears as shown in **Figure 2-37**, please select "Options" tab, click "Shared Folders" icons, check "Always enabled" radio button, and then click "Add...".



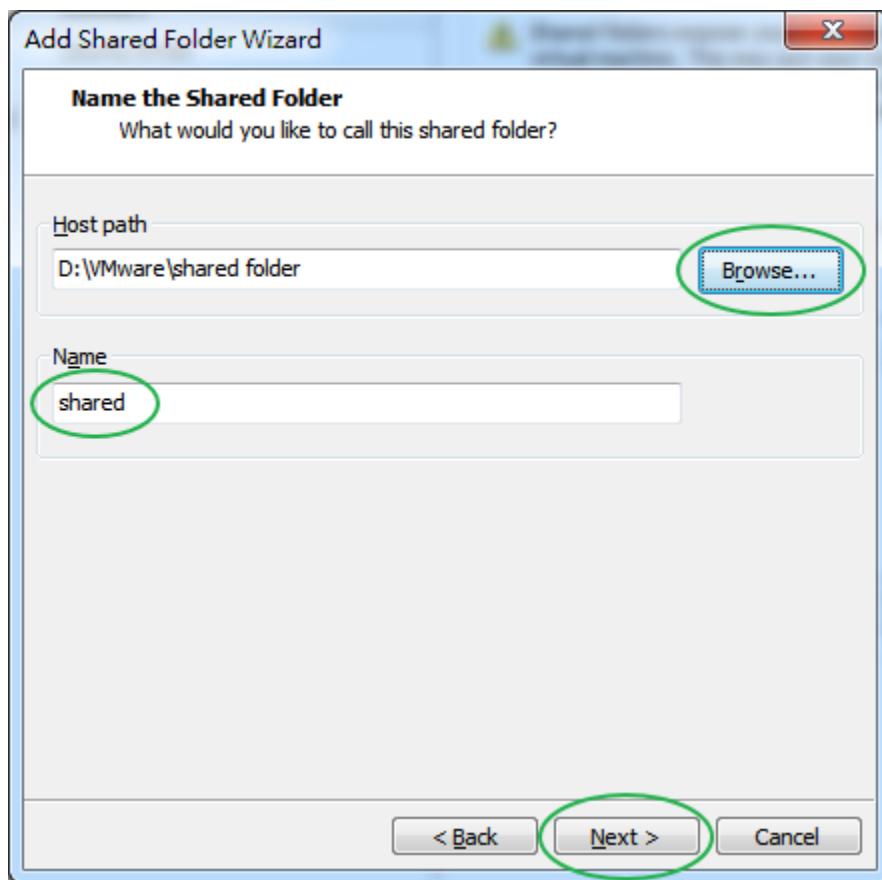
**Figure 2-37 Add Shared Folders**

**Add Shared Folder Wizard** is launched for shared folder settings. A Welcome dialog appears as shown in **Figure 2-38**, please click "Next >" to go to the next step.



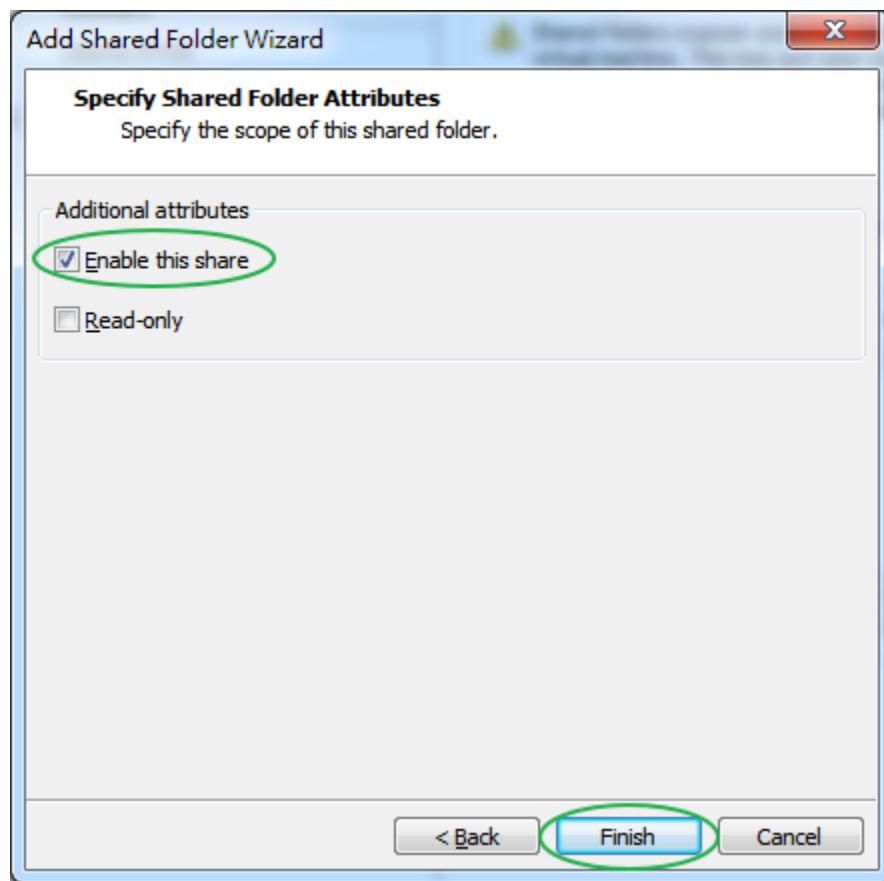
**Figure 2-38 Welcome Dialog of Add Shared Folder Wizard**

When the **Name the Shared Folder** dialog appears as shown in **Figure 2-39**, click "Browse..." to select a folder as the shared folder, gives a name to the shared folder(in this tutorial, **shared** is used ), and click "Next >" to proceed.



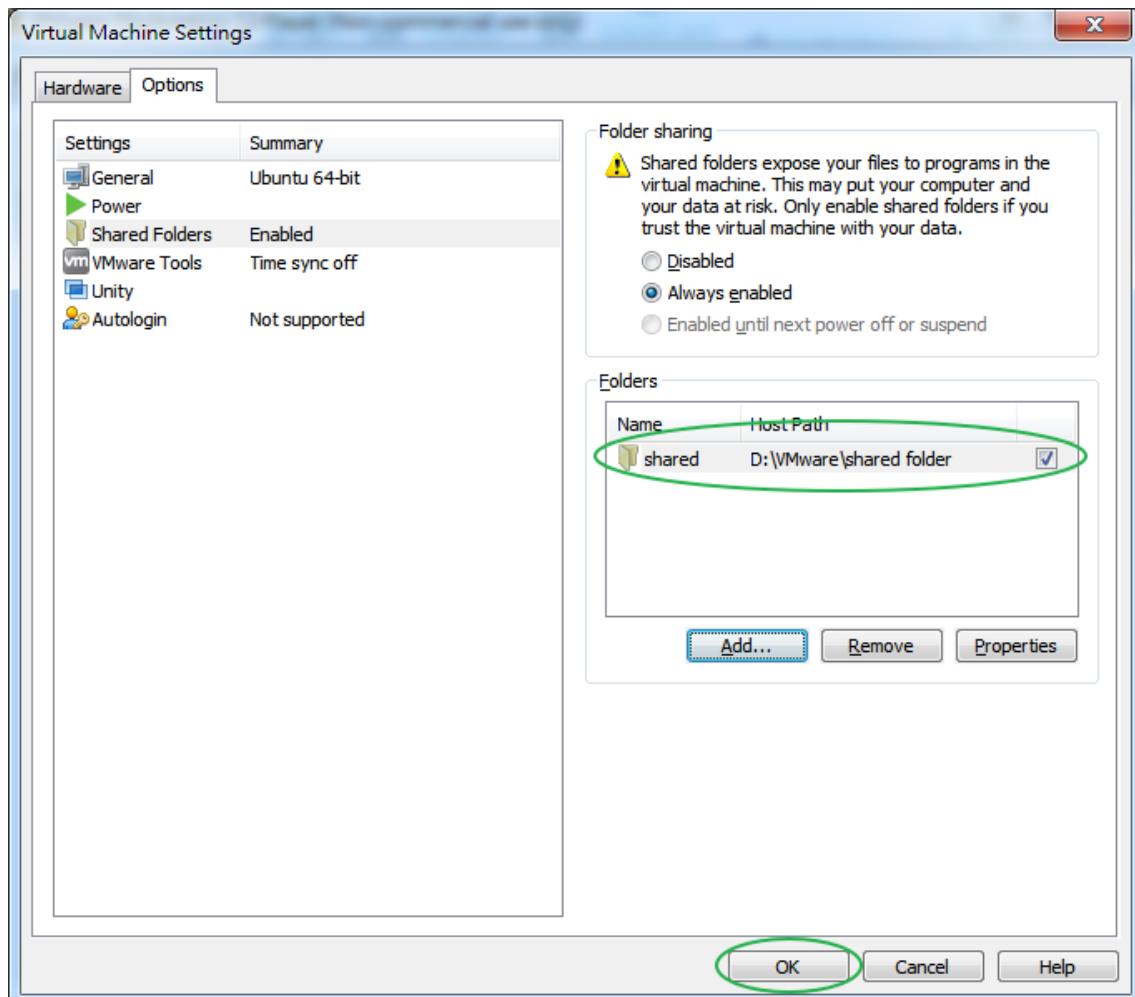
**Figure 2-39 Specify Shared Folder Location and Name**

When the **Specify Shared Folder Attributes** dialog appears as shown in **Figure 2-40**, check "Enable this share" checkbox and click "Finish".



**Figure 2-40 Finish Shared Folder Setup**

When going back to the **Virtual Machine Settings** dialog as shown in **Figure 2-41**, check "OK" to complete the settings. Now the Shared Folder feature is applied.



**Figure 2-41 Complete Virtual Machine Setting**

# Chapter 3

## *Qt Creator Installation*

The Control Panel is a GUI-based program and Qt Creator is a convenient tool to help create GUI-based programs. Users can simply follow our GUI application development explained in this tutorial to develop needed GUI applications.

Chapter 3 illustrates how to install Qt Creator on the Linux x64 which has been installed in previous chapters. Also, we will show how to create, compile, and build a hello program running on the Linux x64. In order for Qt Creator to be able to build a project correctly, we also provide steps of installing necessary x86\_64 GCC toolchain needed for the Qt Creator.

### **3.1 Install Toolchain for Linux x64**

In order for the Qt Creator to be able to build a project correctly, a proper toolchain is required. Use the following command to install the required toolchain:

```
$ sudo apt install build-essential libgl1-mesa-dev -y
```

System will prompt the user to input a password (in this tutorial, the password is "123") as shown in **Figure 3-1**.

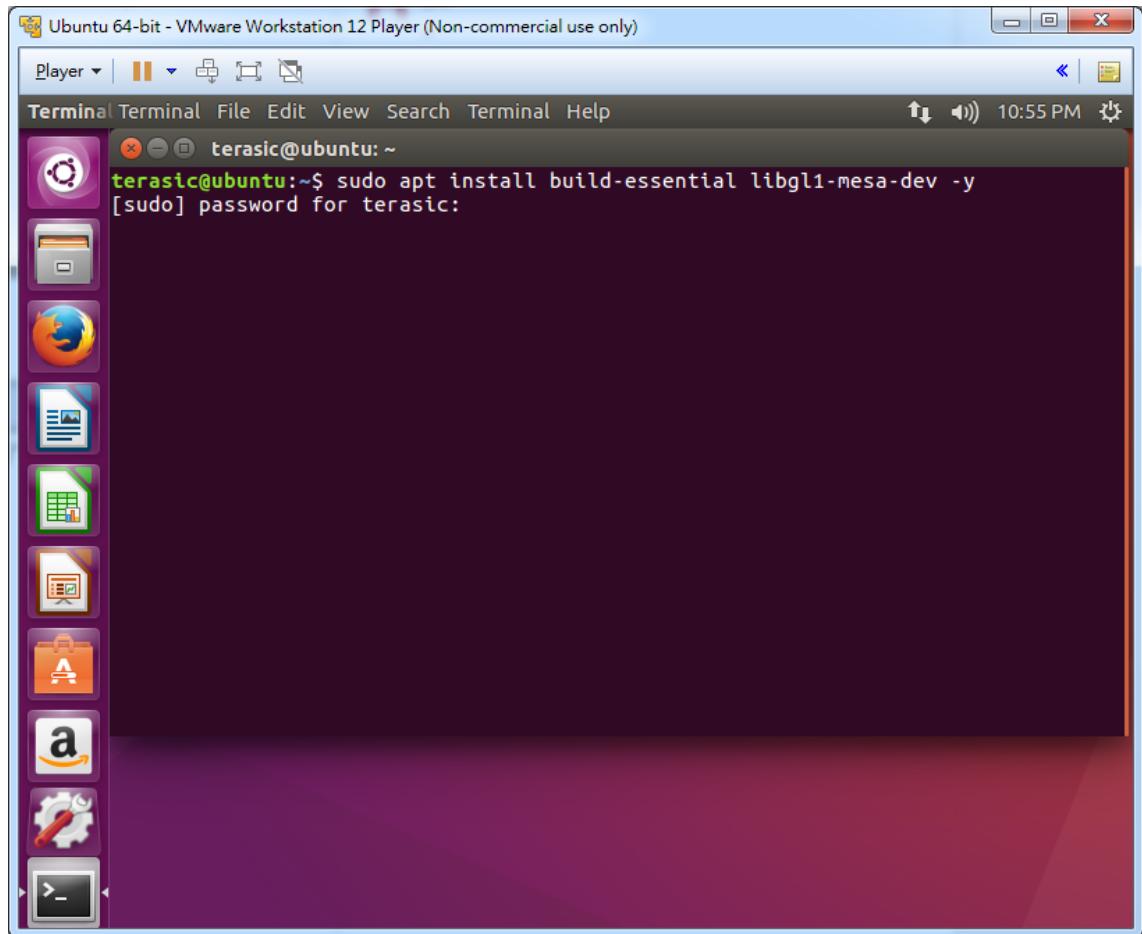


Figure 3-1 Install Software Package

**Figure 3-2** shows the screenshot after the installation has been completed. Type "exit" to close the terminal.

```
Setting up libxcb-dri3-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-glx0-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-render0-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-randr0-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-shape0-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-xfixes0-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-sync-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up libxcb-present-dev:amd64 (1.11.1-1ubuntu1) ...
Setting up x11proto-xext-dev (7.3.0-1) ...
Setting up x11proto-fixes-dev (1:5.0-2ubuntu2) ...
Setting up libxfixes-dev:amd64 (1:5.0.1-2) ...
Setting up x11proto-damage-dev (1:1.2.1-2) ...
Setting up libxdamage-dev:amd64 (1:1.1.4-2) ...
Setting up libxext-dev:amd64 (2:1.3.3-1) ...
Setting up libxshmfence-dev:amd64 (1.2-1) ...
Setting up x11proto-xf86vidmode-dev (2.3.1-2) ...
Setting up libxf86vm-dev:amd64 (1:1.1.4-1) ...
Setting up mesa-common-dev:amd64 (11.2.0-1ubuntu2.2) ...
Setting up x11proto-dri2-dev (2.8-2) ...
Setting up x11proto-gl-dev (1.4.17-1) ...
Setting up libgl1-mesa-dev:amd64 (11.2.0-1ubuntu2.2) ...
terasic@ubuntu:~$
```

**Figure 3-2 Software Package Installation is Now Complete**

## 3.2 Download and Install Qt Installer

### ■ Download Qt Installer

As shown in **Figure 3-3**, click the Firefox web browser to open the web page

<http://download.qt.io/archive/qt/5.7/5.7.0/>

Then click "qt-opensource-linux-x64-5.7.0.run" to download the Qt installer.

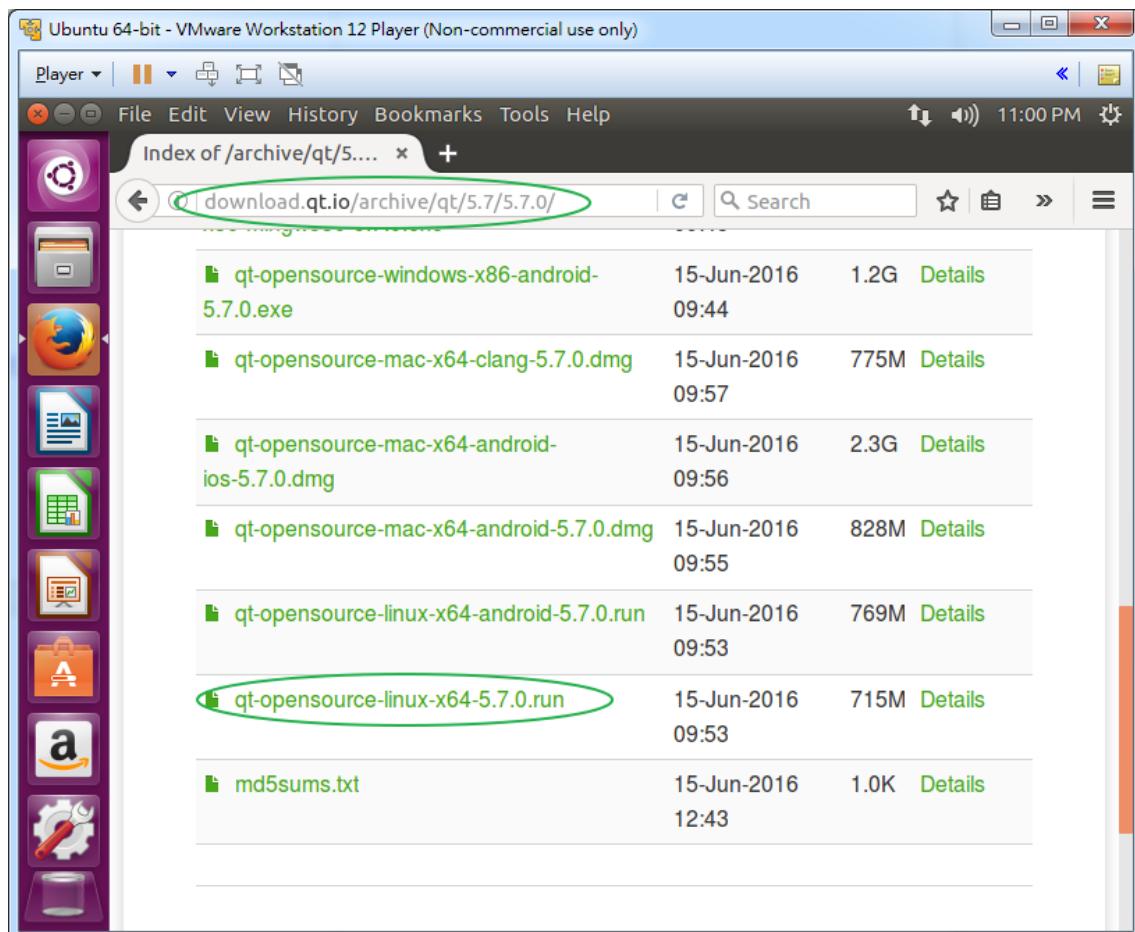
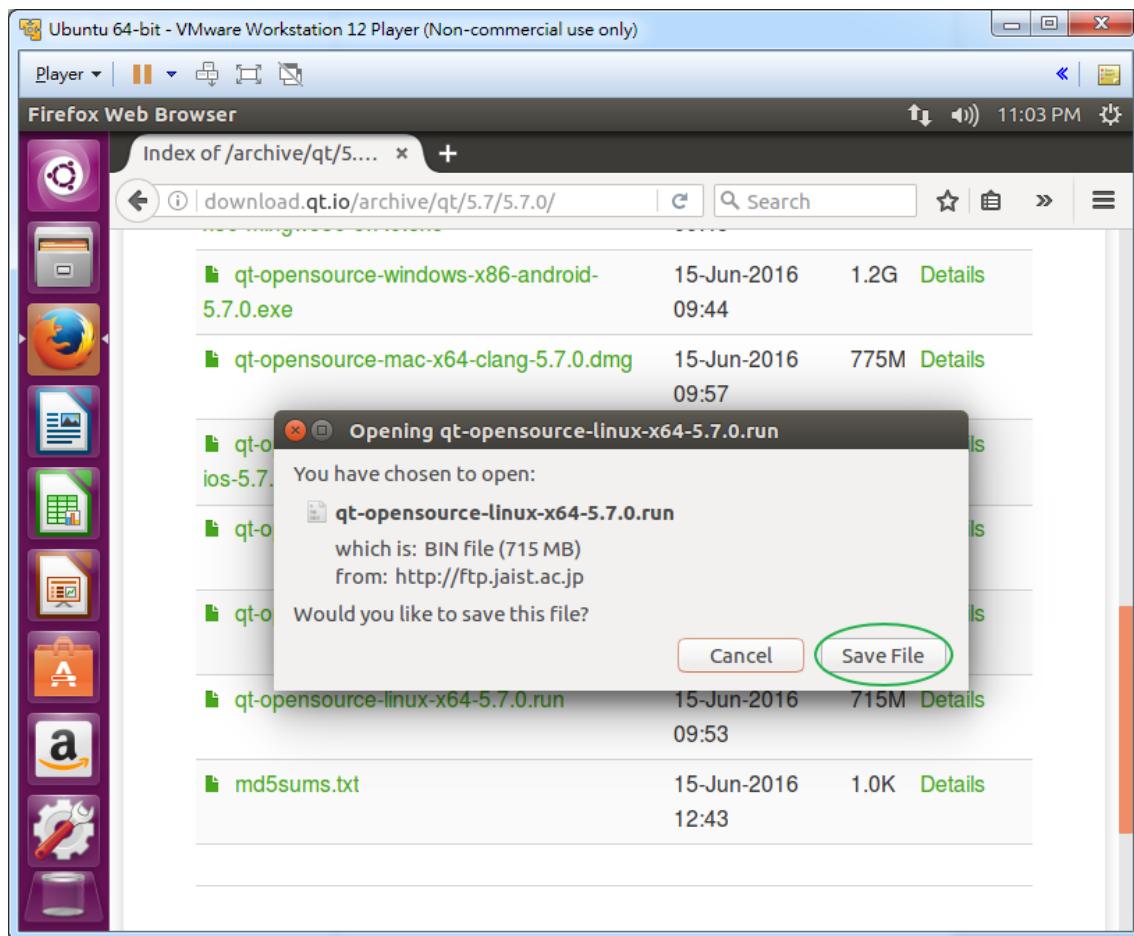


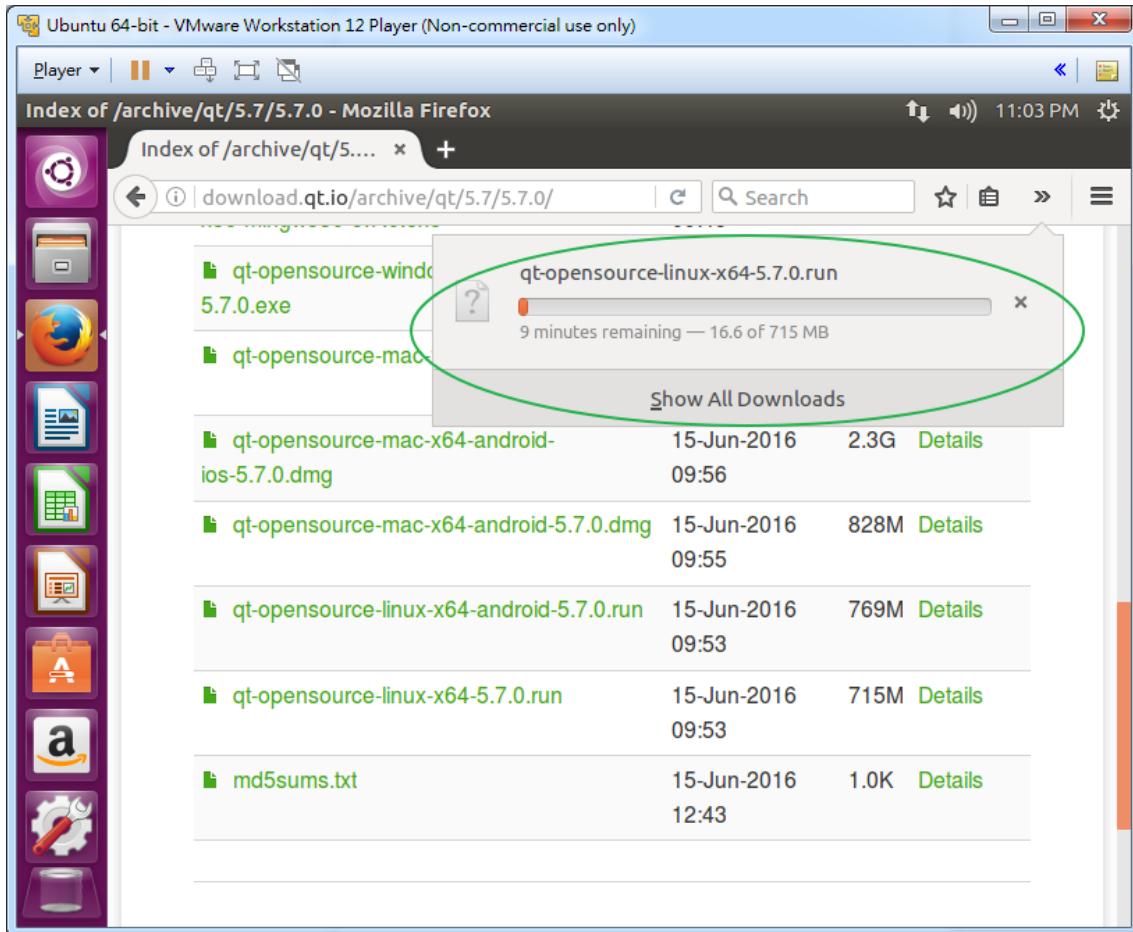
Figure 3-3 Web Page to Download Qt

When an Opening dialog appears as shown in **Figure 3-4**, click "Save File".



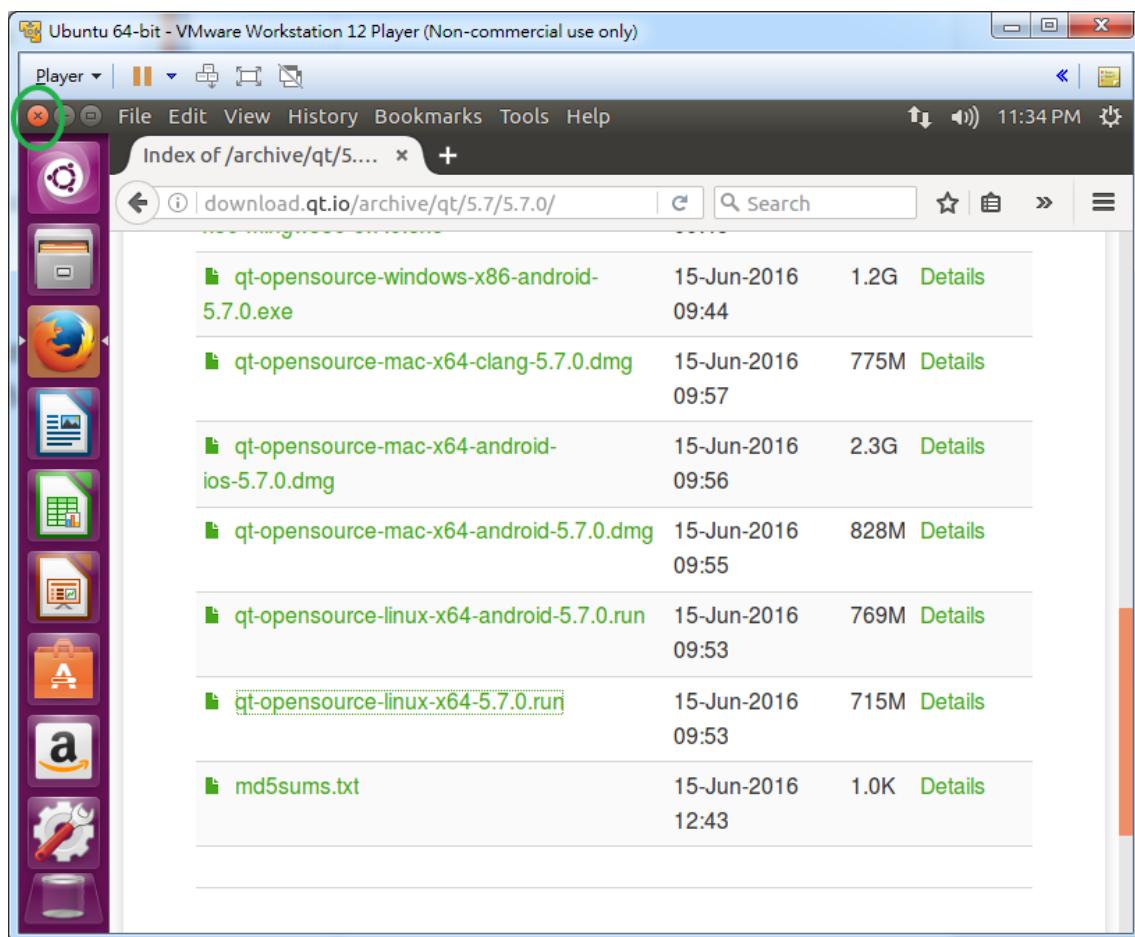
**Figure 3-4 Opening Dialog**

Figure 3-5 shows the download progress.



**Figure 3-5 Download Process of qt-opensource-linux-x64-5.7.0.run**

The file is saved as "qt-opensource-linux-x64-5.7.0.run", and saved under the folder "~/Download". When download is completed, click the Close icon, located on the top-left corner as shown in **Figure 3-6**, to close the Firefox web browser.



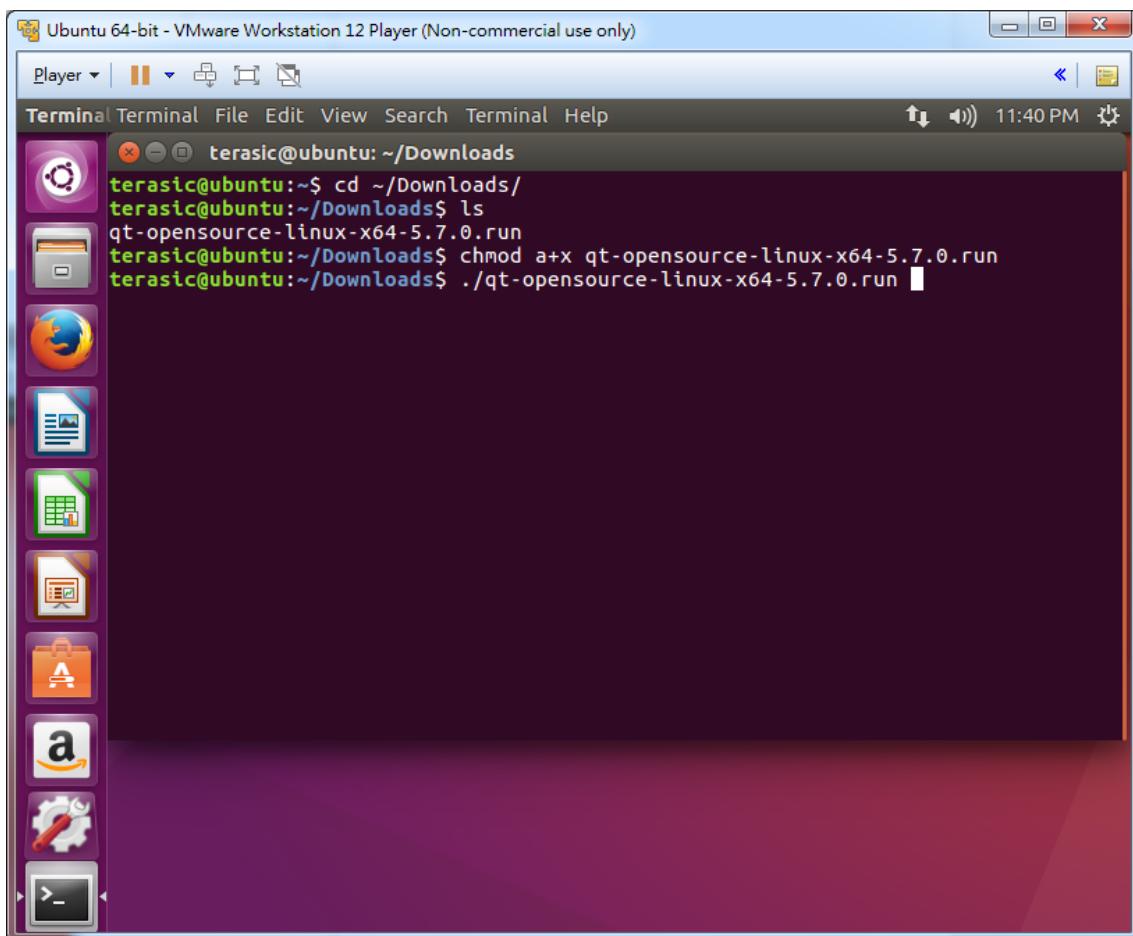
**Figure 3-6 Click 'Close' Icon to Close Firefox Web Browser**

## ■ Install Qt

Type in the following commands to locate and launch the Qt Installer, as shown in **Figure 3-7**.

```
$ cd ~/Downloads/  
$ ls  
$ chmod a+x qt-opensource-linux-x64-5.7.0.run  
$ ./qt-opensource-linux-x64-5.7.0.run
```

The 2<sup>nd</sup> command line **ls** is used to check whether the `qt-opensource-linux-x64-5.7.0.run` existed or not while the 3<sup>rd</sup> command line **chmod a+x** is used to add "execution" attribute to the file such that it can be executed.



**Figure 3-7 Locate and Launch the Qt Installer**

Figure 3-8 shows the Welcome dialog of the Qt Installer.

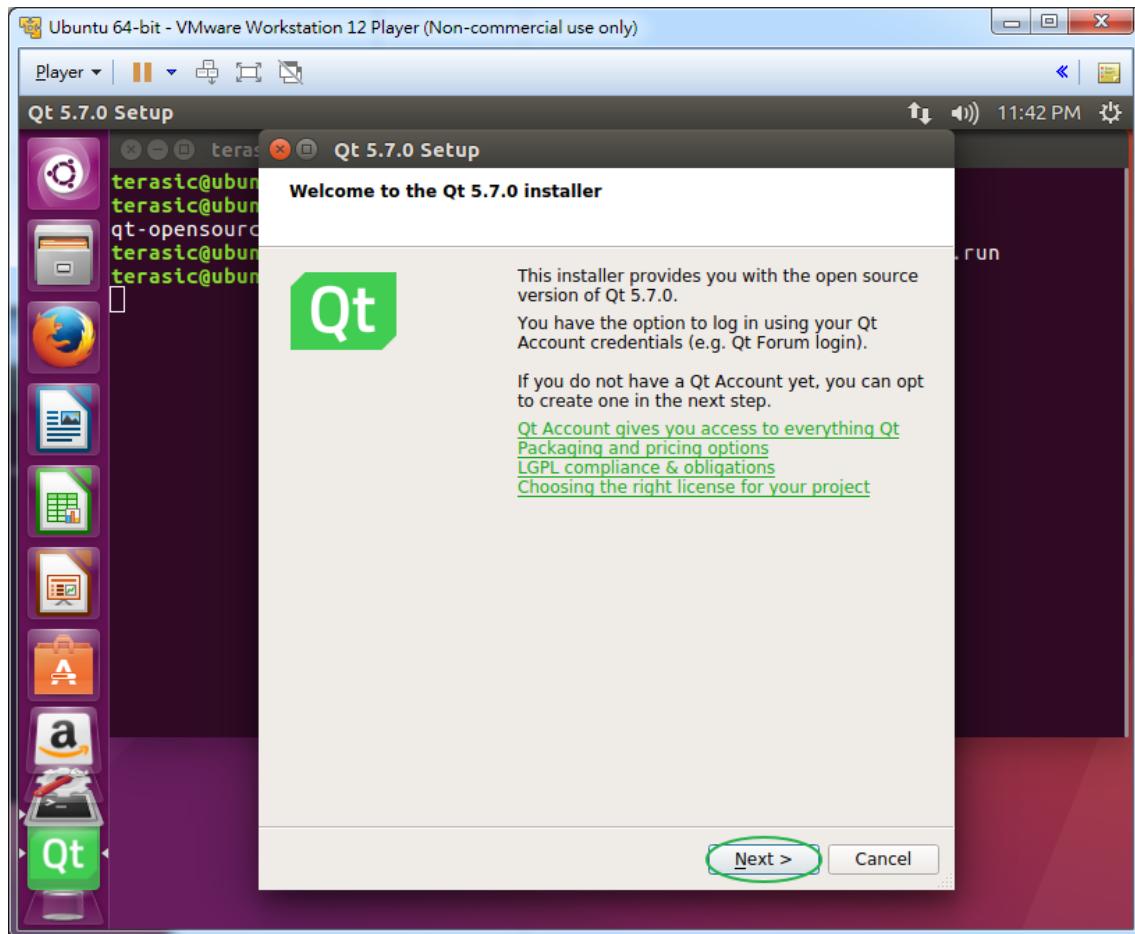
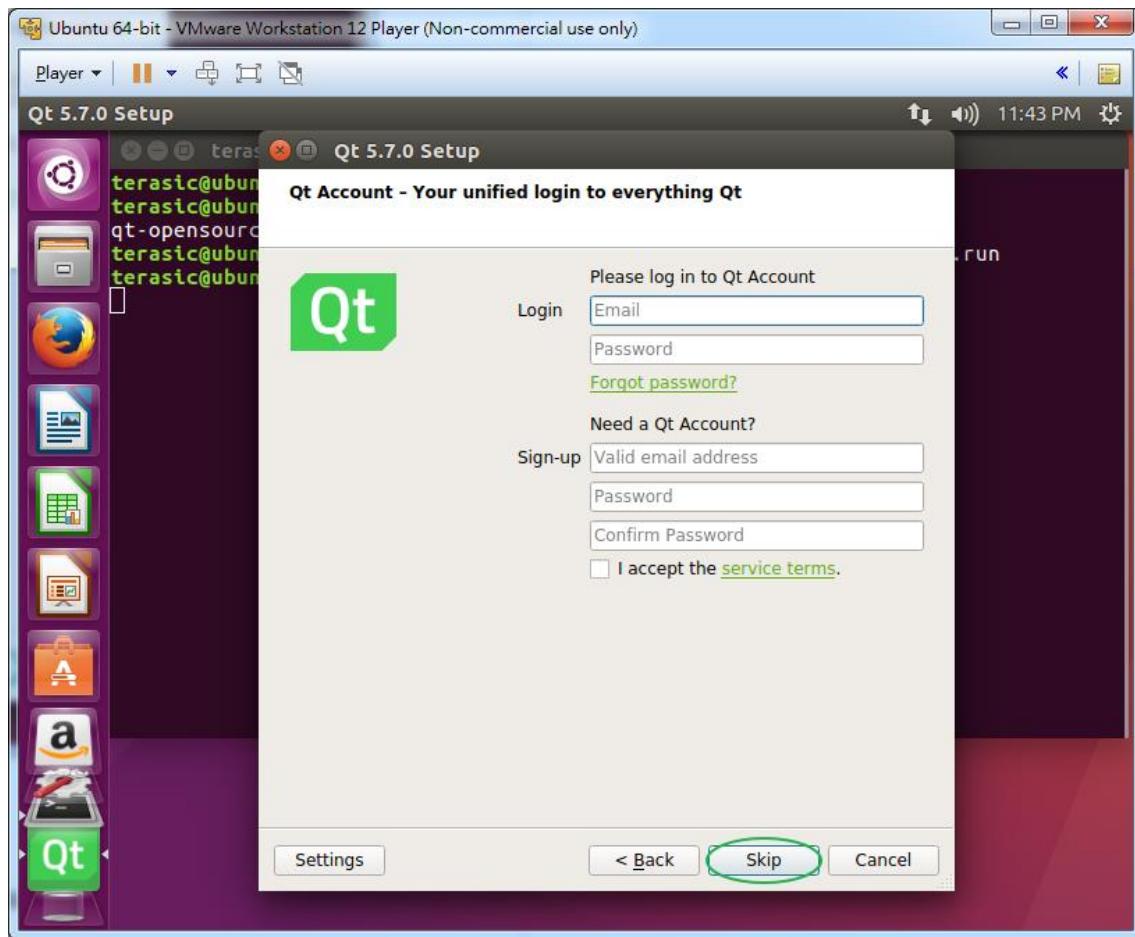


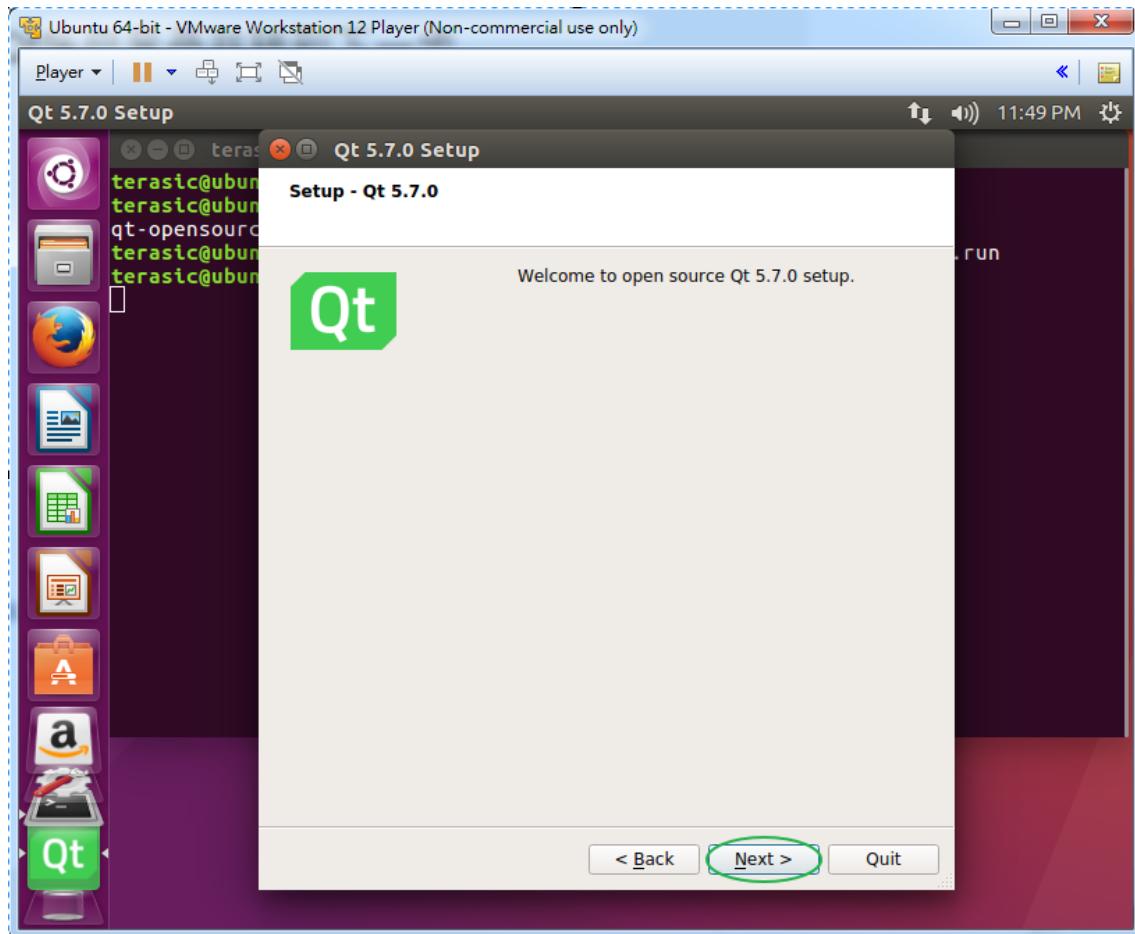
Figure 3-8 Welcome Dialog

In the **Qt Account – Your unified login to everything Qt** dialog, click "Skip" to skip Qt account log in and go to the next step, as shown in **Figure 3-9**.



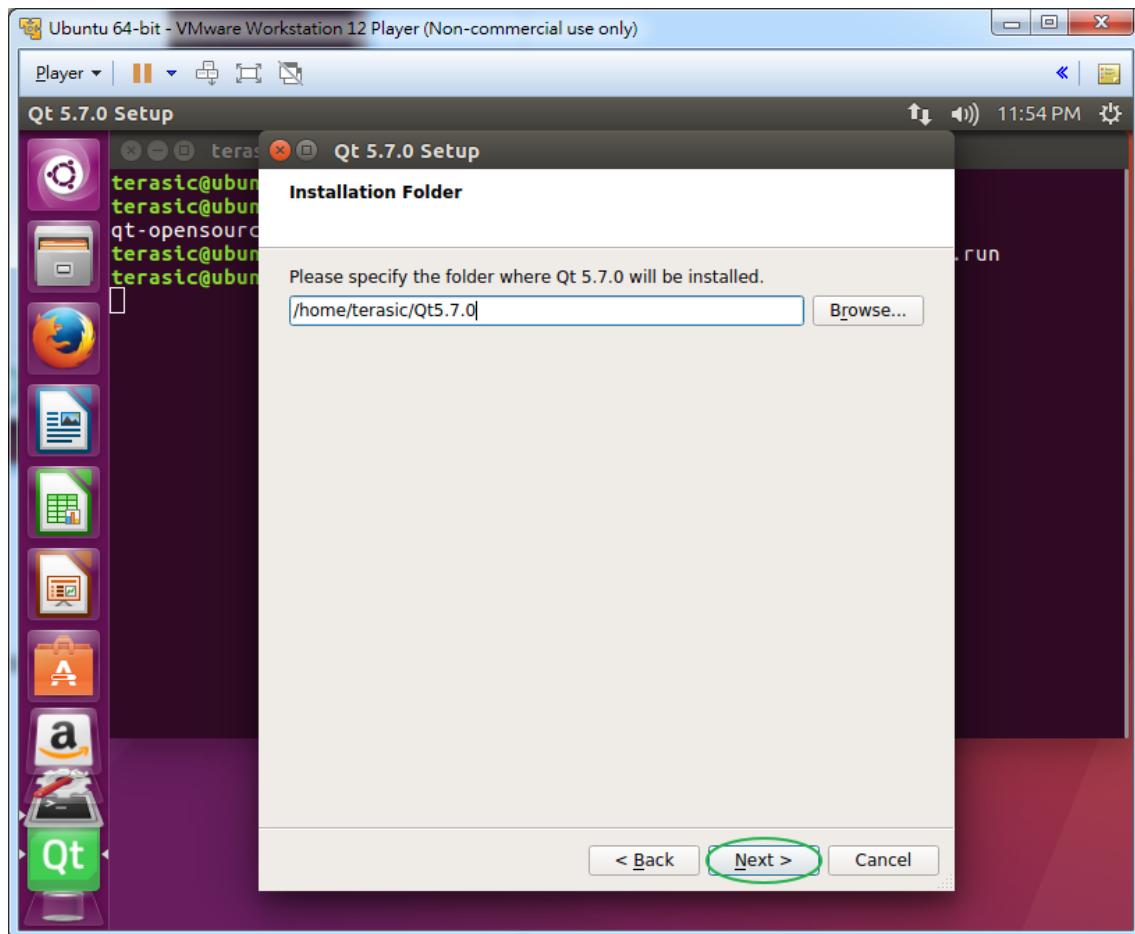
**Figure 3-9 Skip Qt Account log in**

In the **Setup – Qt 5.7.0** dialog, click "Next >" to go to the next step, as shown in **Figure 3-10**.



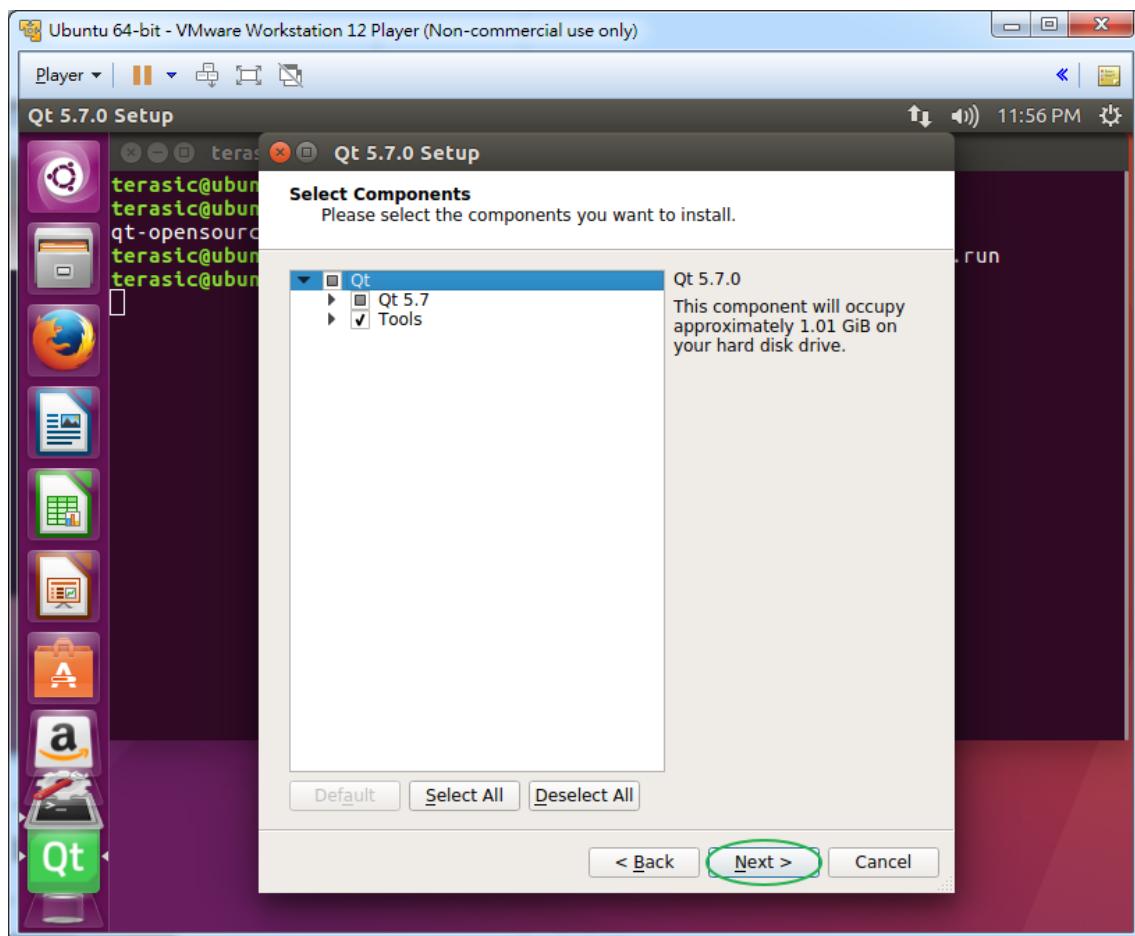
**Figure 3-10 Qt Setup Dialog**

In the **Installation Folder** dialog, please specify the folder where you wish to install Qt 5.7.0 and click "Next >" to go to the next step, as shown in **Figure 3-11**.



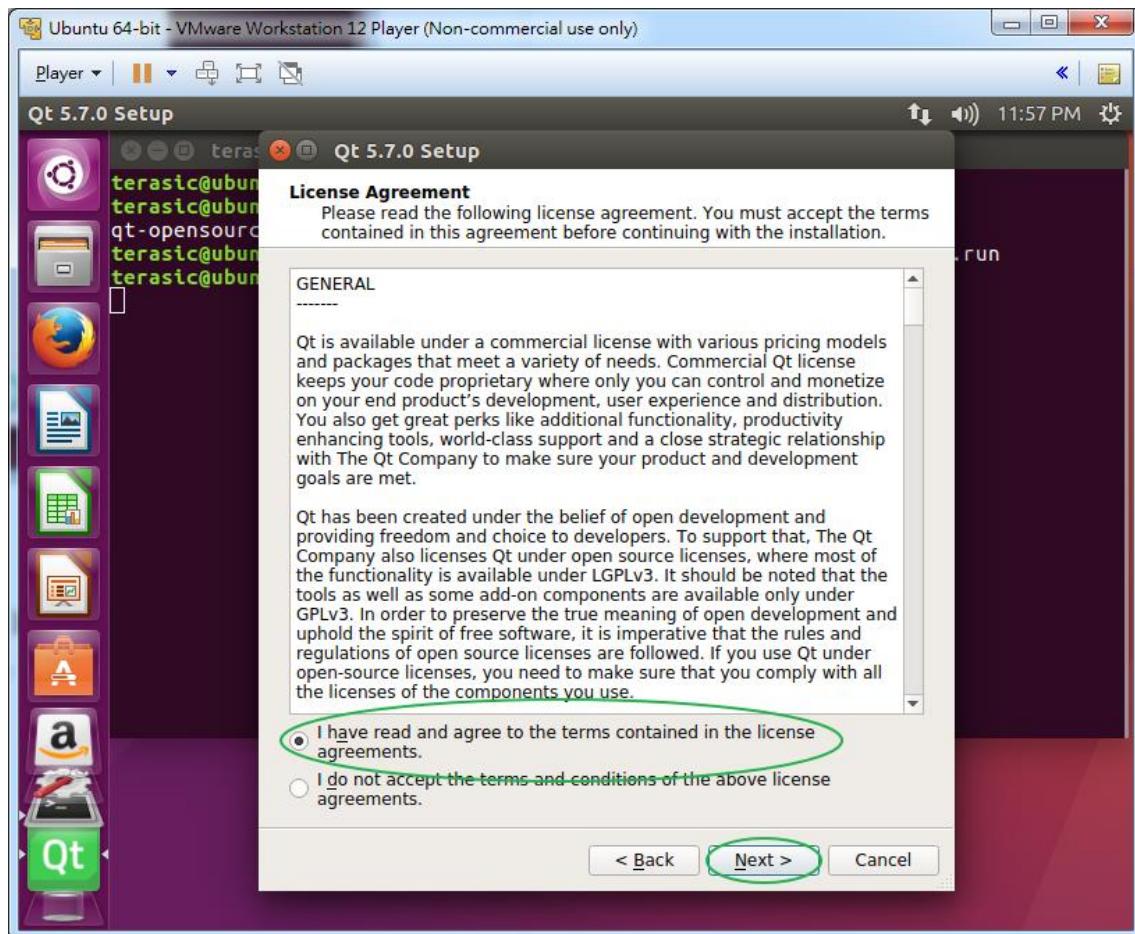
**Figure 3-11 Specify the Folder to Install Qt**

In the **Select Components** dialog, keep default settings and click "Next >" to go to the next step, as shown in **Figure 3-12**



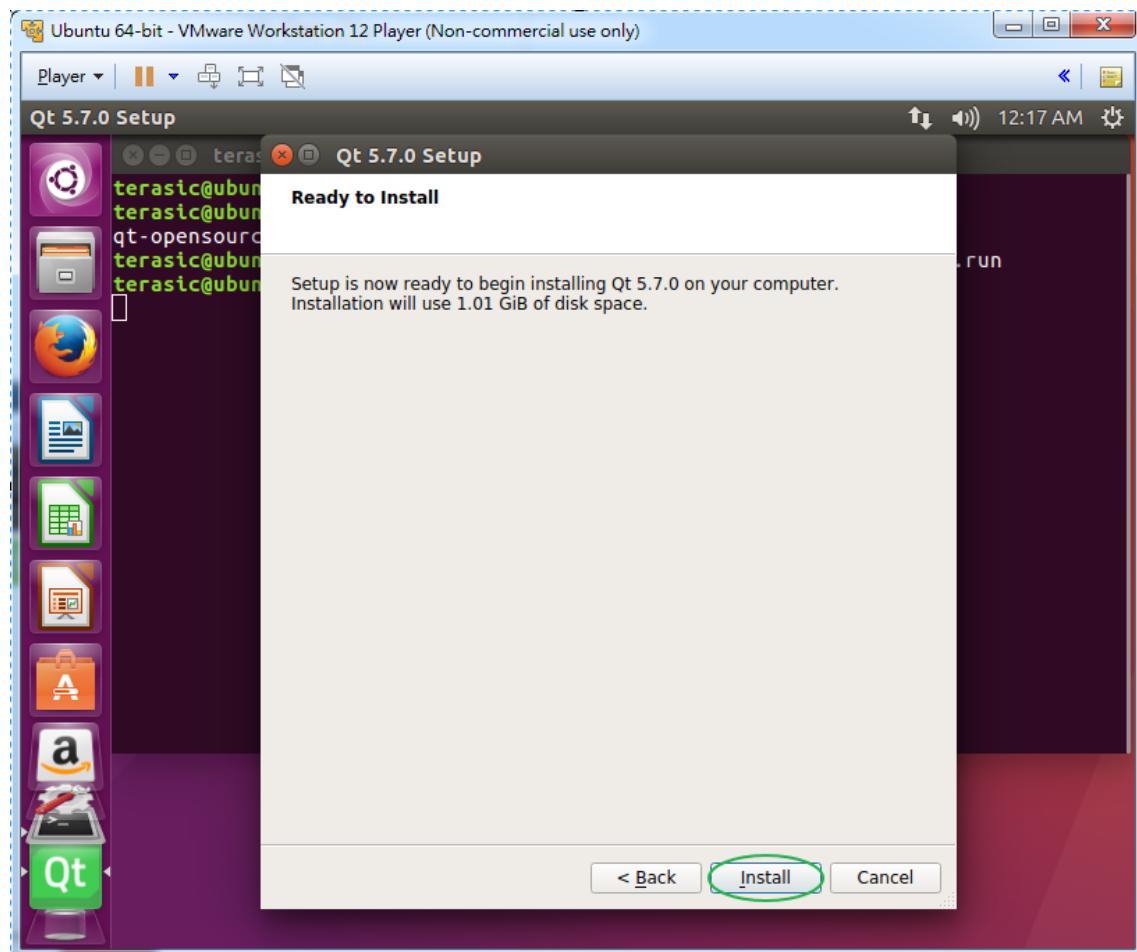
**Figure 3-12 Select Components Dialog**

In the **License Agreement** dialog, select a license and select the "I have read and agree ..." radio button. Click "Next >" to go to the next step, as shown in **Figure 3-13**



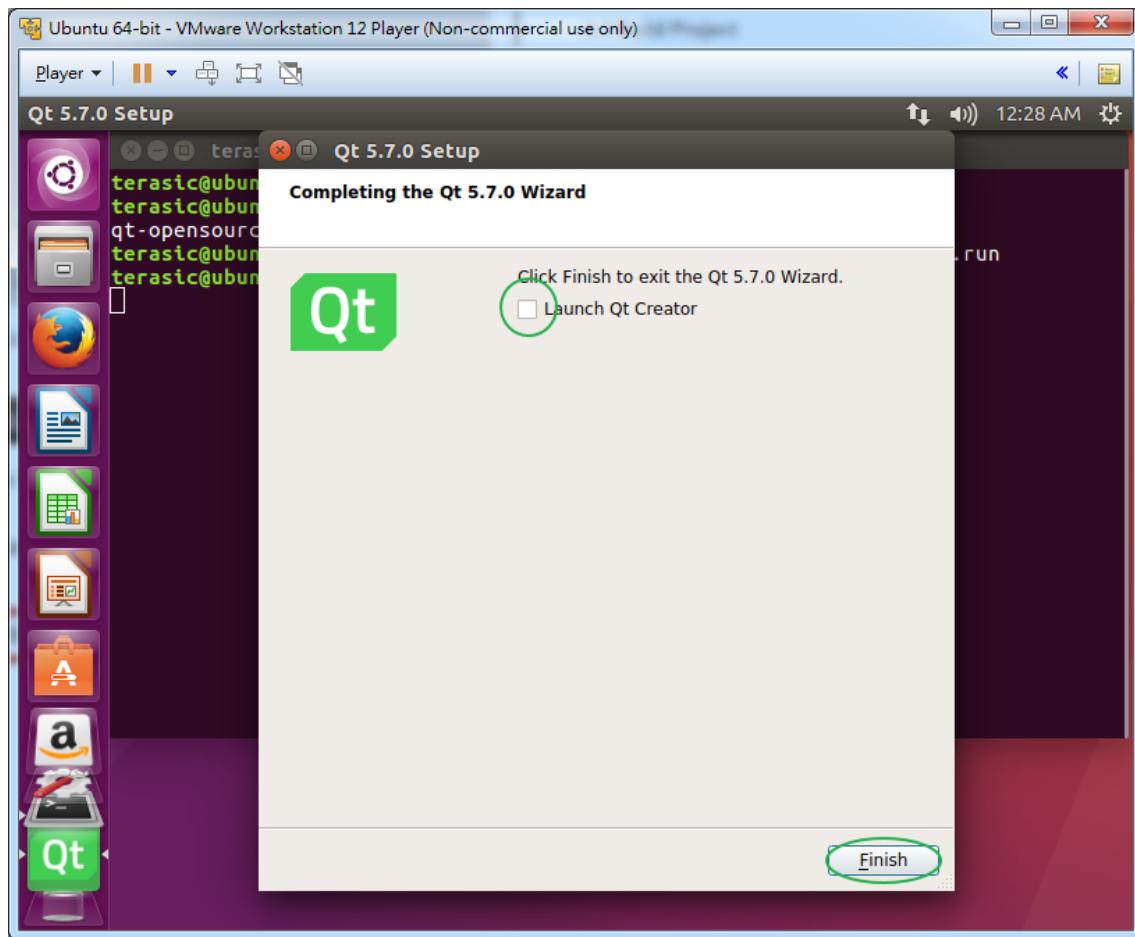
**Figure 3-13 License agreement dialog of the Qt installer**

In the Ready to Install dialog, as shown in **Figure 3-14**, click "Install" to go to the next step.



**Figure 3-14 Ready to Install Dialog of Qt Installer**

In the **Completing the Qt 5.7.0 Wizard** dialog, as shown in [Figure 3-15](#), unselect "Launch Qt Creator" and click "Finish" to close the window.



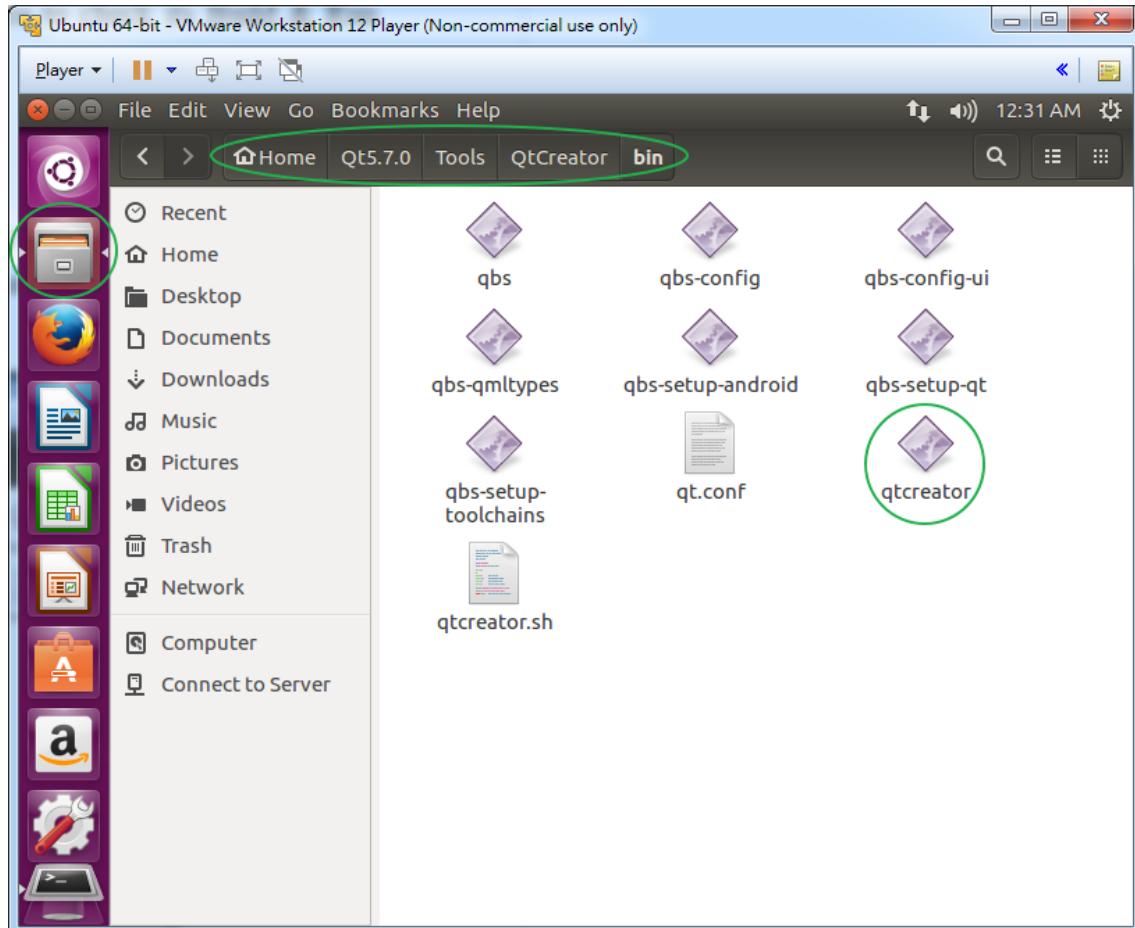
**Figure 3-15 Completing Dialog of Qt Installer**

### 3.3 Launch Qt Creator and Check Configure

This section describes how to launch Qt Creator in Linux and where to check its Build & Run configuration settings.

#### ■ Launch Qt Creator

The Qt Creator program is located under \home\Qt5.7.0\Tools\QtCreator\bin as shown in [Figure 3-16](#). To launch Qt creator, simply double click "qtcreator" program icon.



**Figure 3-16 Launch Qt Creator**

## ■ Check Build & Run Configuration Settings

When Qt Creator is launched, browse the menu and click "Tools→Options..." as shown in Figure 3-17, to open the Option dialog.

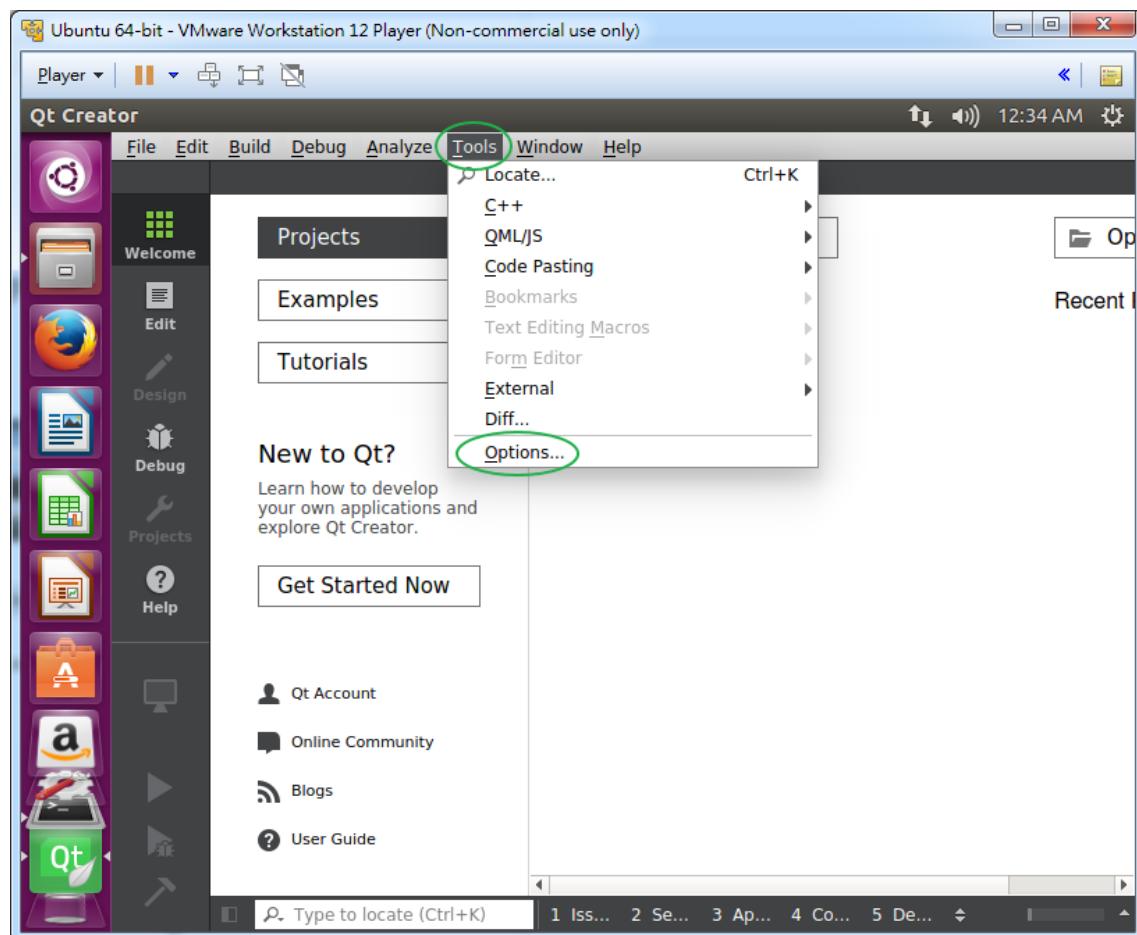
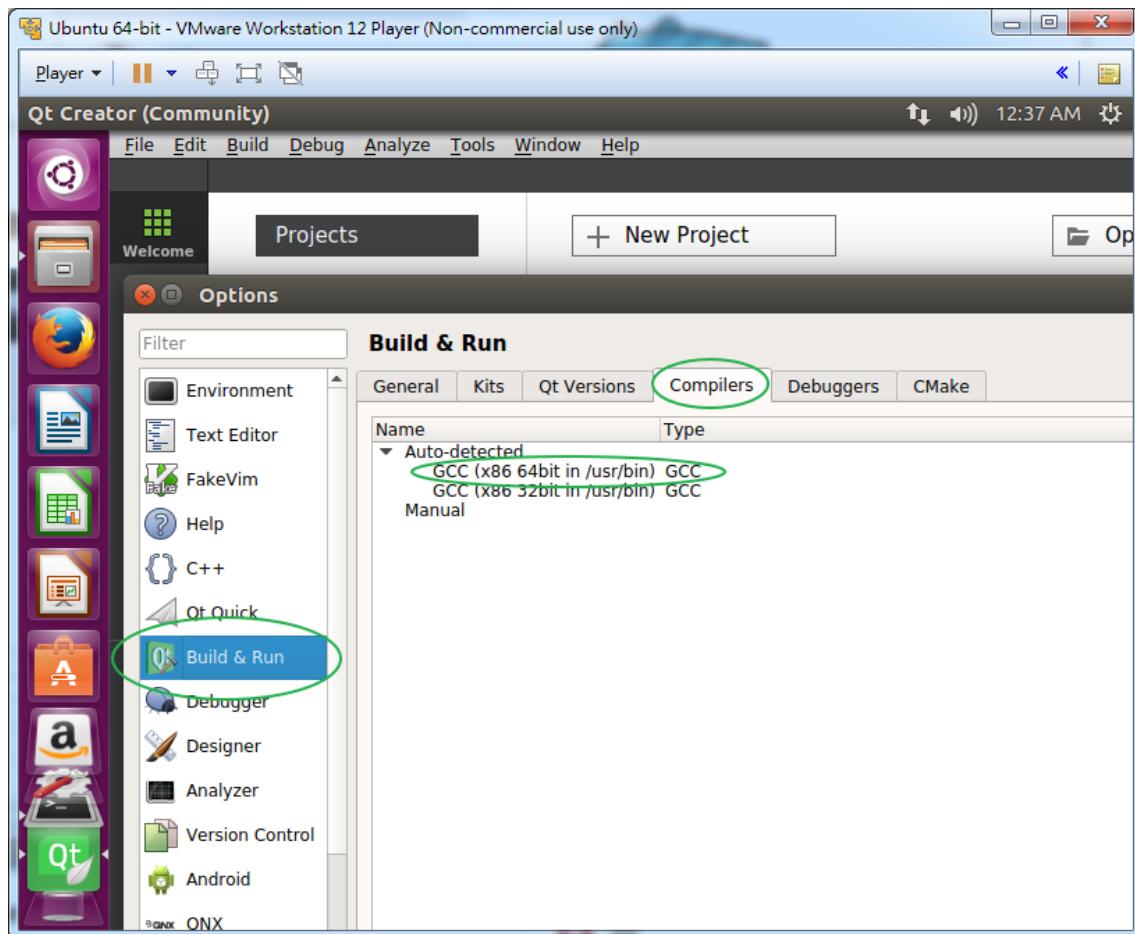


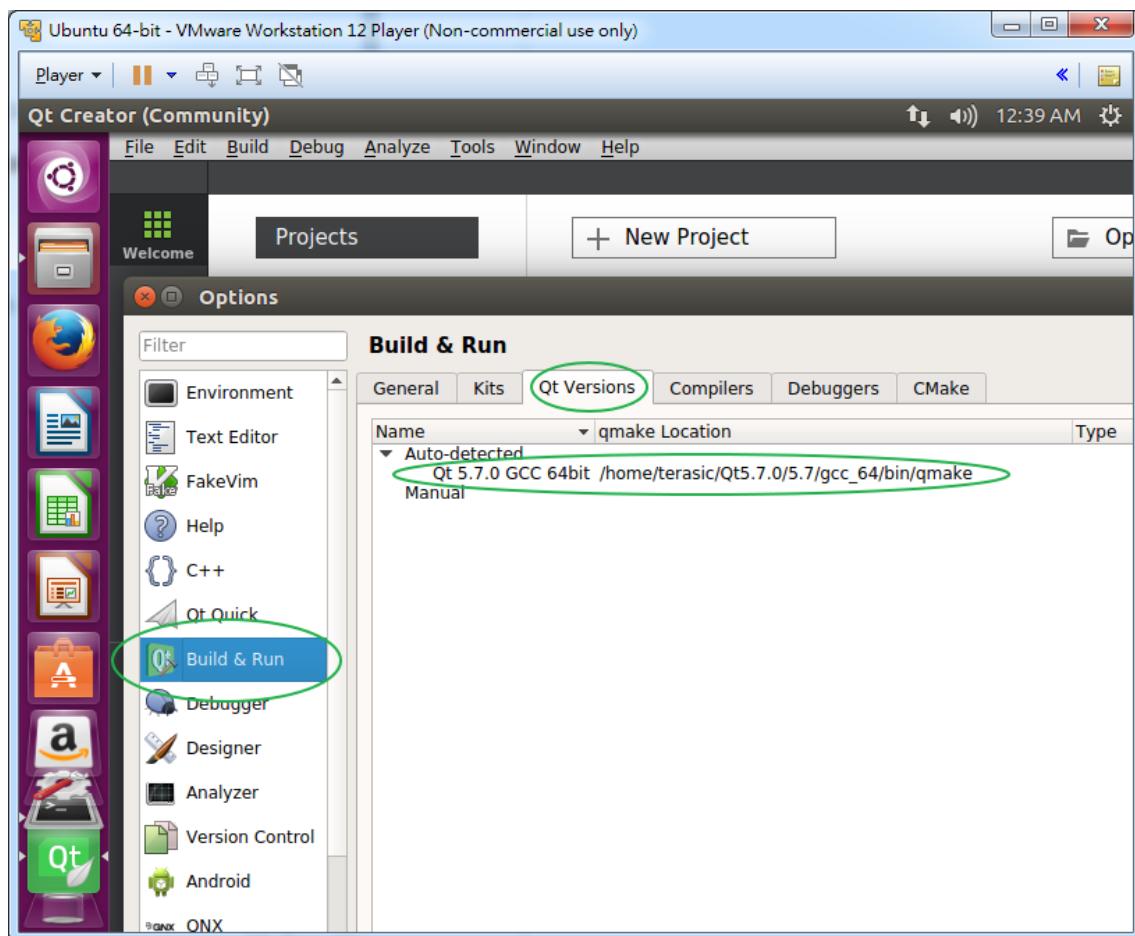
Figure 3-17 Open Option Dialog

In the **Options** dialog, first click "Build & Run" on the left and select the "Compilers" tab on the right to check if the "GCC" is detected as shown in **Figure 3-18**.



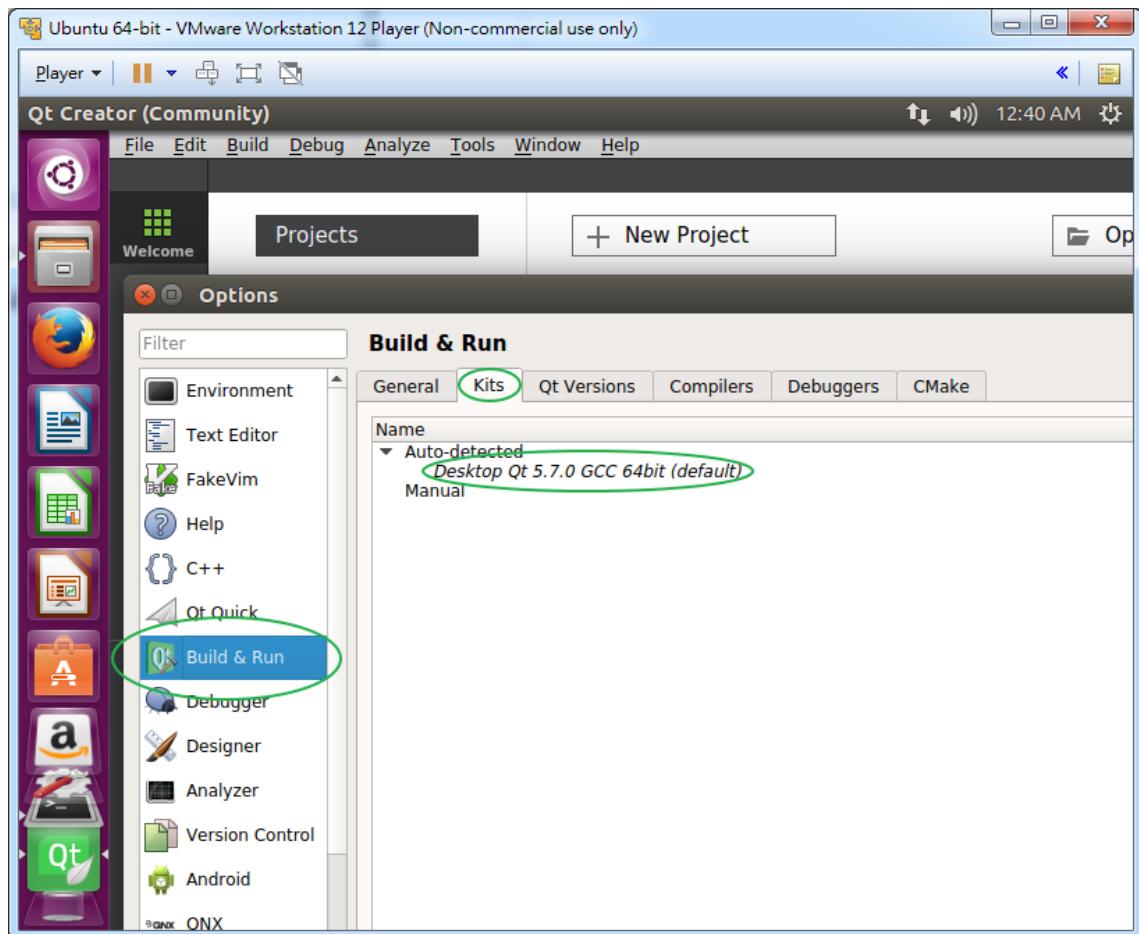
**Figure 3-18 Compilers Options in Qt Creator**

Next, select "Qt Versions" tab (to the left of the "Compilers" tab) to check if the "Qt 5.7.0 GCC 64bit" is detected as shown in **Figure 3-19**.



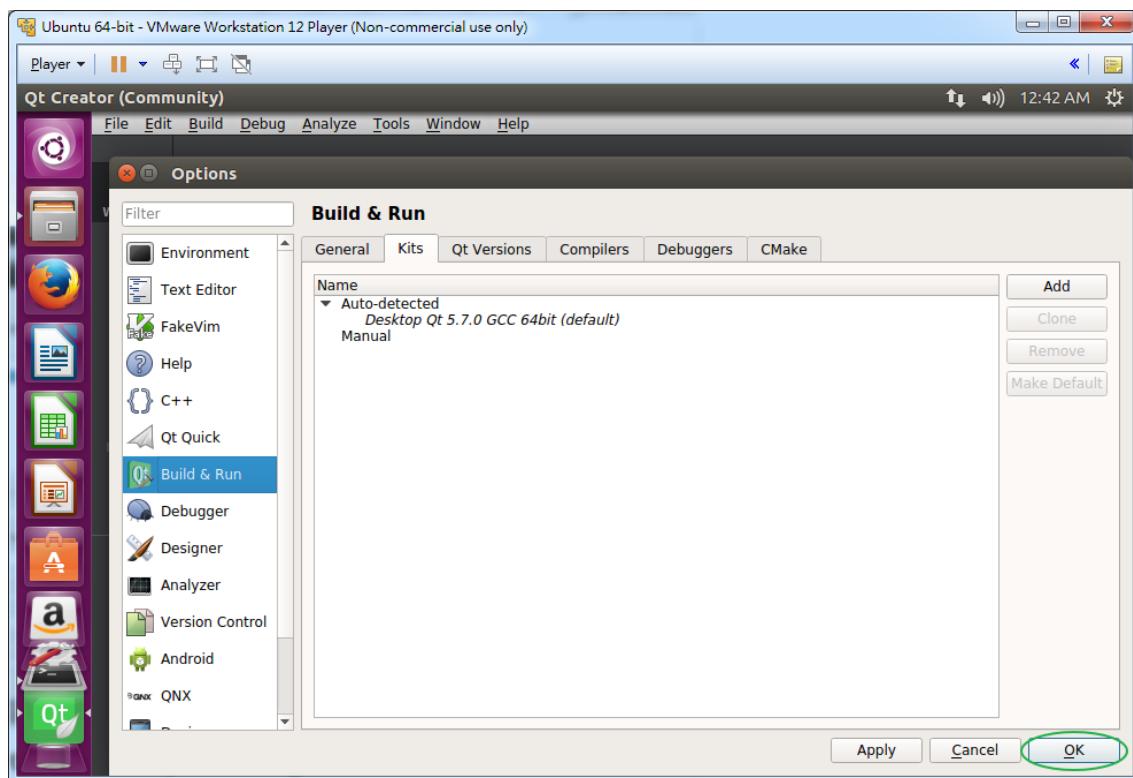
**Figure 3-19 Qt Version Option in Qt Creator**

Similarly, select "Kits" tab to check if the "Desktop Qt 5.7.0 GCC 64bit" is detected as shown in **Figure 3-20**.



**Figure 3-20 Kits Option in Qt Creator**

Finally, click "OK" to close the dialog as shown in **Figure 3-21**.



**Figure 3-21 Click "OK" to Close Options Dialog**

## 3.4 Hello Program

Now we are ready to create, build, and run our first program "Hello" in Qt Creator. Please follow carefully with the following instructions.

### ■ Create a New Project

After launching the Qt Creator, browse the menu and select item "File→New File or Project..." as shown in **Figure 3-22** to open a new dialog.

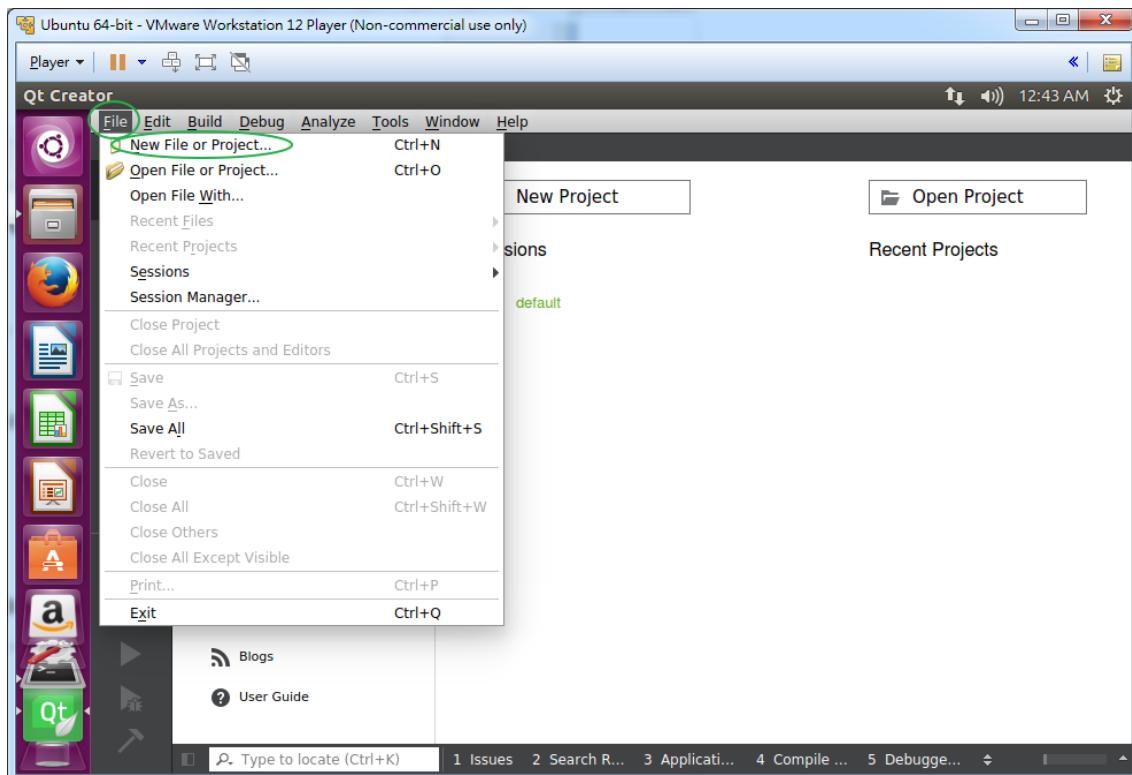
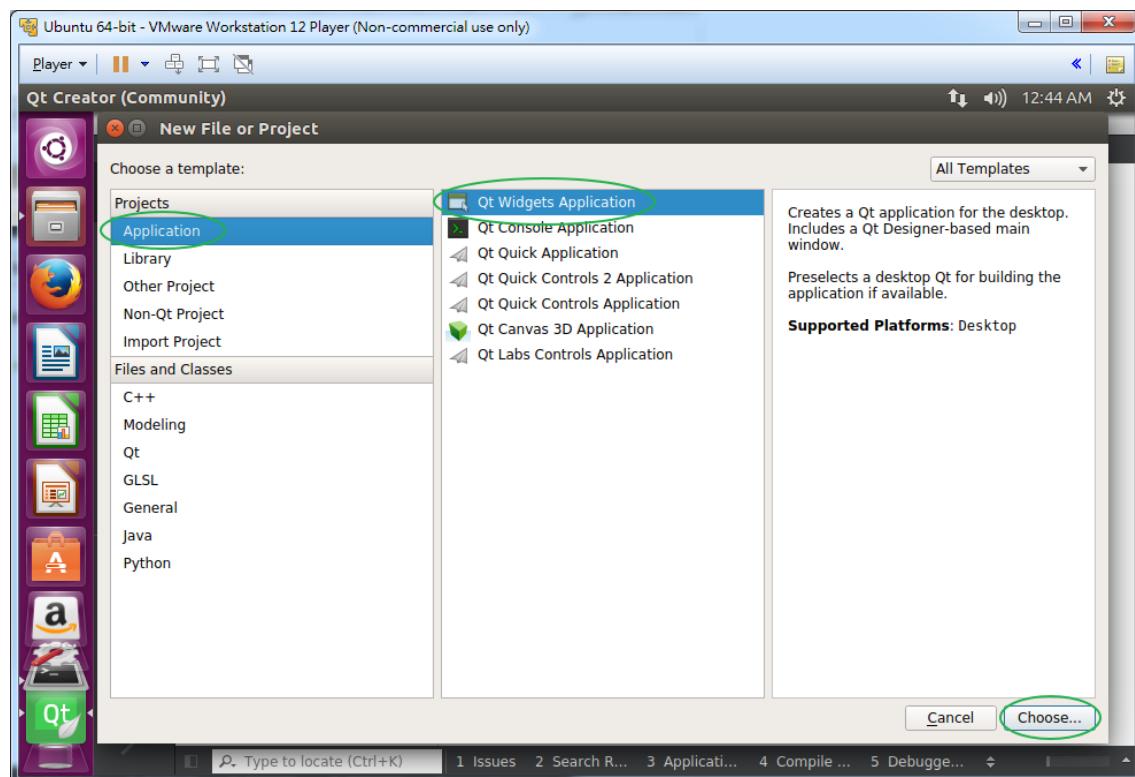


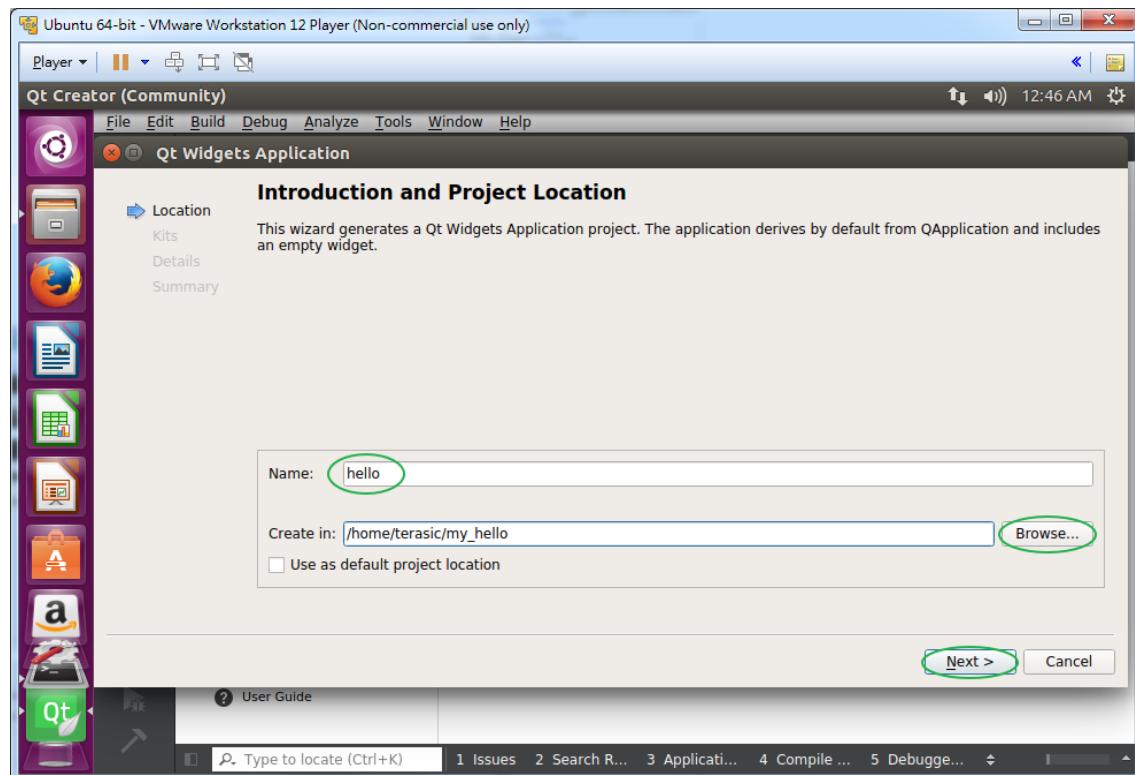
Figure 3-22 Open a New Project Dialog

In the New dialog, select "Applications" under Projects, and choose "Qt Widgets Application" as shown in **Figure 3-23**. Click "Choose..." to go to the next step.



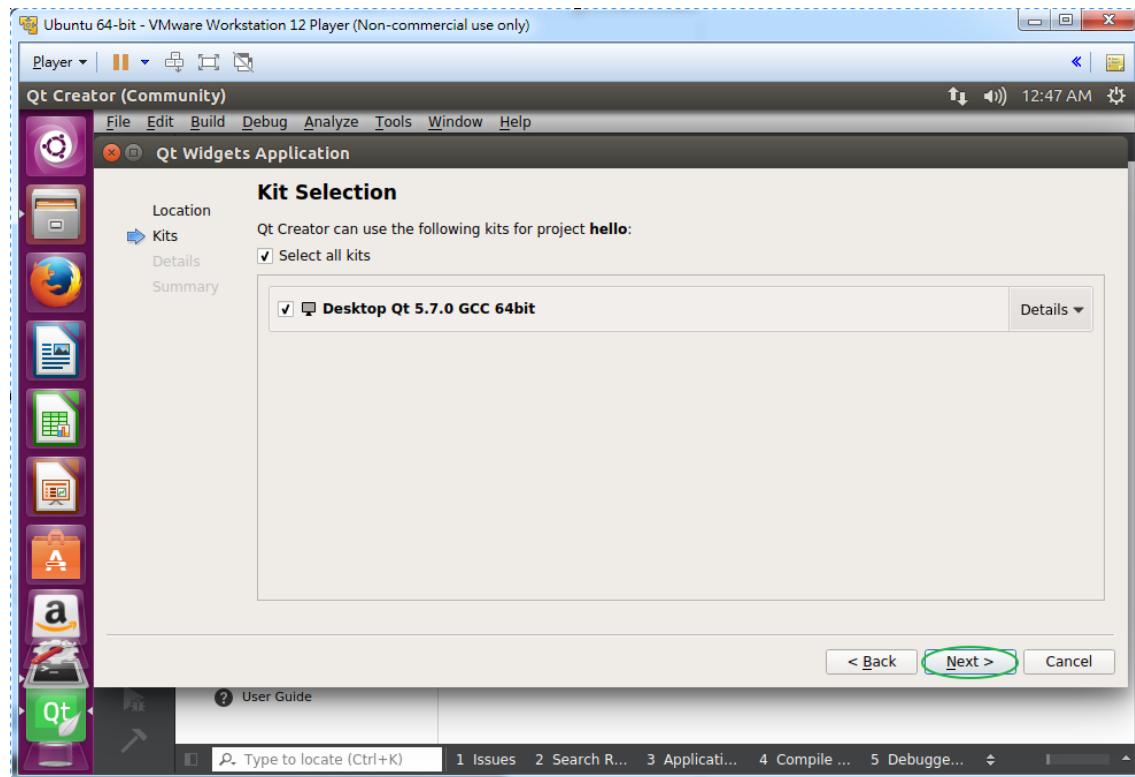
**Figure 3-23 Dialog of Creating a New Project**

In the **Qt Widgets Application** Dialog, specify the project name and project location, then click "Next >" as shown in **Figure 3-24**.



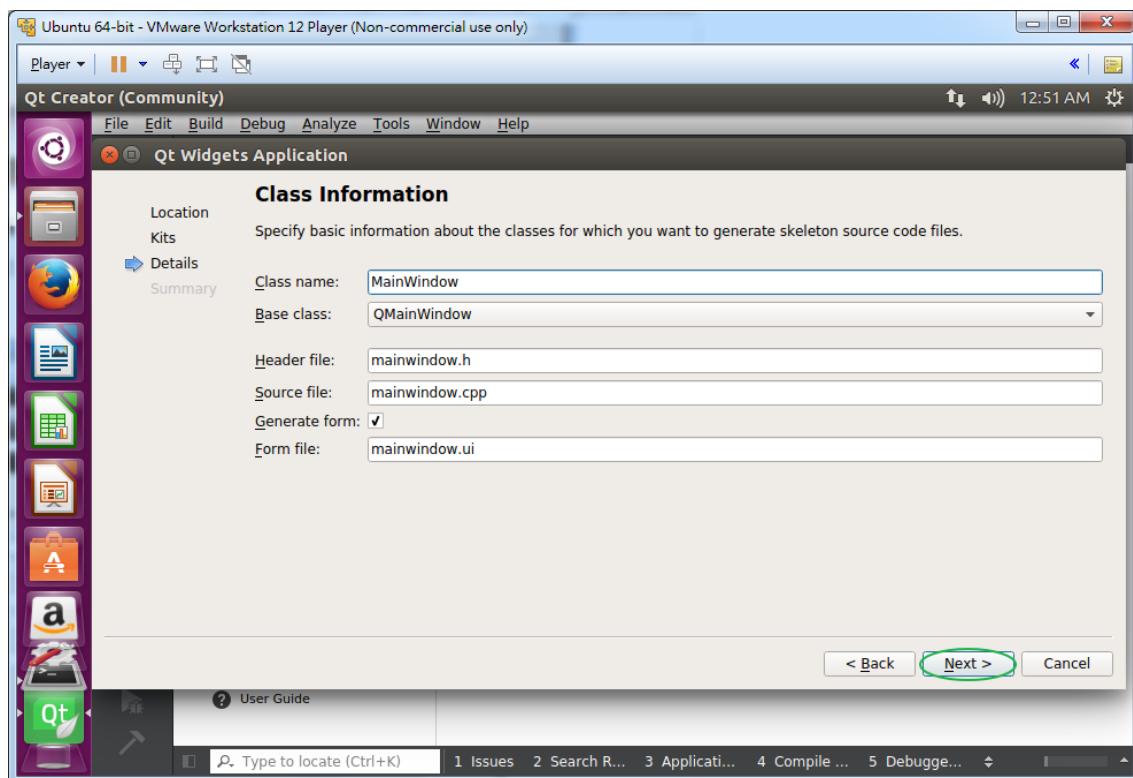
**Figure 3-24 Project Name and Location Dialog**

In the **Kit Selection** dialog, keep default settings and click "Next >" to go to next step as shown in **Figure 3-25**.



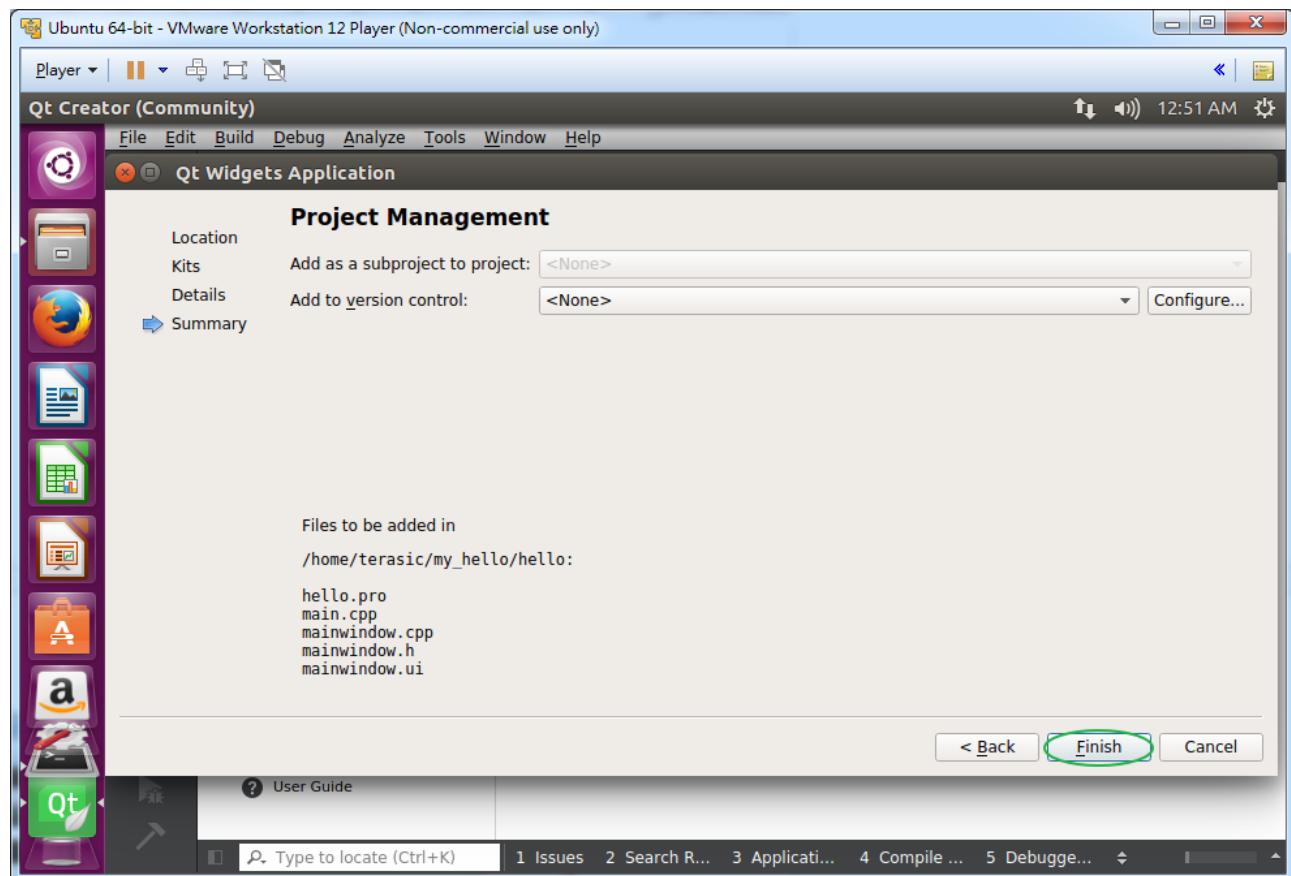
**Figure 3-25 Kit Selection Dialog**

In the **Class Information** dialog, click "Next >" as shown in **Figure 3-26** to proceed.



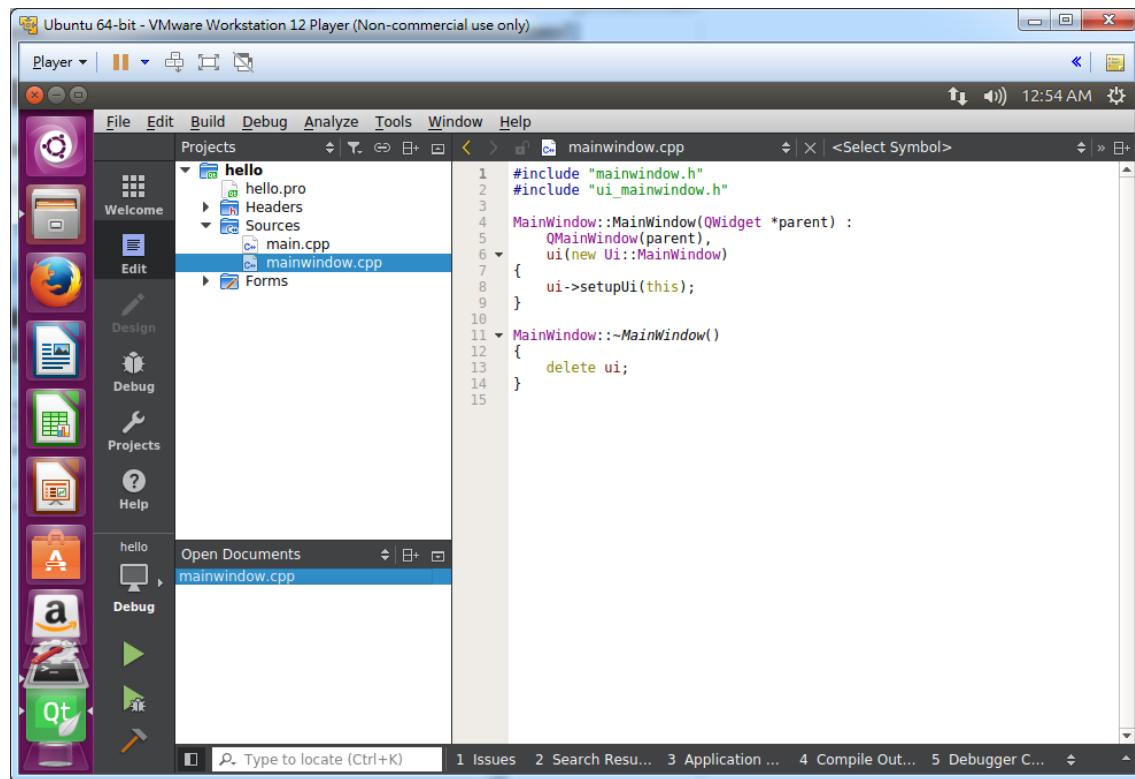
**Figure 3-26 Class Information Dialog**

In the **Project Management** dialog, click "Finish" as shown in **Figure 3-27** to proceed.



**Figure 3-27 Final Dialog of Qt Widgets Application**

**Figure 3-28** shows the hello project now has been created and the mainwindow.cpp and main.cpp files are included in the Sources folder under the hello project.



The screenshot shows the Qt Creator IDE interface. The left sidebar displays the project tree under 'Projects' with a single entry 'hello'. The 'Sources' folder contains 'main.cpp' and 'mainwindow.cpp'. The right pane shows the code editor with the 'mainwindow.cpp' file open. The code defines a MainWindow class that includes ui\_MainWindow.h and sets up the user interface using ui\_MainWindow::setupUi(this). It also includes a destructor that deletes the ui object.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

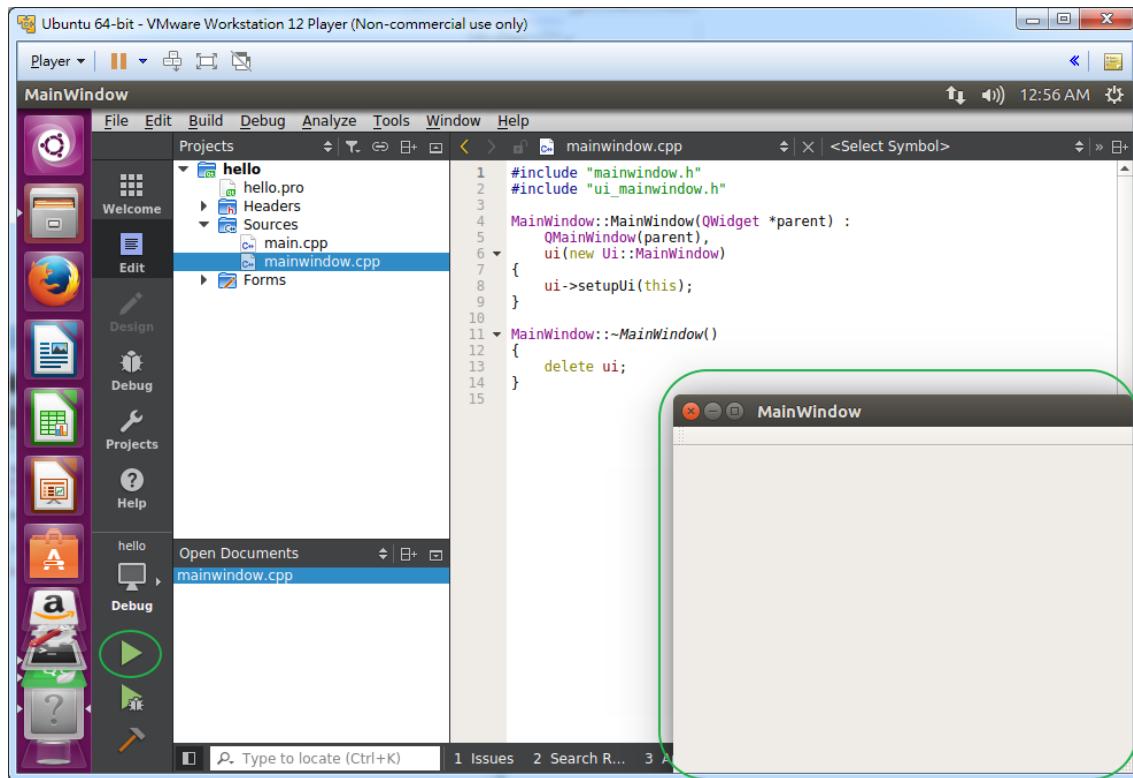
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

**Figure 3-28 mainwindow.cpp of the Hello project**

## ■ Build & Run

Click on the "Run" icon to build and run the Hello program. The GUI program should appear as shown in **Figure 3-29**.



**Figure 3-29 Run Hello Program**

# Chapter 4

## *Intel SoC Toolchain Installation*

Linaro ARM cross compiler toolchain is required while building the Qt application for the Intel SoC ARM on the Linux x64. In this chapter, we show the users steps needed to install the Linaro ARM cross compiler toolchain. Here is a quick look of the 2 simple steps performed on Linux x64:

- Download the Linaro ARM cross compiler toolchain and extract the file
- Include Toolchain path into the system environment variable \$PATH

### **4.1 Download and Install Toolchain**

Launch Linux terminal on the Linux x64 (CTRL+ALT+T) and execute the following commands to download and extract the Linaro ARM cross compiler toolchain,

```
$ cd ~  
$ wget -c https://releases.linaro.org/components/toolchain/binaries/\  
5.2-2015.11-2/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-  
gnueabihf.tar.xz  
$ tar xvf gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz
```

**Figure 4-1** shows the screenshot of toolchain download progress after typing in the wget command. It will take about 10 minutes to download the file.

The screenshot shows a terminal window titled "Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)". The terminal session is running on a user named "terasic" with the command "cd ~". The user then runs "wget -c https://releases.linaro.org/components/toolchain/binaries/5.2-2015.11-2/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86\_64\_arm-linux-gnueabihf.tar.xz". The output shows the download progress, including connection details, HTTP requests, and the final save command: "Saving to: 'gcc-linaro-5.2-2015.11-2-x86\_64\_arm-linux-gnueabihf.tar.xz'". The terminal window has a dark background and light-colored text.

**Figure 4-1 Linaro ARM cross compiler toolchain Download Progress**

The downloaded file is in the compressed format, so we need to decompress it before we can use it.

**Figure 4-2** shows the screenshot after toolchain download has been finished and has been extracted by the **tar** command. After extraction, the toolchain is located in the folder "`~/ gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf`"

```

Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)
Player Terminal File Edit View Search Terminal Help
terasic@ubuntu:~$ cd ~
terasic@ubuntu:~$ wget -c https://releases.linaro.org/components/toolchain/binaries/
> 5.2-2015.11-2/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz
--2016-12-21 01:22:04-- https://releases.linaro.org/components/toolchain/binaries/5.2-2015.11-2/ar
m-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz
Resolving releases.linaro.org (releases.linaro.org)... 52.221.254.75
Connecting to releases.linaro.org (releases.linaro.org)|52.221.254.75|:443... connected.
HTTP request sent, awaiting response... 302 FOUND
Location: https://publishing-ap-linaro-org.s3.amazonaws.com/releases/components/toolchain/binaries/
5.2-2015.11-2/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz?Signat
ure=zyEQPDVMFQLcL8amozIYTPrucZk%3D&Expires=1482312212&AWSAccessKeyId=AKIAIELXV2RYNAHFUP7A [followin
g]
--2016-12-21 01:22:06-- https://publishing-ap-linaro-org.s3.amazonaws.com/releases/components/tool
chain/binaries/5.2-2015.11-2/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabih
f.tar.xz?Signature=zyEQPDVMFQLcL8amozIYTPrucZk%3D&Expires=1482312212&AWSAccessKeyId=AKIAIELXV2RYNAH
FUP7A
Resolving publishing-ap-linaro-org.s3.amazonaws.com (publishing-ap-linaro-org.s3.amazonaws.com)...
54.231.242.168
Connecting to publishing-ap-linaro-org.s3.amazonaws.com (publishing-ap-linaro-org.s3.amazonaws.com)
|54.231.242.168|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165674656 (158M) [application/octet-stream]
Saving to: 'gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz'
gcc-linaro-5.2-2015.11-2 100%[=====] 158.00M 222KB/s in 12m 43s
2016-12-21 01:34:50 (212 KB/s) - 'gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz' saved
[165674656/165674656]
terasic@ubuntu:~$ tar xf gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf.tar.xz

```

**Figure 4-2 Toolchain Download Finished and Extracted by the Tar Command**

## 4.2 Setup Toolchain Path

Now, the toolchain path needs to be defined and added into the system variable **\$PATH**. We use the system editor tool **gedit** to add the path. Please refer to the following for more details to set up the toolchain path.

Launch terminal (CTRL+ALT+T) and type in the following command to edit the personal initialization file "`~/.bashrc`"

```
$gedit ~/.bashrc
```

**Figure 4-3** shows the screenshot that the "`~/.profile`" opened by gedit. Please add the following line,

```
export PATH=~/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf/bin:$PATH
```

to the **end** of the initialization file. Click "save" icon to save the file followed by clicking "close" icon (red circle with a x on the upper left corner) to terminate gedit tool.

```
# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "$( [ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''$s/^\s*[0-9]\+\s*//;s/[;&]$/s*alert$/'\''")"
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
if [ -f /usr/share/bash-completion/bash_completion ]; then
. /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
. /etc/bash_completion
fi
fi

export PATH=/home/linaro/gcc-linaro-5.2-2015.11-2-x86_64_arm-linux-gnueabihf/bin:$PATH
```

**Figure 4-3 .bashrc Opened by gedit**

In order for the path setting to take effect immediately, please type in "source ~/.bashrc" in the terminal or restart the OS.

## Qt Library Installation for Intel SoC

Up to now, you should have successfully installed the Linaro ARM cross compiler toolchain in the Linux x64 system. Now we install prebuilt Qt library for Intel SoC ARM on the Linux x64.

Here is a quick look at the 2 simple steps performed on Linux x64:

- Copy prebuilt Qt library from System CD
- Decompress prebuilt Qt library

### 5.1 Copy Prebuilt Qt Library from System CD

#### ■ Copy prebuilt Qt library to Shared Folder

The prebuilt Qt library is located on the System CD:

```
\Demonstrations\SoC_FPGA\ControlPanel\bin\qt5.5.1_for_intel_soc.tar.gz
```

Copy the prebuilt Qt library compressed file from the System CD to the shared folder, which is specified as in **Figure 2-39**, on Windows host. Then, launch the "Ubuntu 64-bit" virtual machine. After logging in Linux x64, launch a terminal (CTRL+ALT+T) and type the following command to go to the shared folder on Linux x64:

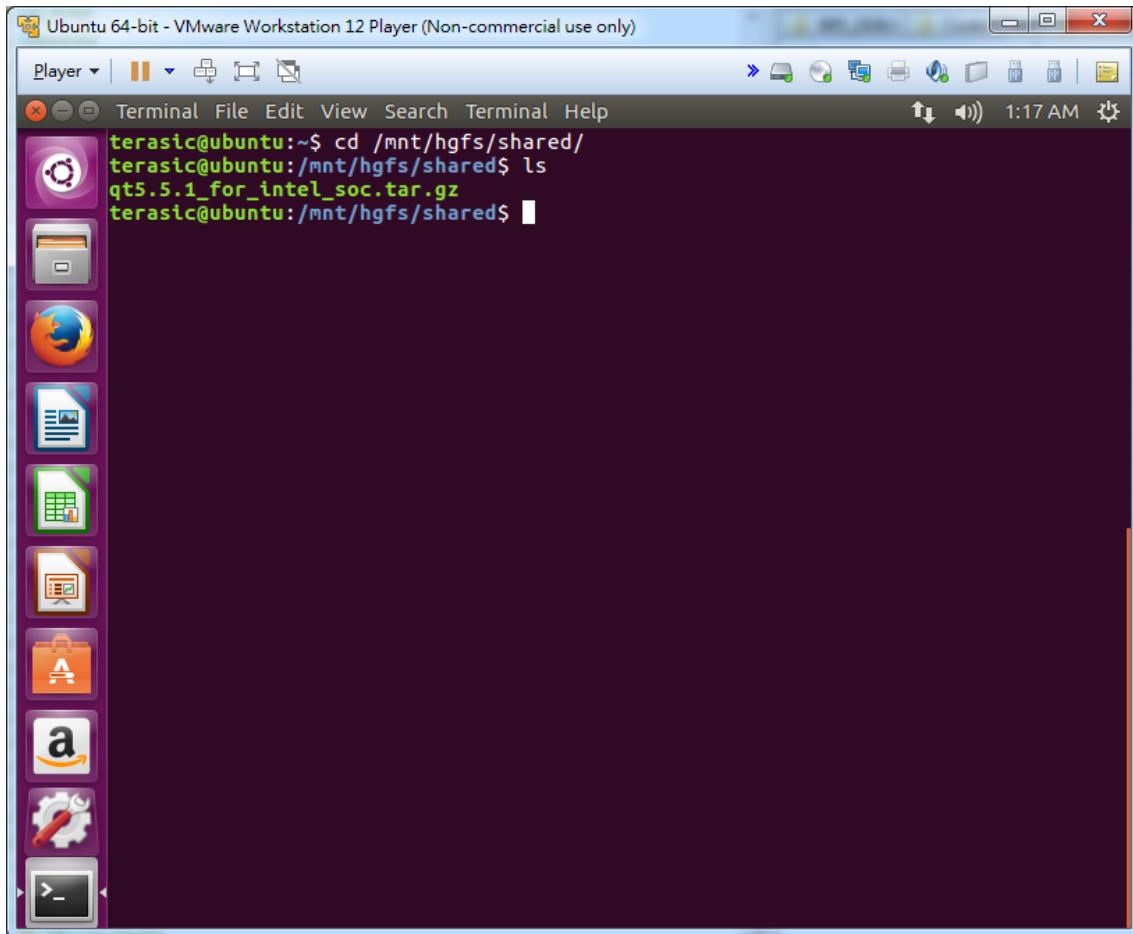
```
$cd /mnt/hgfs/shared
```

to go to the shared directory on Linux.

Now type in

```
$ ls
```

To see the "qt5.5.1\_for\_intel\_soc.tar.gz" file, as shown in **Figure 5-1**.



**Figure 5-1 ‘qt5.5.1\_for\_intel\_soc.tar.gz’ File of Prebuilt Qt Library**

## 5.2 Install Prebuilt Qt Library

## ■ Decompress Prebuilt Qt Library to System Path

After copy prebuilt Qt library to the shared folder, we need to install prebuilt Qt library to system path.

In the terminal, type the following command and press ENTER.

```
$ sudo tar zxvf qt5.5.1_for_intel_soc.tar.gz -C /opt
```

The system will prompt users to input a password. Please input your password (in this tutorial, the password is "123") and press ENTER as shown in **Figure 5-2**.

The screenshot shows a terminal window titled "Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)". The terminal window has a dark background and contains the following command and its output:

```
terasic@ubuntu:~$ cd /mnt/hgfs/shared/
terasic@ubuntu:/mnt/hgfs/shared$ ls
qt5.5.1_for_intel_soc.tar.gz
terasic@ubuntu:/mnt/hgfs/shared$ sudo tar zxvf qt5.5.1_for_intel_soc.tar.gz -C /opt
[sudo] password for terasic:
qt5/
qt5/qt5.5.1/
qt5/qt5.5.1/bin/
qt5/qt5.5.1/bin/qdbuscpp2xml
qt5/qt5.5.1/bin/uic
qt5/qt5.5.1/bin/qmlmin
qt5/qt5.5.1/bin/moc
qt5/qt5.5.1/bin/syncqt.pl
qt5/qt5.5.1/bin/qdoc
qt5/qt5.5.1/bin/qmllint
qt5/qt5.5.1/bin/rcc
qt5/qt5.5.1/bin/qlalr
qt5/qt5.5.1/bin/qmlexportscanner
qt5/qt5.5.1/bin/qdbusxml2cpp
qt5/qt5.5.1/bin/qmake
qt5/qt5.5.1/mkspecs/
qt5/qt5.5.1/mkspecs/linux-clang/
qt5/qt5.5.1/mkspecs/linux-clang/qplatformdefs.h
qt5/qt5.5.1/mkspecs/linux-clang/qmake.conf
qt5/qt5.5.1/mkspecs/win32-g++
qt5/qt5.5.1/mkspecs/win32-g++/qplatformdefs.h
qt5/qt5.5.1/mkspecs/win32-g++/qmake.conf
qt5/qt5.5.1/mkspecs/irix-g++
qt5/qt5.5.1/mkspecs/irix-g++/qplatformdefs.h
qt5/qt5.5.1/mkspecs/irix-g++/qmake.conf
qt5/qt5.5.1/mkspecs/qmodule.pri
qt5/qt5.5.1/mkspecs/linux-clang-libc++
qt5/qt5.5.1/mkspecs/linux-clang-libc++/qplatformdefs.h
```

**Figure 5-2 Decompress Prebuilt Qt Library**

# Chapter 6

## *Qt App for Intel SoC*

Chapter 6 describes how to cross-compile the Hello project we created back in chapter 3, such that the hello program can run on the target platform – DE10-Nano FPGA development board in this tutorial. Here we assume that Intel SoC toolchain and Qt library for Intel SoC FPGA development board have been successfully installed and are now available on the system.

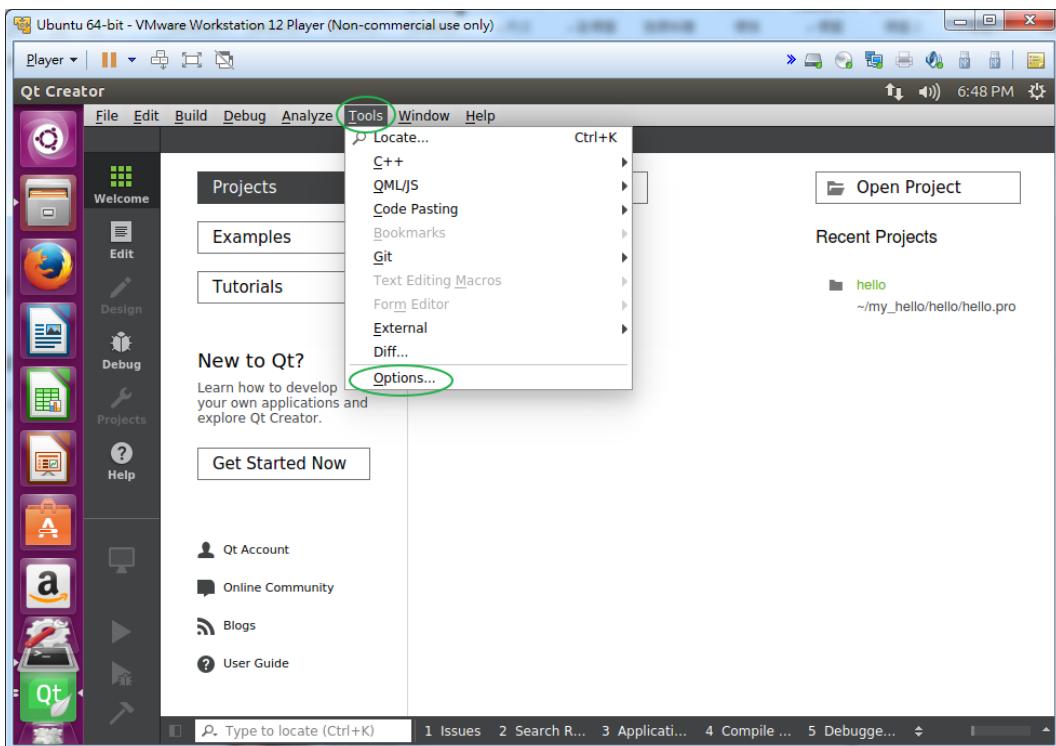
Here is a quick look of the 3 simple steps:

- Set up "Build & Run" in Qt Creator
- Cross-compile the Hello project
- Execute Hello program

### **6.1 Set up "Build & Run" in Qt Creator**

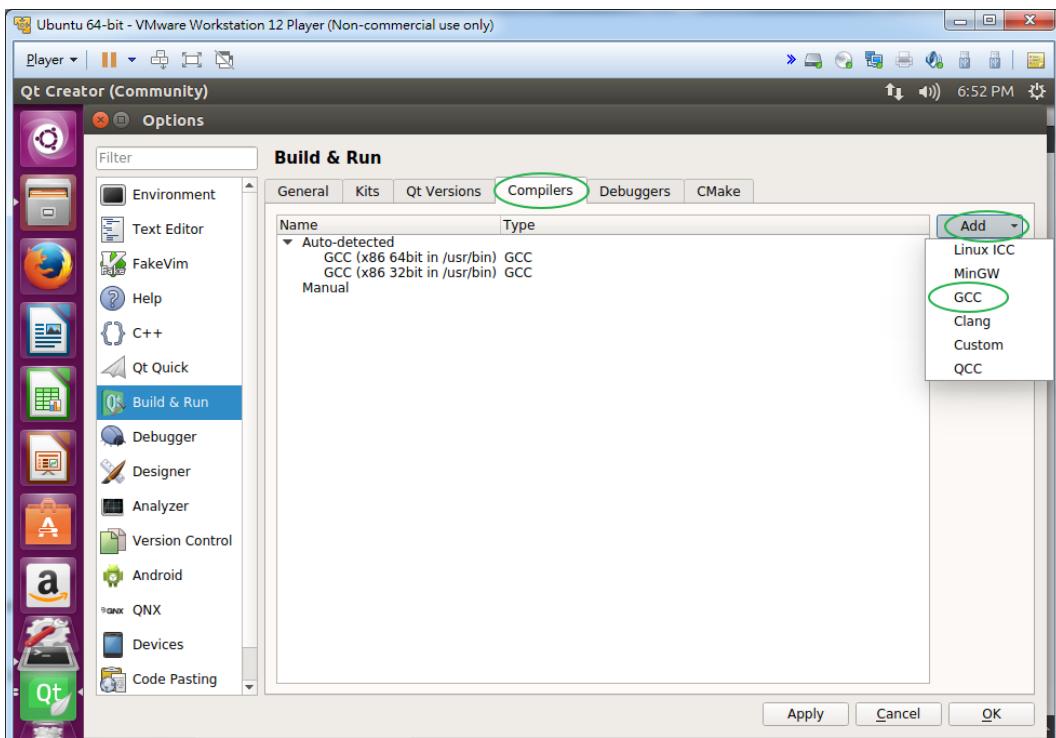
#### ■ Compiler Setup

First launch the Qt Creator, and select the menu item "Tools→Options..." as shown in **Figure 6-1** to open the **Option** dialog.



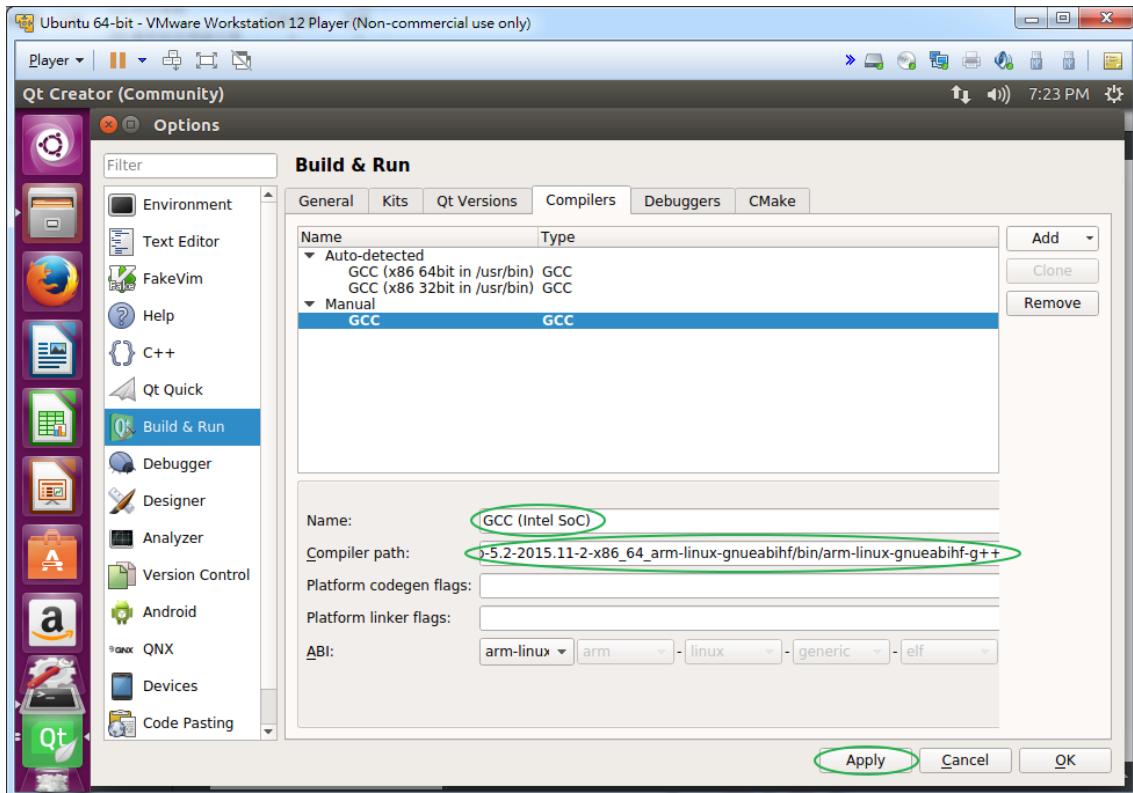
**Figure 6-1 Open Option Dialog**

To add the "GCC" compiler, in the Option dialog, select "Build & Run" item on the left and then select "Compilers" tab. Now you can select "GCC" item from the "Add" pull down menu as shown in **Figure 6-2**.



**Figure 6-2 Add GCC Compiler**

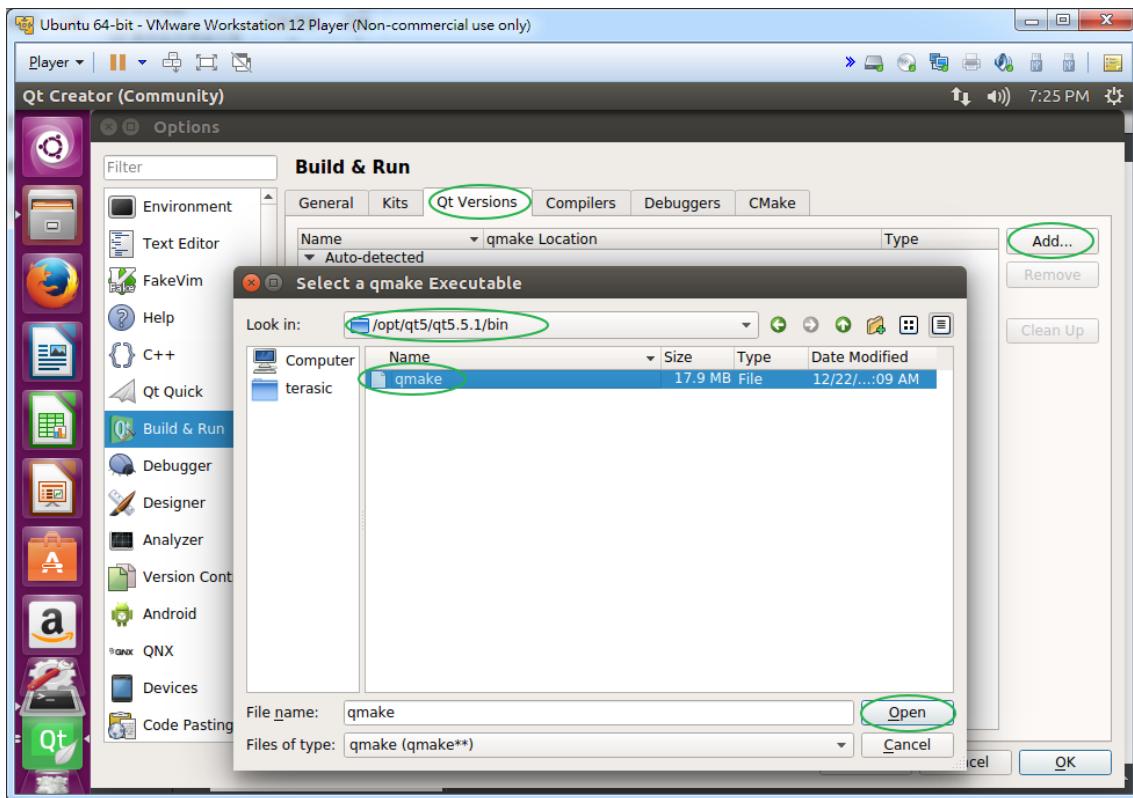
Next, you need to specify the GCC compiler details. First, please type in "GCC (Intel SoC)" in the Name option box and click "Browse..." to select the location of the compiler of Intel SoC ARM. The compiler is located at "/home/terasic/gcc-linaro-5.2-2015.11-2-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-g++". Note that in the path string, you should replace "terasic" with your linux user name. Then click "Apply" to finish the **Compiler** setup as shown in **Figure 6-3**.



**Figure 6-3 Specify Compiler Name and Path**

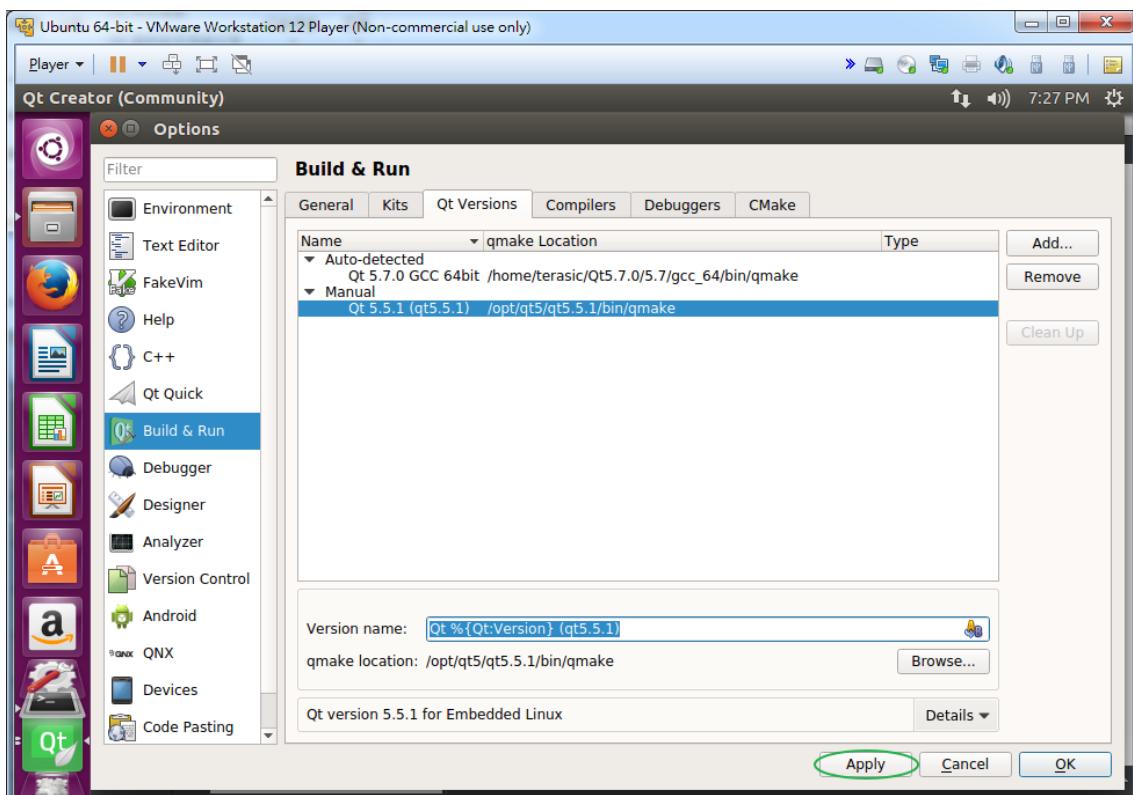
## ■ Qt Versions Setup

To add Qt Versions, first click on the "Qt Versions" tab and click "Add" as shown in **Figure 6-4**. When an **Open File** dialog appears to ask the location of qmake executable, please specify the location as: "/opt/qt5/qt5.5.1/bin/qmake".



**Figure 6-4 Add Qt Version**

Finally, click "Apply" to complete the **Qt Versions** setup as shown in **Figure 6-5**.



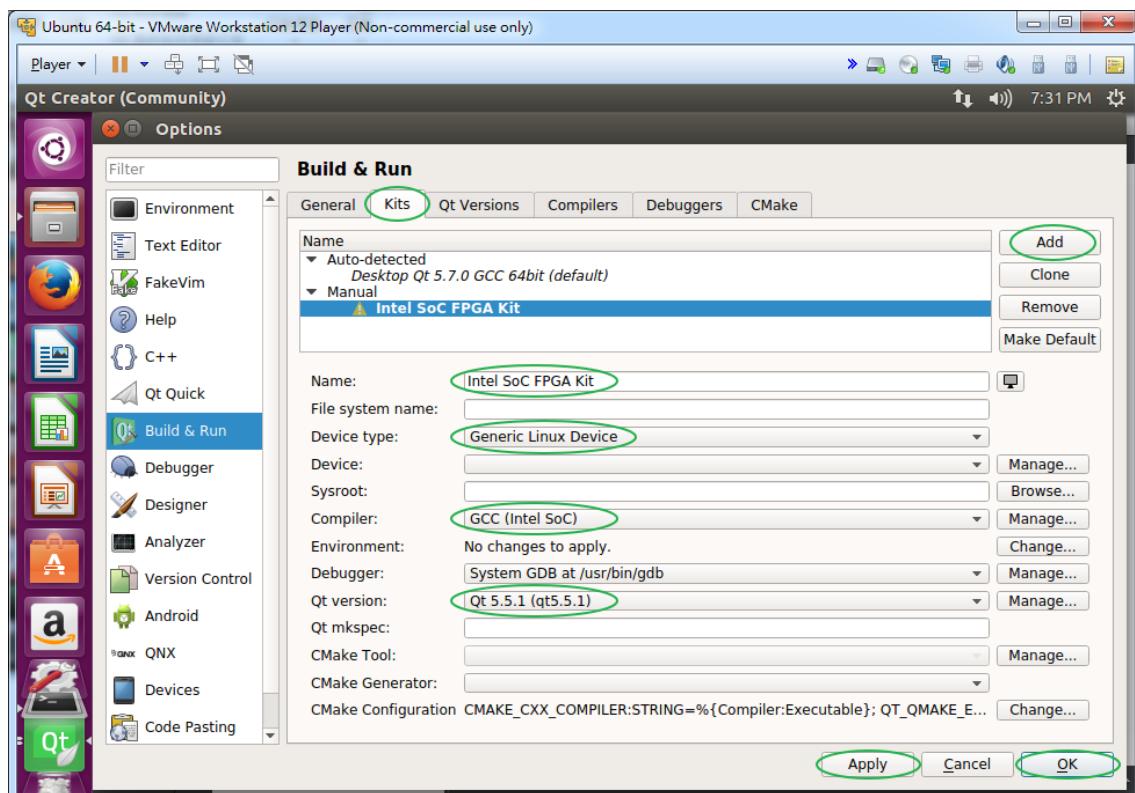
**Figure 6-5 Apply the Qt Version**

## ■ Kits Setup

To add Kits, first click on the "Kits" tab and click on "Add" as shown in **Figure 6-6**. Specify the kit detail as below:

- Name: Intel SoC FPGA Kit
- Device Type: Select "Generic Linux Device"
- Compiler: Select "GCC (Intel SoC)"
- Qt Version: Select "Qt 5.5.1 (qt5.5.1)"

Then, click "Apply" to finish **Kits** setup and click "OK" to finish Options setup.



**Figure 6-6 Add Intel SoC FPGA Kit**

## 6.2 Cross-Compile the Hello Project

Now, after properly setting up "Build & Run" settings, we are ready to cross-compile the Hello project. First, please launch the Qt Creator, and select the menu item "File→Recent Projects" to open the hello project as shown in **Figure 6-7**.

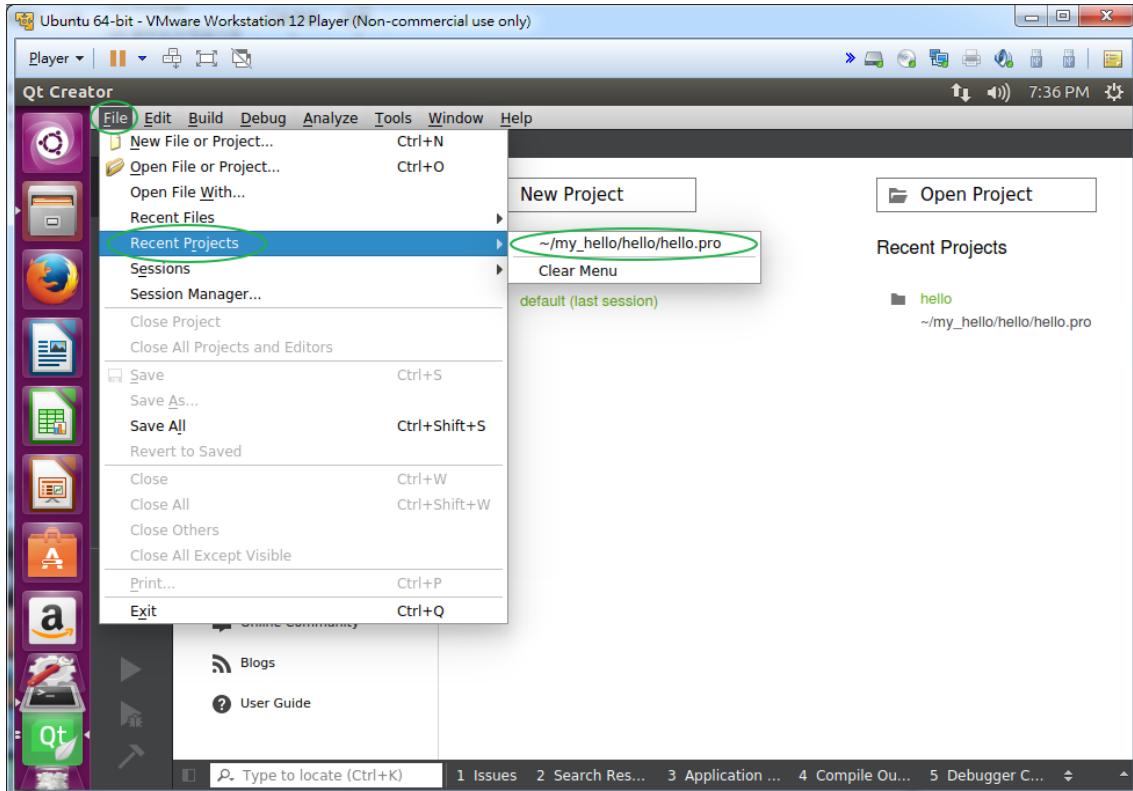
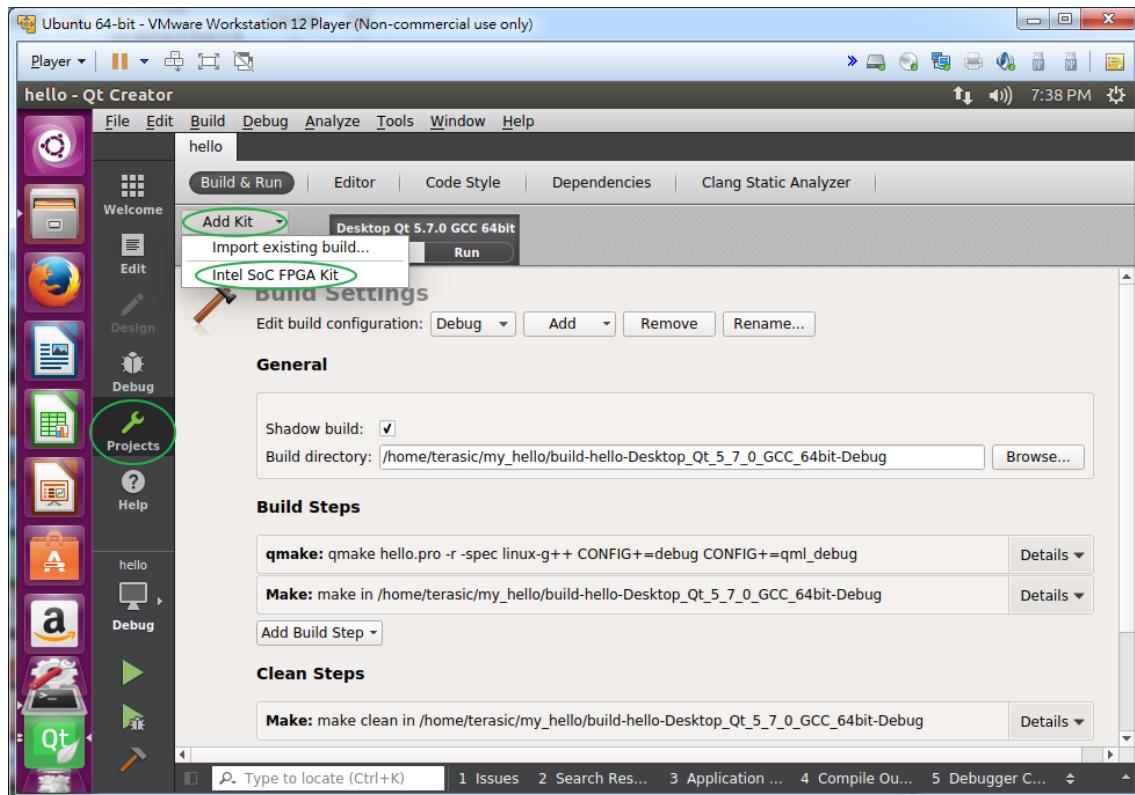


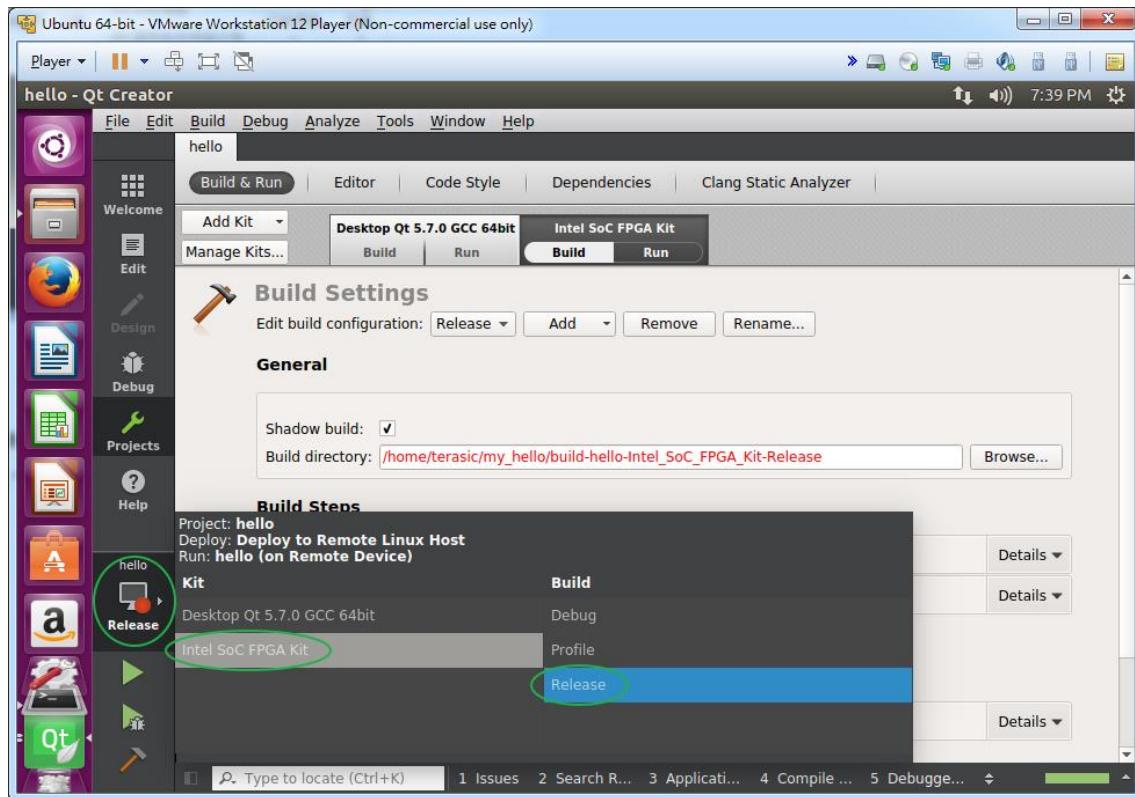
Figure 6-7 Open the Hello Project

To add "Intel SoC FPGA Kit" into the Hello project, click on the "Projects" icon and select "Intell SoC FPGA Kit" from the "Add Kit" pull-down menu, as shown in **Figure 6-8**.



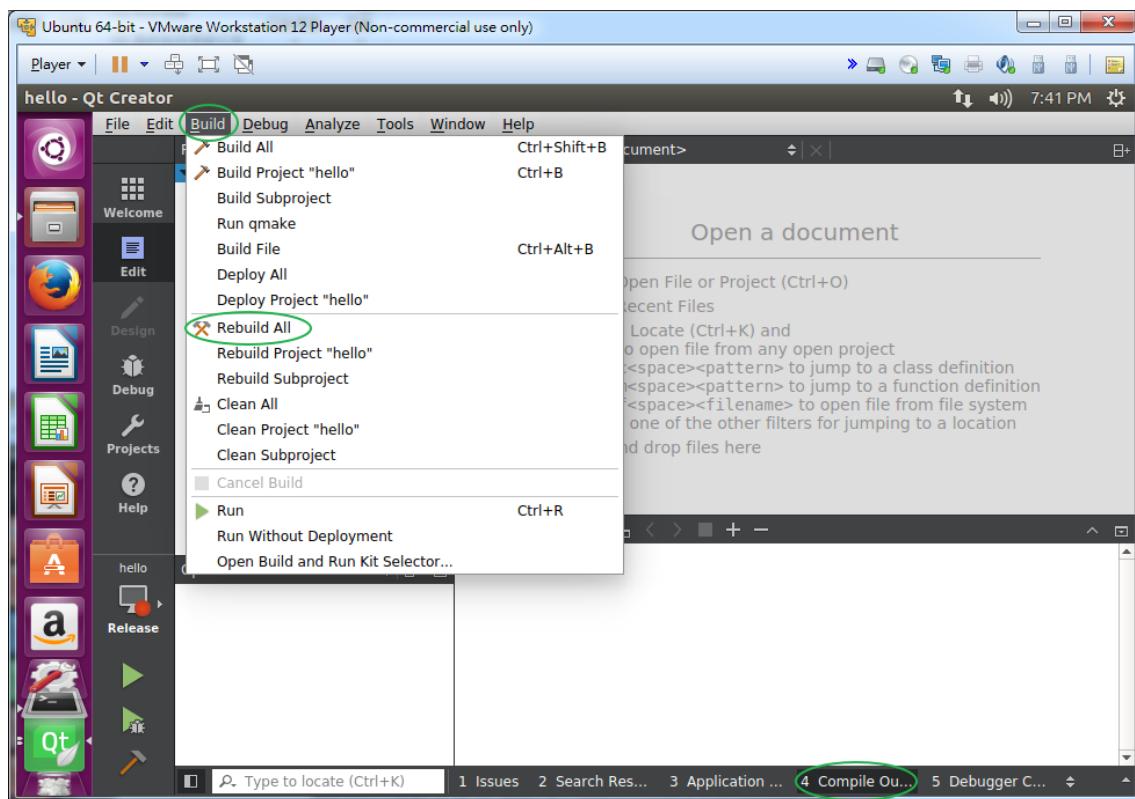
**Figure 6-8 Add Intel SoC FPGA Kit into the Hello Project**

To select **Release** type, click on "Release" icon and select "Intel SoC FPGA Kit" item under the **Kit** list menu and "Release" in **Build** list menu, as shown in **Figure 6-9**.



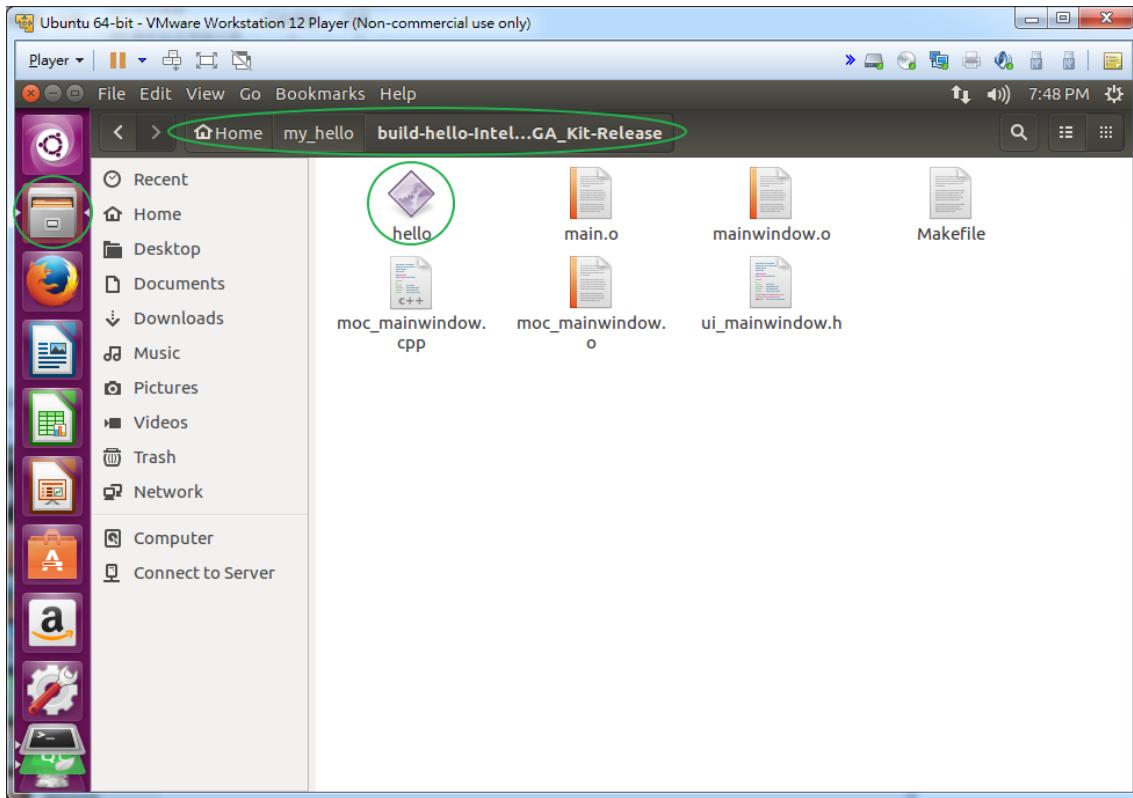
**Figure 6-9 Specify Release Setting**

To compile the hello project, click on "Compile Output" item on the bottom toolbar of the Qt Creator and select the menu item "Build→Rebuild All" as shown in **Figure 6-10**. While compiling, the system message will be simultaneously displayed on the "Compile Output" Windows.



**Figure 6-10 Build Hello Project**

When the build process has been completed, the output execution file can be found under the folder "/home/**terasic**/my\_hello/build-hello-Intel\_SoC\_FPGA\_Kit-Release/hello". Note that in the path string, you should replace "terasic" with your linux user name as shown in **Figure 6-11**.



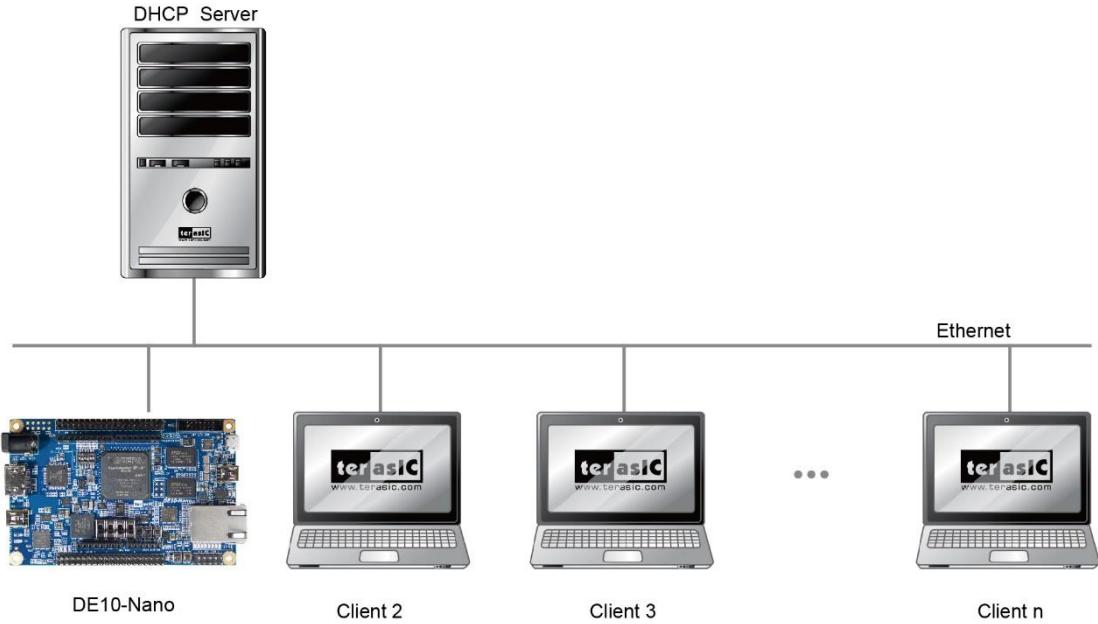
**Figure 6-11 Location of the Output Hello Execution File**

## 6.3 Execute Hello Program

To execute the Hello program on the DE10-Nano FPGA board, we need to copy the **hello** execution file to the Intel SoC FPGA Board which runs on Linux. In this demonstration, we use Linux "scp" command to remote copy these files from host PC to the DE10-Nano FPGA development board.

### ■ Setup Network for Intel SoC FPGA Board

First, use a RJ45 cable to connect the Intel SoC FPGA development board to an Ethernet network with a DHCP server, like **Figure 6-12**.



**Figure 6-12 Connect Intel SoC FPGA Board To the Ethernet Network**

Then, use the microSD card which Linux LXDE Desktop image is written to, described in chapter 1, to boot the Intel SoC FPGA development board. After booting LXDE Desktop, launch a terminal on the LXDE Desktop (CTRL+ALT+T) and type in "dhclient -v eth0" command to request an Ethernet IP Address from the DHCP server as shown in [Figure 6-13](#).

```
root@DE10_NANO:/root# dhclient -v eth0
Internet Systems Consortium DHCP Client 4.3.3
Copyright 2004-2015 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/fe:91:89:03:17:1b
Sending on LPF/eth0/fe:91:89:03:17:1b
Sending on Socket/fallback
DHCPOFFER on eth0 to 255.255.255.255 port 67 interval 3 (xid=0x78ceb156)
DHCPREQUEST of 192.168.1.191 on eth0 to 255.255.255.255 port 67 (xid=0x56b1ce78)
DHCPACK of 192.168.1.191 from 192.168.1.1
DHCPACK of 192.168.1.191 from 192.168.1.1
bound to 192.168.1.191 -- renewal in 1527 seconds.
root@DE10_NANO:/root#
```

**Figure 6-13 Linux dhclient Command**

To get a IP address from the DHCP server, type Linux "ifconfig" command to find the assigned IP address, as shown in [Figure 6-14](#). In this case, the assigned IP address is "192.168.1.191".

```

root@DE10_NANO:/root# ifconfig
eth0      Link encap:Ethernet HWaddr fe:91:89:03:17:1b
          inet addr:192.168.1.191 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::9a34:b532:2ff6:8f8e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:5227 errors:0 dropped:0 overruns:0 frame:0
          TX packets:846 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:579109 (579.1 KB) TX bytes:1021798 (1.0 MB)
          Interrupt:37 Base address:0x4000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:240 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:17760 (17.7 KB) TX bytes:17760 (17.7 KB)

```

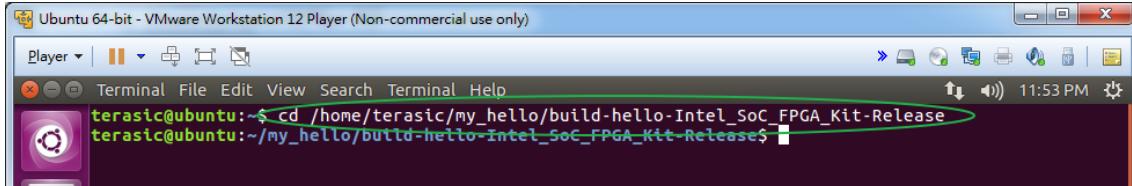
**Figure 6-14 Use Linux ifconfig Command to Check Assigned Ethernet IP Address**

## ■ Copy Files from PC Host to Intel SoC FPGA Board

Now, back to the host PC part. Please make sure the host PC is connected to the Ethernet which DE10-Nano FPGA development board also is connected to. Launch a terminal on the Linux x64, and type in:

```
$ cd /home/terasic/my_hello/build-hello-Intel_SoC_FPGA_Kit-Release
```

to go to the directory where the hello execution file is located, as shown in **Figure 6-15**. Note, in the path string, you should replace "terasic" with your linux user name.



**Figure 6-15 Go to the Hello Program Folder**

Then, type:

```
$ scp hello root@192.168.1.191:/home/root
```

To remote copy the **hello** program to the /home/root directory of the file system of DE10-Nano FPGA development board, as shown in **Figure 6-16**. Note that the IP address 192.168.1.191 should be changed to the actual IP address assigned.

During the first time of the connection, system will ask "Are you sure you want to continue connecting (yes/no)?". Please type in "yes" and press ENTER. After confirming, the system will ask you to input password for the remote machine, please type "**terasic**" (the default setting with the microSD image) and press ENTER.

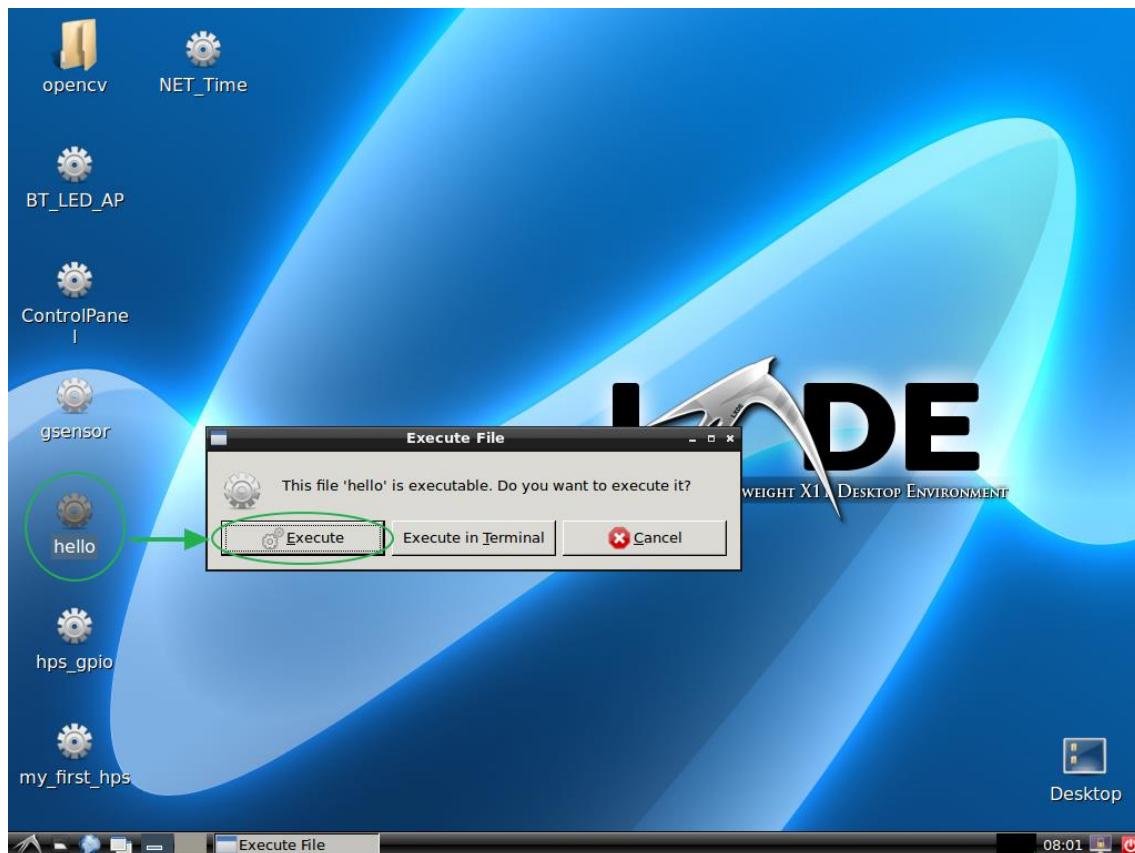
```

Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)
Player Terminal File Edit View Search Terminal Help
terasic@ubuntu:~ cd /home/terasic/my_hello/build-hello-Intel_SoC_FPGA_Kit-Release
terasic@ubuntu:~/my_hello/build-hello-Intel_SoC_FPGA_Kit-Release$ scp hello root@192.168.1.191:/ho
me/root
The authenticity of host '192.168.1.191 (192.168.1.191)' can't be established.
ECDSA key fingerprint is SHA256:YAVGTDiDJS5Pwbxi04bkfYZtfVcvYKJiITzwVRnIJP4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.191' (ECDSA) to the list of known hosts.
root@192.168.1.191's password:
100%   26KB  25.8KB/s  00:00
terasic@ubuntu:~/my_hello/build-hello-Intel_SoC_FPGA_Kit-Release$
```

**Figure 6-16 Remote Copy the Hello Program**

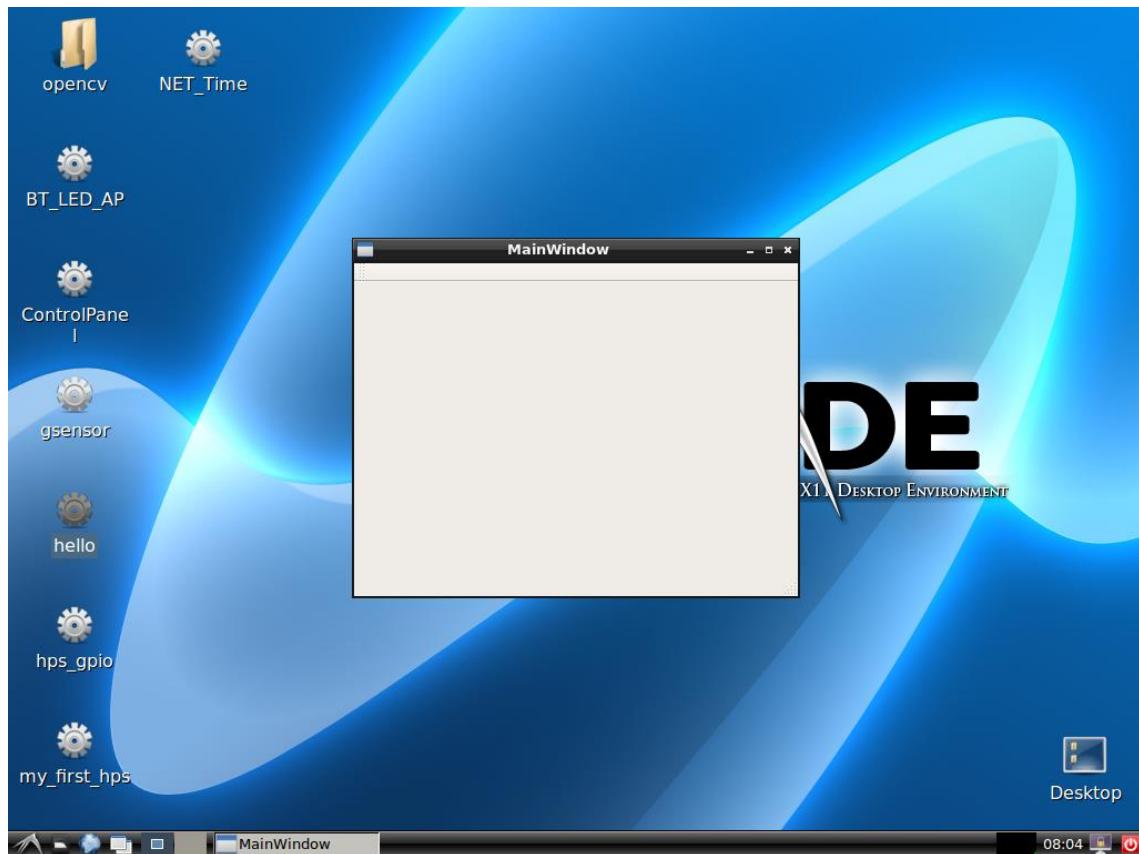
## ■ Launch Hello Program

Please double click "hello" icon and click "Execute" button to launch the Control Panel as shown in **Figure 6-17**.



**Figure 6-17 Screenshot of the Hello Program**

If the Hello program is launched successfully, you should see the Hello windows as shown in **Figure 6-18**.



**Figure 6-18 Screenshot of the Hello Program**

# Chapter 7

## Control Panel Quartus Project

We have so far shown how to build and run hello program on the Intel SoC FPGA development board with Qt library. In Chapter 7, we show how to build the Quartus project "Control Panel" over on the Windows host, translate the .sof to .rbf, and copy the .rbf file to the microSD Card such that the FPGA can be configured while booting Linux. We also introduce how to use **HPS** component in Qsys in order for the Linux program to access Qsys Avalon Memory-Mapped bus through **HPS** component.

Quartus 16.0 or later which supports Cyclone V Device on Windows is required in the Control Panel Quartus Project can be downloaded from <https://www.altera.com/downloads/download-center.html> or as shown in **Figure 7-1**.

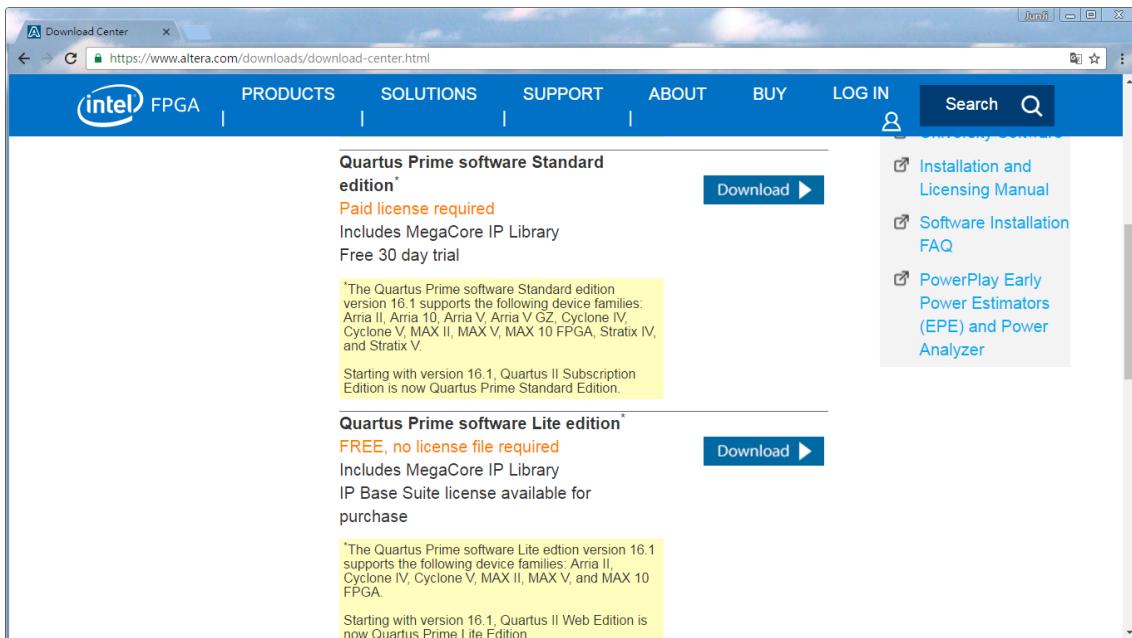
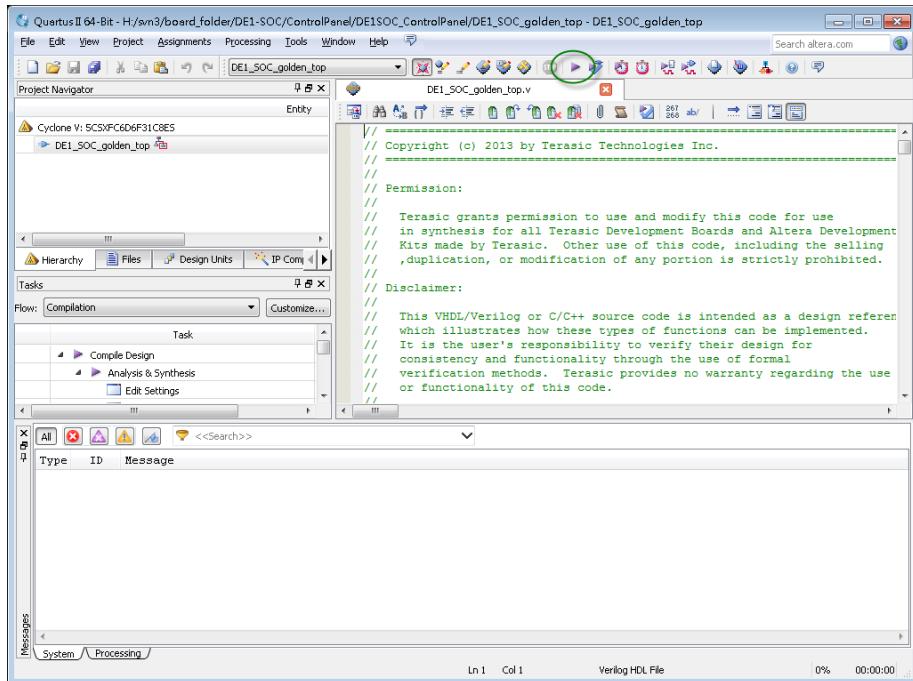


Figure 7-1 Quartus Download Web Page

### 7.1 Build Quartus Project of Control Panel

The Quartus project of Control Panel is located on the system CD:

"**Demonstrations\SoC\_FPGA\ControlPanel\Quartus**". Please copy the folder to the Windows host and open it with Quartus, as shown in **Figure 7-2**. Click "Start Compilation" icon to start to compile.



**Figure 7-2 Quartus Project of the Control Panel**

When finished compiling, **DE10\_NANO\_SOC\_FB.sof** is generated under the **output\_files** folder of Quartus project.

## 7.2 Test FPGA Configuration File

When Intel SoC FPGA boots from the microSD Card, it also reads **soc\_system.rbf** in the microSD Card to configure the FPGA. Therefore, we now show how to translate the .sof to .rbf.

### ■ Generate **soc\_system.rbf**

First, we need to translate the .sof to .rbf with the following command

```
$ quartus_cpf -c -o bitstream_compression=on \
DE10_NANO_SOC_FB.sof soc_system.rbf
```

We have provided the batch file "**sof\_to\_rbf.bat**" which implements the translation command for the users convenience. The batch file is located in the **output\_files** folder of the Quartus project. Execute the batch file will start the translation. When the file translation is complete, **soc\_system.rbf** will be generated on the same folder.

## ■ Update microSD card

Next, we need copy the **soc\_system.rbf** to the microSD Card with the following procedures:

1. Insert the microSD Card in the Windows Host
2. Delete **soc\_system.rbf** in the microSD Card if there is one
3. Copy your **soc\_system.rbf** into the microSD Card

## ■ Test

Finally, boot the DE10-Nano FPGA development board with the updated microSD Card. When the Linux boot process is complete, Double click "ControlPanel" icon and click "Execute" button to test if everything works well.

# 7.3 More on the Quartus Project

The Control Panel circuit is built by Qsys. The HPS Linux program accesses **Avalon Memory Mapped Bus** through the **HPS** component. Also, the FPGA controller can use HPS resource, like DDR3 SDRAM, through the **HPS** component. The HPS configuration is board-peripheral- dependent, so users don't need to modify the HPS setting if the board has not been changed.

## ■ Linux Access FPGA Controller

The HPS Linux program accesses **Avalon Memory Mapped Slave** port through HPS **AXI Master** port as shown **Figure 7-3**. If you wish that Linux program can access a Qsys instance, just connect its **Avalon Memory Mapped Slave** port of the controller to HPS **AXI Master** port.

Linux program access specific **Avalon Memory Mapped Slave** port based on its port address, so please make sure the port address are not overlapped with each other. We strongly recommend that the developer properly defines the port addresses and lock port addresses in Qsys.

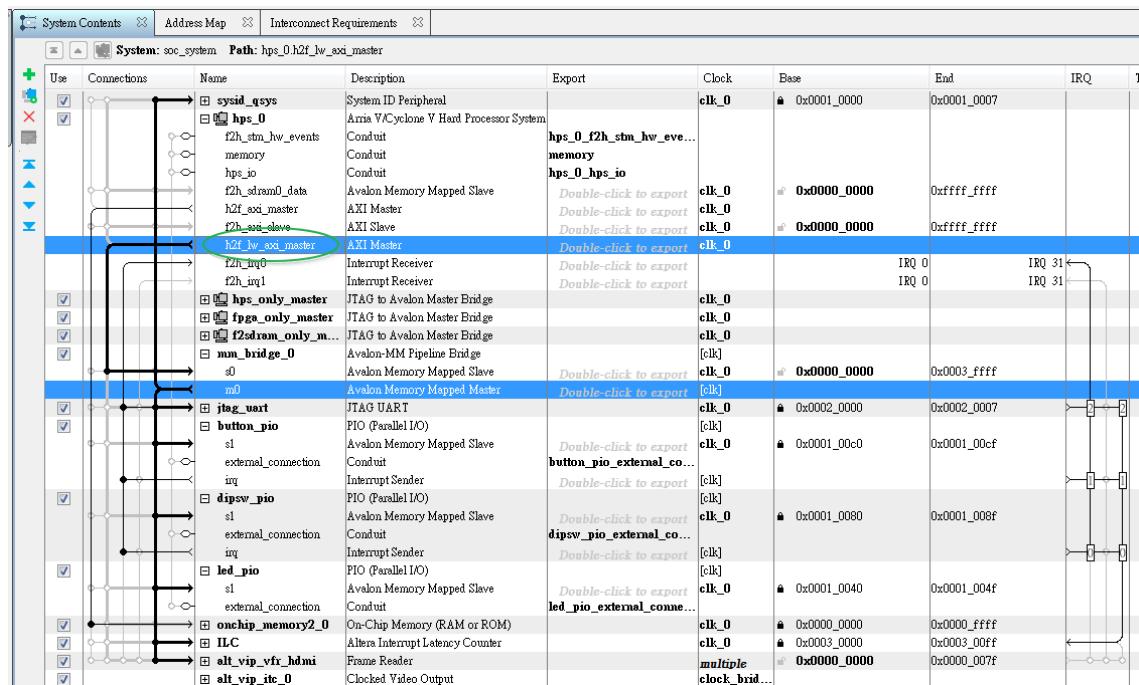


Figure 7-3 HPS AXI Master Port

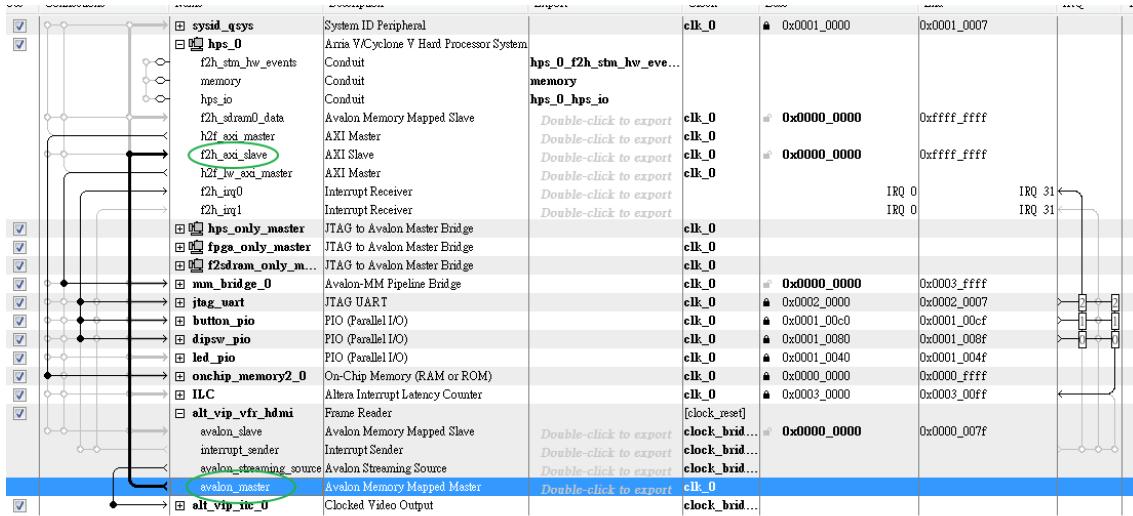
In this tutorial, the slave port of LED, Button, Switch and VIP frame reader are connected to the **HPS h2f\_lw\_axi\_master AXI Master** port. Use Qsys "Address Map" tab can view the connection summary and associated address, as shown in [Figure 7-4](#).

System	Path	hps_0_h2f_lw_axi_master	alt_vip_vfr_hdmi.avalon_master	f2sdram_only_m...
mm_bridge_0.s0		0x0000_0000 - 0x0003_ffff		
alt_vip_vfr_hdmi.avalon_slave via mm_...		0x0000_0000 - 0x0000_007f		
sysid_qsys.control_slave via mm_bridge...		0x0001_0000 - 0x0001_0007		
led_pio.s1 via mm_bridge_0		0x0001_0040 - 0x0001_004f		
dipsw_pio.s1 via mm_bridge_0		0x0001_0080 - 0x0001_008f		
button_pio.s1 via mm_bridge_0		0x0001_00c0 - 0x0001_00cf		
jtag_uart.avalon_jtag_slave via mm_brid...		0x0002_0000 - 0x0002_0007		
ILC.avalon_slave via mm_bridge_0		0x0003_0000 - 0x0003_00ff		
<b>hps_0_f2h_axi_slave</b>			<b>0x0000_0000 - 0xffff_ffff</b>	
ILC.avalon_slave				
alt_vip_vfr_hdmi.avalon_slave				
button_pio.s1				
dipsw_pio.s1				
hps_0_f2h_sdram0_data				0x0000_0000 -
jtag_uart.avalon_jtag_slave				
led_pio.s1				
onchip_memory2_0.s1				
sysid_qsys.control_slave				

Figure 7-4 HPS h2f\_lw\_axi\_master Connection and Address Map

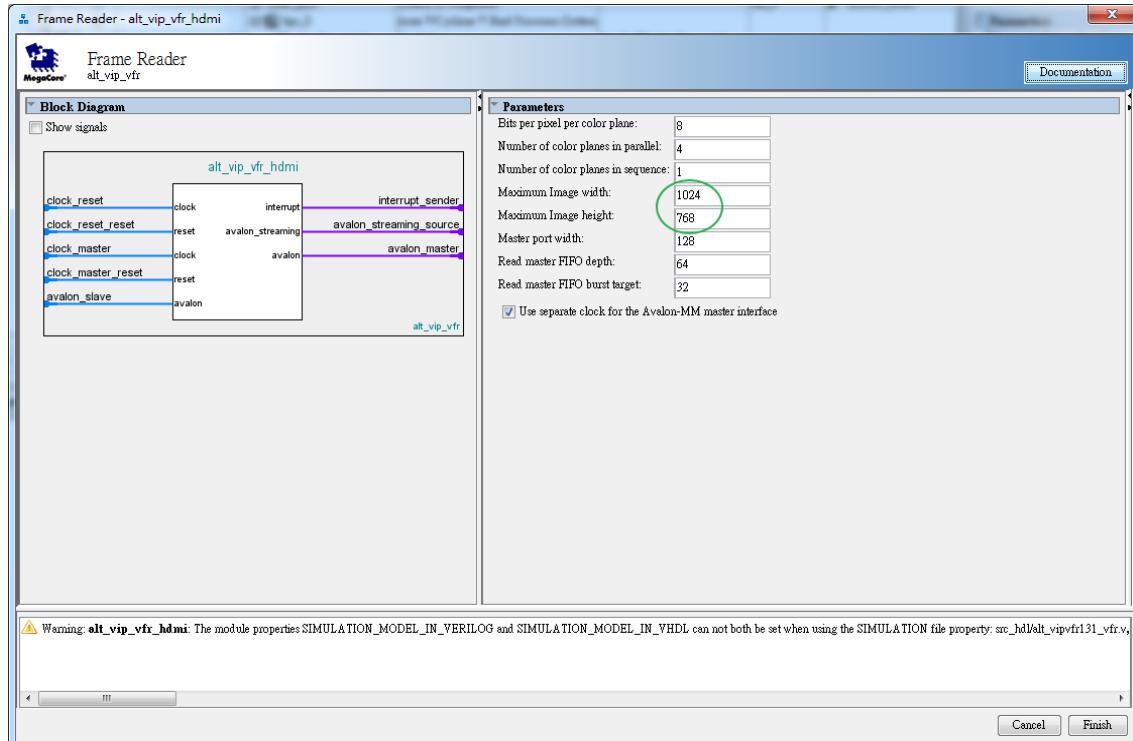
## ■ Hardware of Linux Frame Buffer Display

VIP Frame Reader is used to implement the Frame Buffer Display of Linux. The Frame Reader uses HPS's DDR3 SDRAM as frame buffer, so we need to connect the **Avalon Memory Mapped Master** port of Frame Reader instance to the **AXI Slave** port of HPS instance, as shown in **Figure 7-5**.



**Figure 7-5 HPS AXI Master Port**

Figure 7-6 shows Frame Reader instance setting. The display resolution is set as 1024x768.



**Figure 7-6 Frame Reader Setting**

# Chapter 8

## Control Panel Qt Project

Chapter 8 describes how to build the Control Panel Qt project by the Qt Creator on the Linux x64. Here, we assume the Qt library for Intel SoC ARM built in previous chapters is still in place.

Control Panel Qt project uses Intel SoC FPGA hwlib library source code which is included in the Intel SoC EDS (Embedded Development Suite). Therefore make sure to install Intel SoC EDS before compiling the Qt Project.

### 8.1 Install Intel SoC EDS on Linux x64

Intel SoC EDS can be downloaded from <http://dl.altera.com/soceds/16.0/?edition=standard> or the Intel Download Center by clicking on the **Embedded Software** item and clicking on the **SoC EDS** item as shown in **Figure 8-1**.

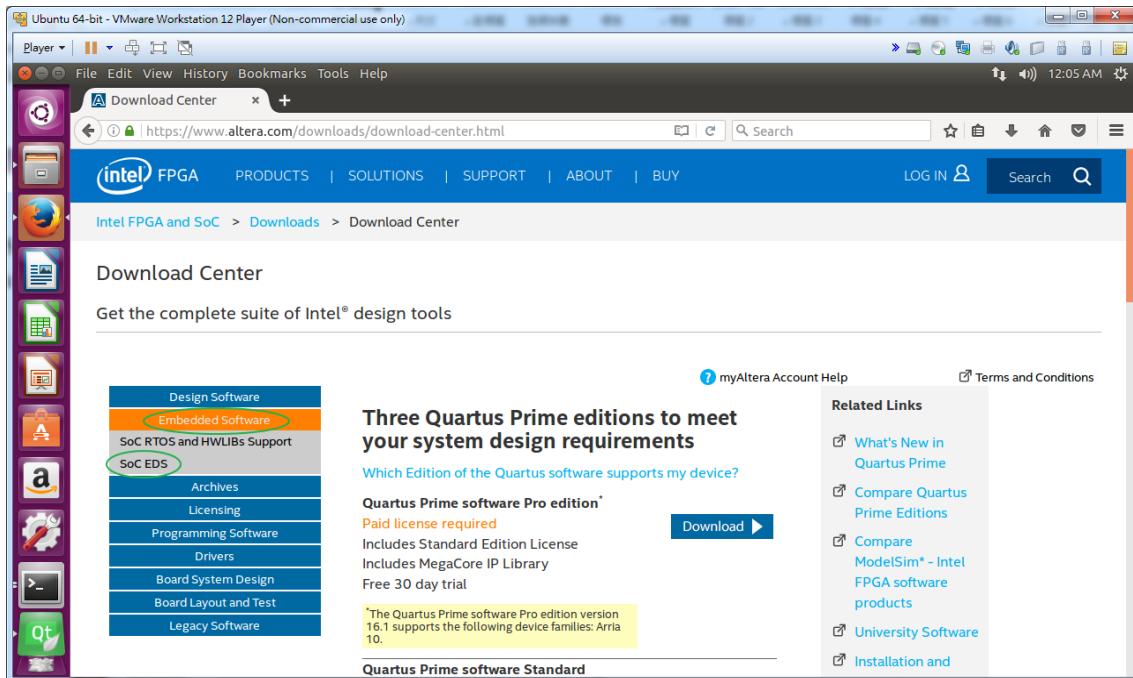
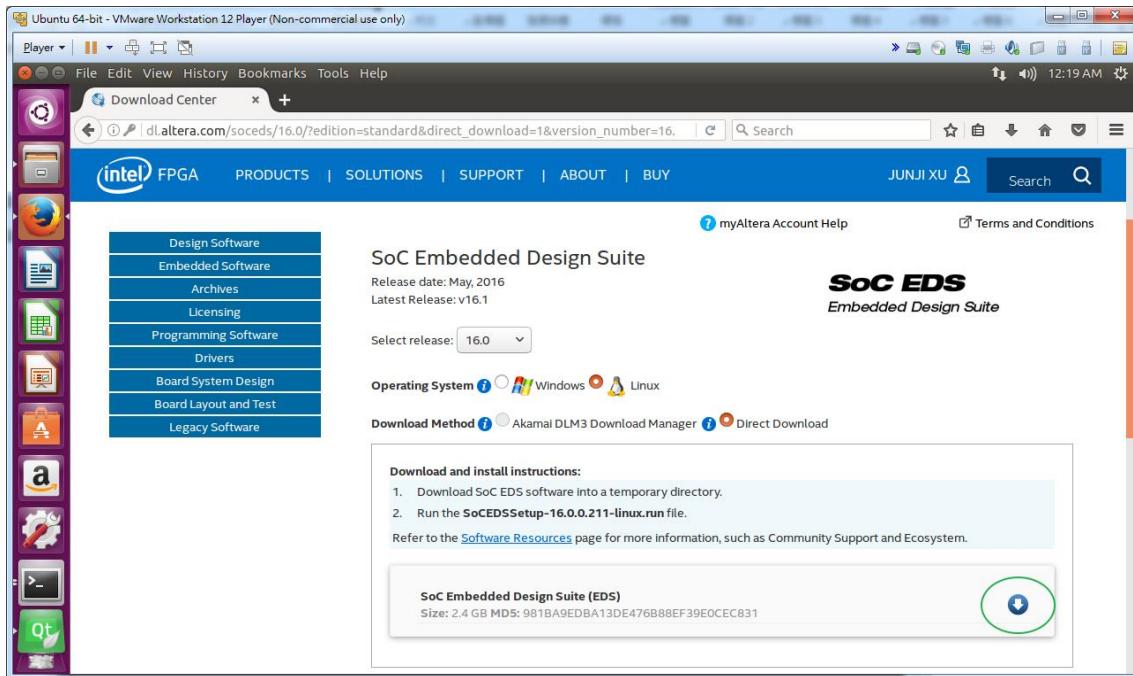


Figure 8-1 Intel SoC EDS Download Web Page

In the SoC Embedded Design Suite Download page, click the "Download" icon to download the SoC EDS installer as shown in **Figure 8-2**.



**Figure 8-2 Intel SoC EDS Download Web Page**

The downloaded SoC EDS installer "**SoCEDSSetup-16.0.0.211-linux.run**" is located in the directory "`~/Downloads`". To launch the installer, first launch Linux terminal on the Linux x64 (CTRL+ALT+T), and type in:

```
$ cd ~/Downloads
```

to go to "`~/Downloads`" directory.

Then, type in:

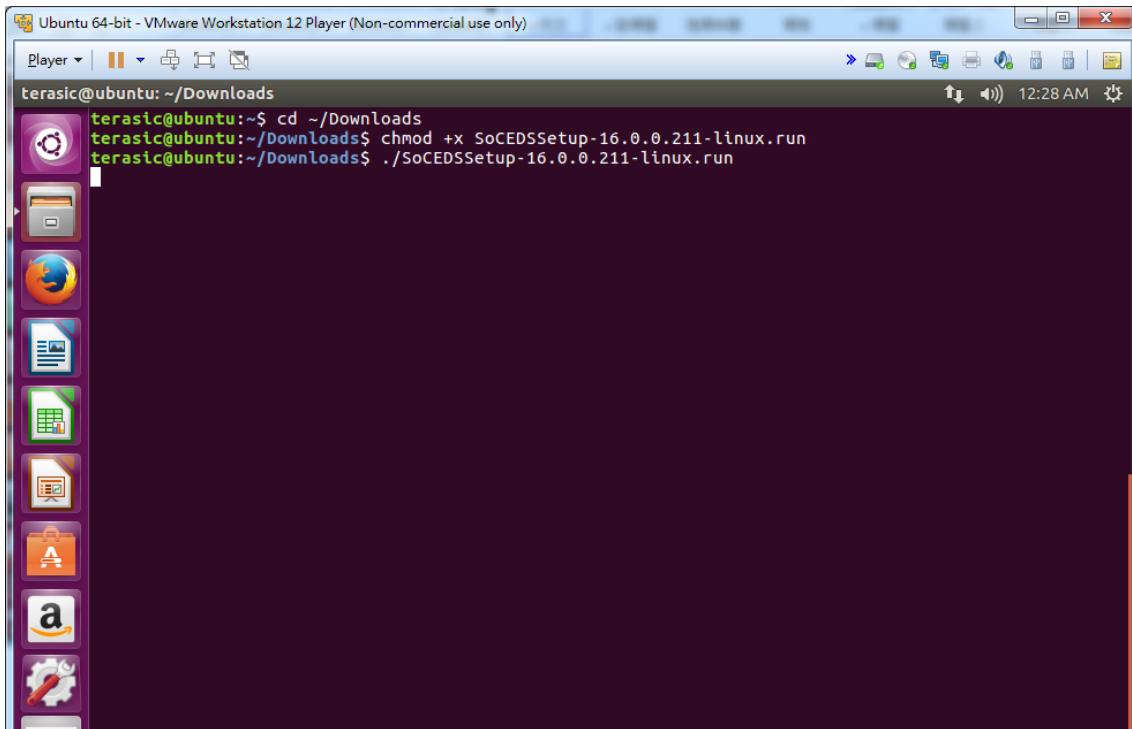
```
$ chmod +x SoCEDSSetup-16.0.0.211-linux.run
```

to add execution attribute to the installer.

Finally, type in:

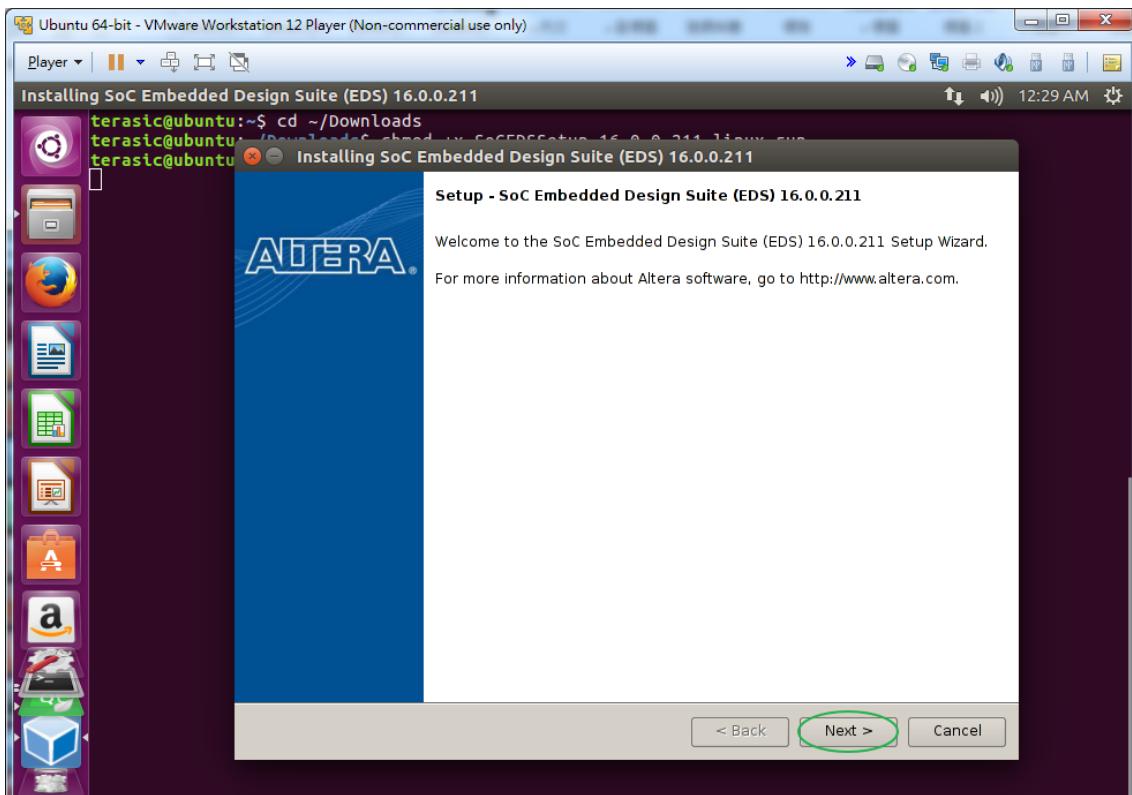
```
$ ./SoCEDSSetup-16.0.0.211-linux.run
```

to launch SoC EDS installer as shown in **Figure 8-3**.



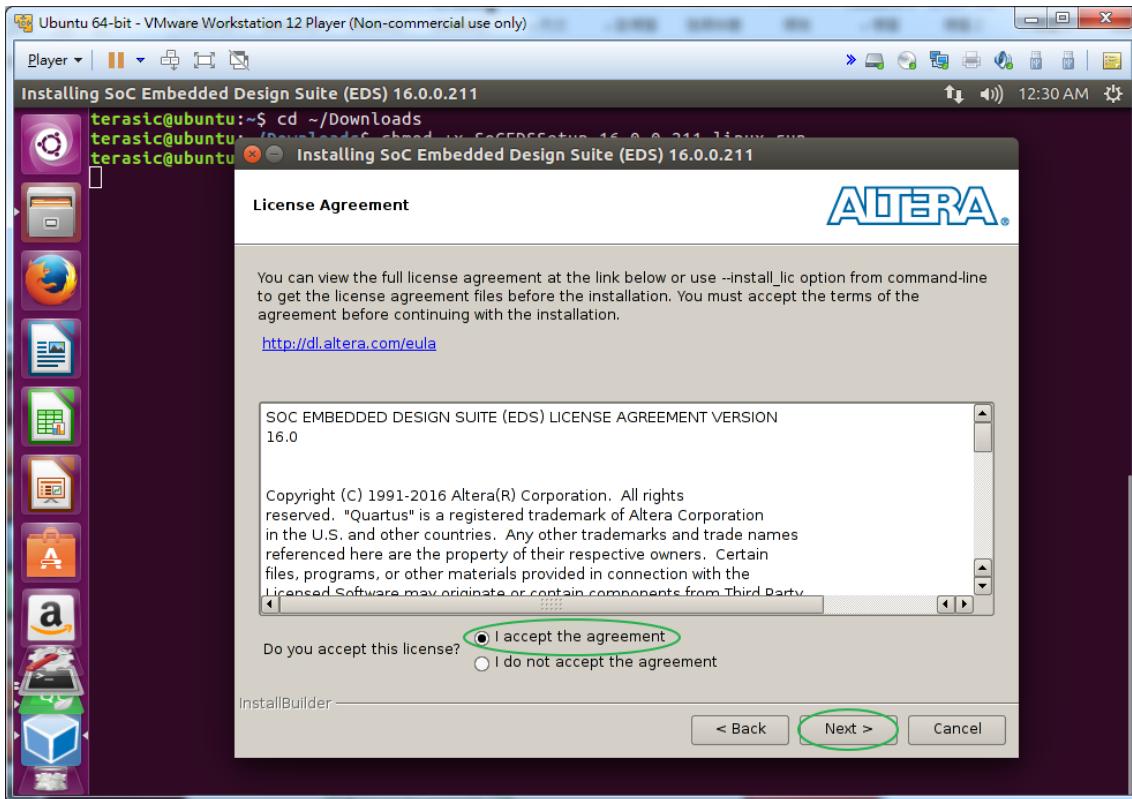
**Figure 8-3 Launch SoC EDS Installer**

When the installer is launched, a **Welcome** dialog appears as shown in **Figure 8-4**. Click "Next >" to go to the next step.



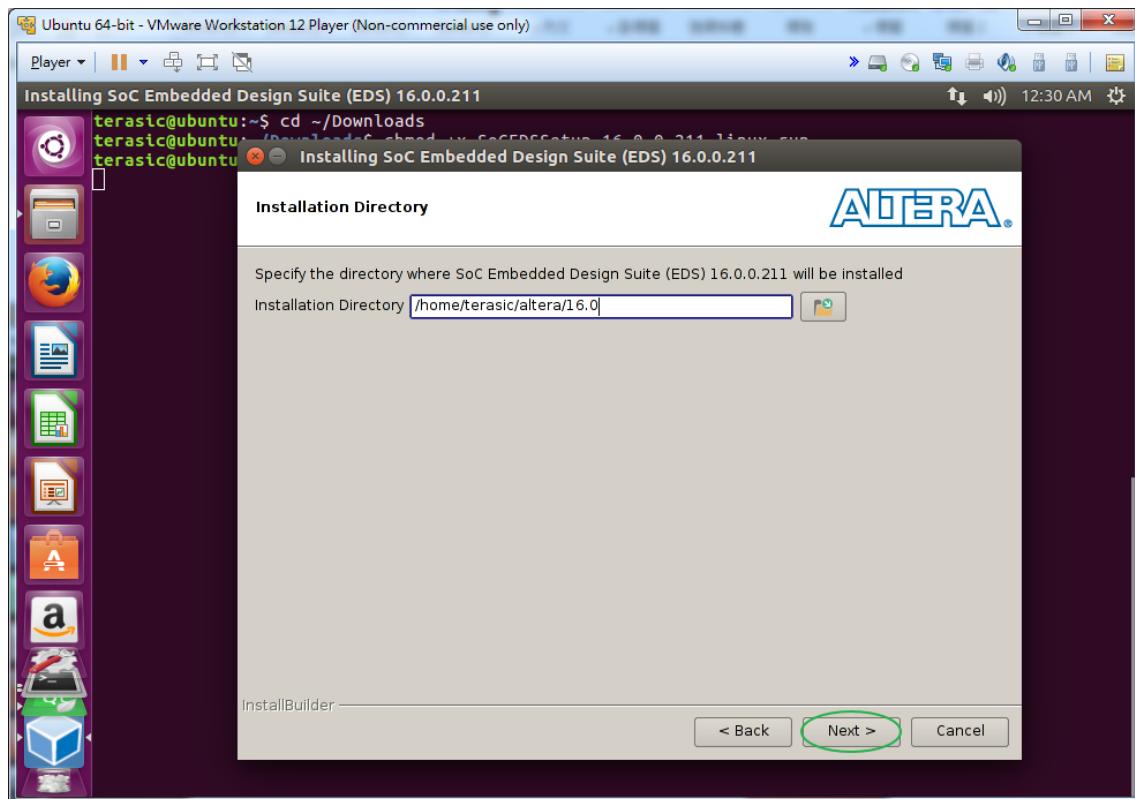
**Figure 8-4 Welcome Dialog of SoC EDS Installer**

When the **License Agreement** dialog appears as shown in **Figure 8-5**, please select the "I accept the agreement" radio button if you agree with the license and click "Next >" to go to the next step.



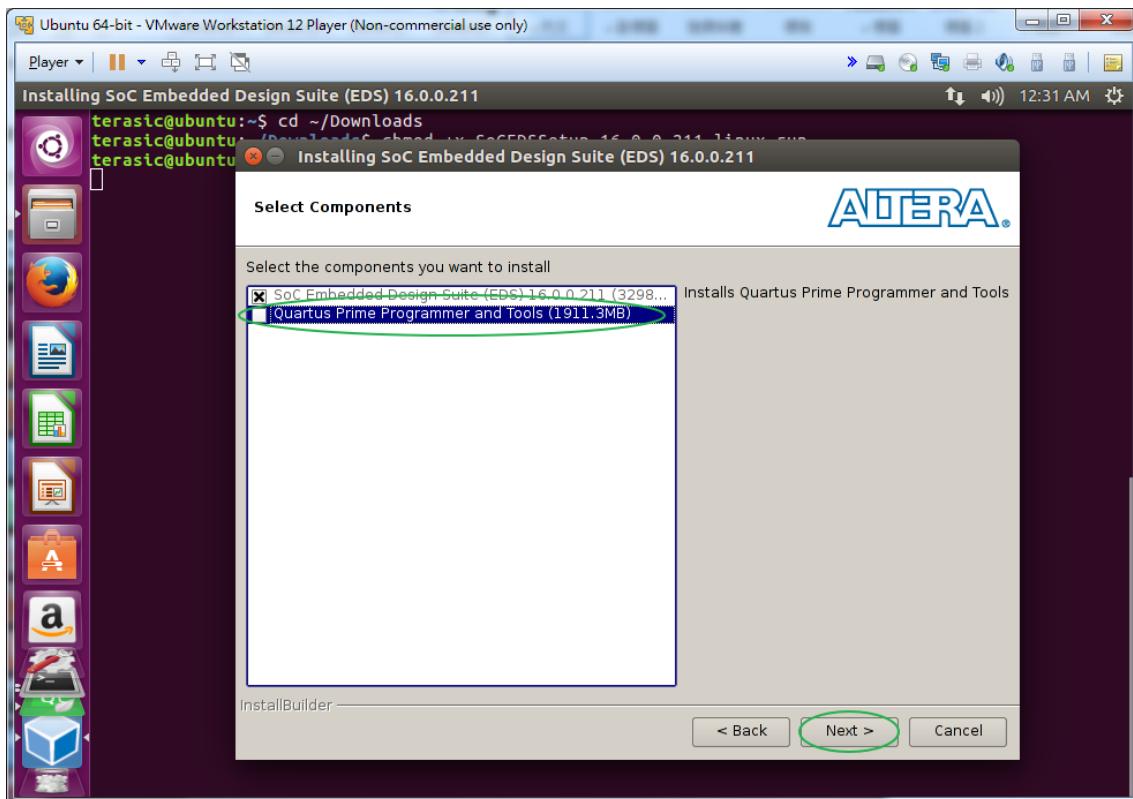
**Figure 8-5 License Agreement Dialog of SoC EDS Installer**

When the **Installation Directory** dialog appears as shown in **Figure 8-6**, we strongly recommend you keep the default directory and click "Next >" to proceed.



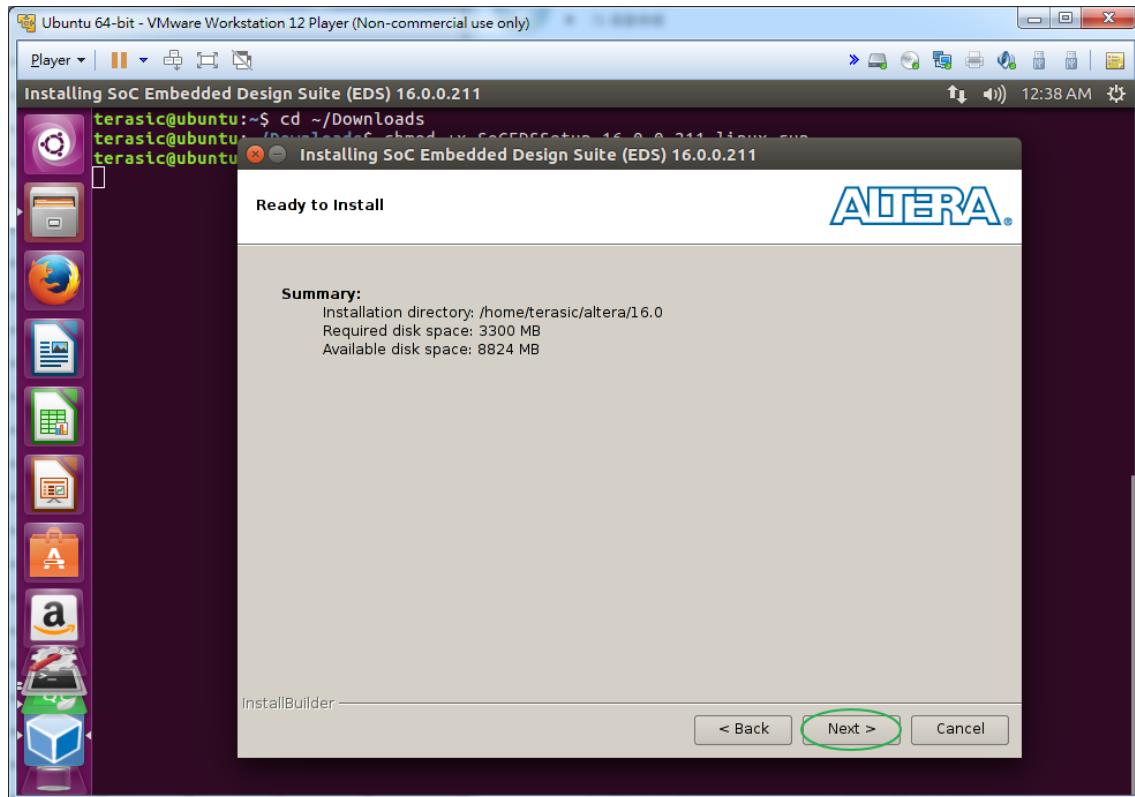
**Figure 8-6 Installation Directory Dialog of SoC EDS Installer**

When the **Select Components** dialog appears as shown in **Figure 8-7**, please **uncheck** the "Quartus Prime Programmer and Tools (1911.3MB)" checkbox and click "Next >" to go to the next step.



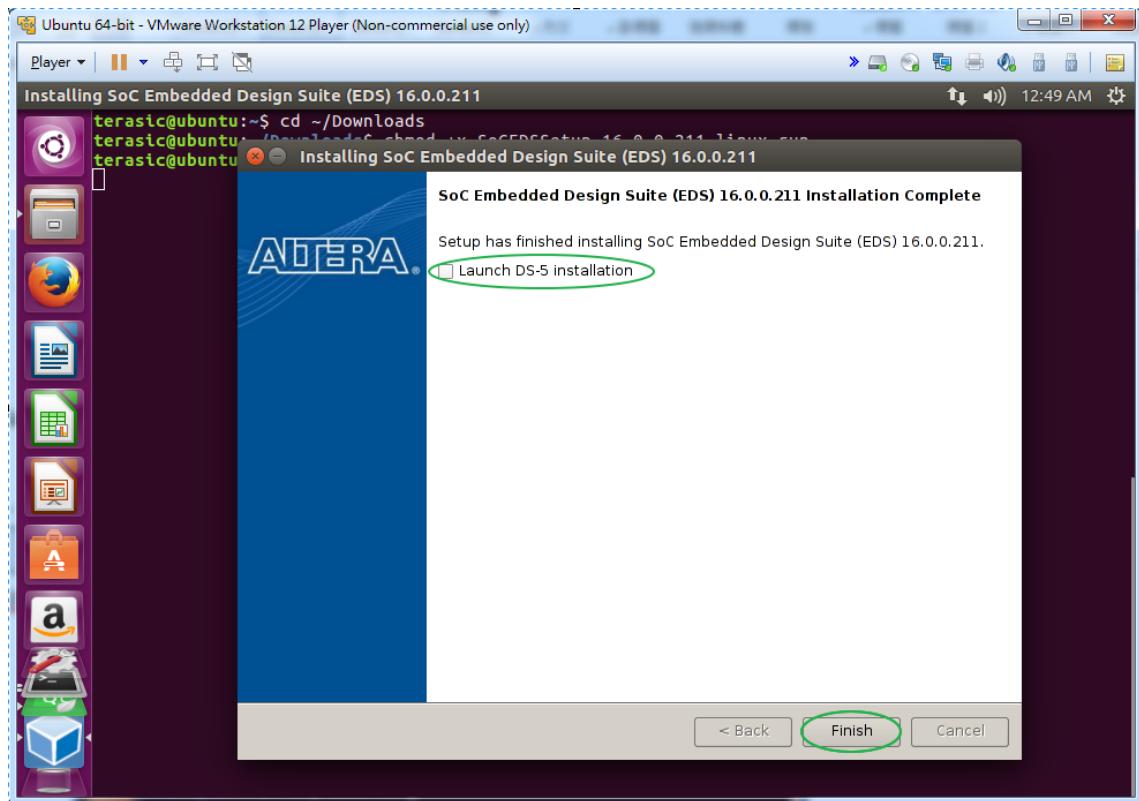
**Figure 8-7 Select Components Dialog of SoC EDS Installer**

When the **Ready to Install** dialog appears as shown in **Figure 8-8**, please click "Next >" to go to the next step.



**Figure 8-8 Ready to Install Dialog of SoC EDS Installer**

When the installation is complete, please uncheck the "Launch DS-5 Installation" checkbox and click "Finish" as shown in **Figure 8-9** to finish.

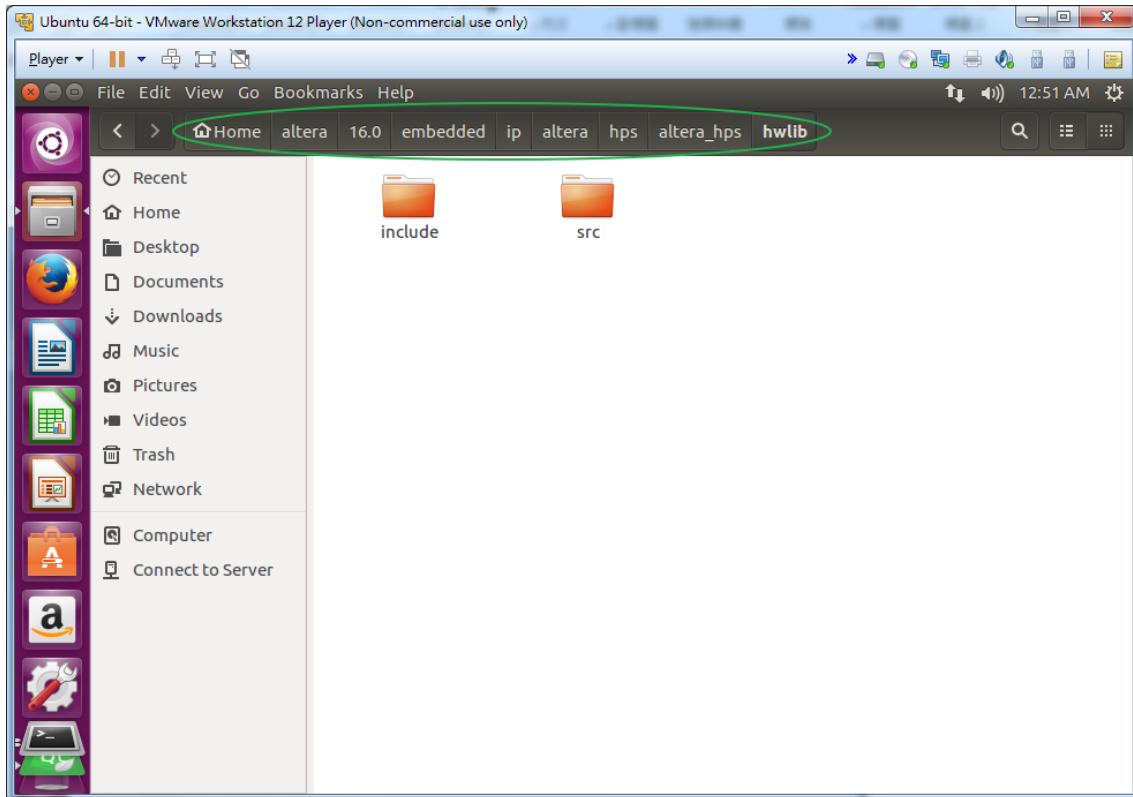


**Figure 8-9 Finish Dialog of SoC EDS Installer**

The source code of **hwlib** library can be found at the below location as shown in **Figure 8-10**.

```
/home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hwlib
```

(Note, in the path string, the "terasic" should be replaced with your linux user name.)



**Figure 8-10 Directory Location of Altera 'hwlib' Library Source Code**

## 8.2 Copy Control Panel Qt Project

The Control Panel Qt project is located on the System CD:

```
\Demonstrations\SoC_FPGA\ControlPanel\ControlPanel_QT
```

Here we copy the project from Windows host by using VMware Workstation Player's **Shared Folder** features, described in Chapter 2.6.

Copy the prebuilt Qt library compressed file from the System CD to the shared folder on Windows host. Then, launch the "Ubuntu 64-bit" virtual machine. After logging in Linux x64, launch a terminal (CTRL+ALT+T) and type the following command to go to the shared folder on Linux:

## ■ Copy Control Panel Qt Project to Shared Folder

Copy the Control Panel Qt project folder "ControlPanel\_QT" from the System CD to the shared folder on Windows host. Then, launch a terminal (CTRL+ALT+T) and type the following command to go to the shared folder on Linux:

```
$ cd /mnt/hgfs/shared
```

to go to the shared directory on Linux.

Now type in

```
$ ls
```

To see the "ControlPanel\_QT" directory of Control Panel Qt Project, as shown in **Figure 8-11**.

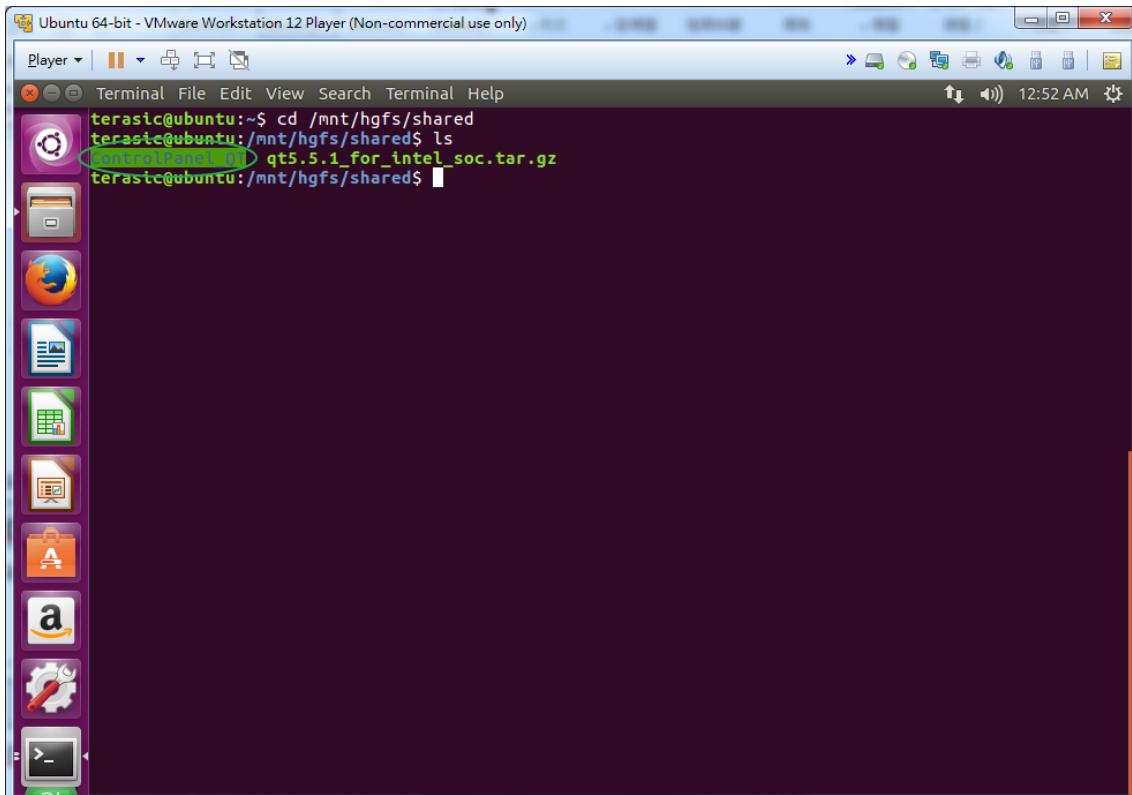
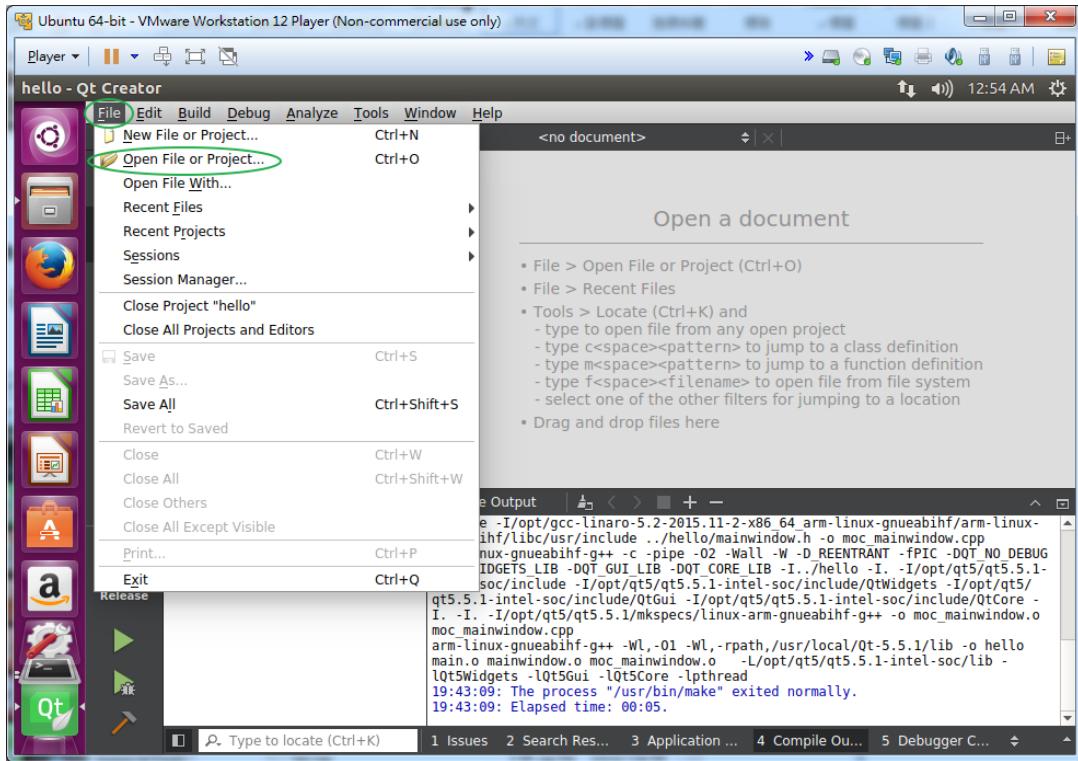


Figure 8-11 'ControlPanel\_QT' Directory of Control Panel Qt Project

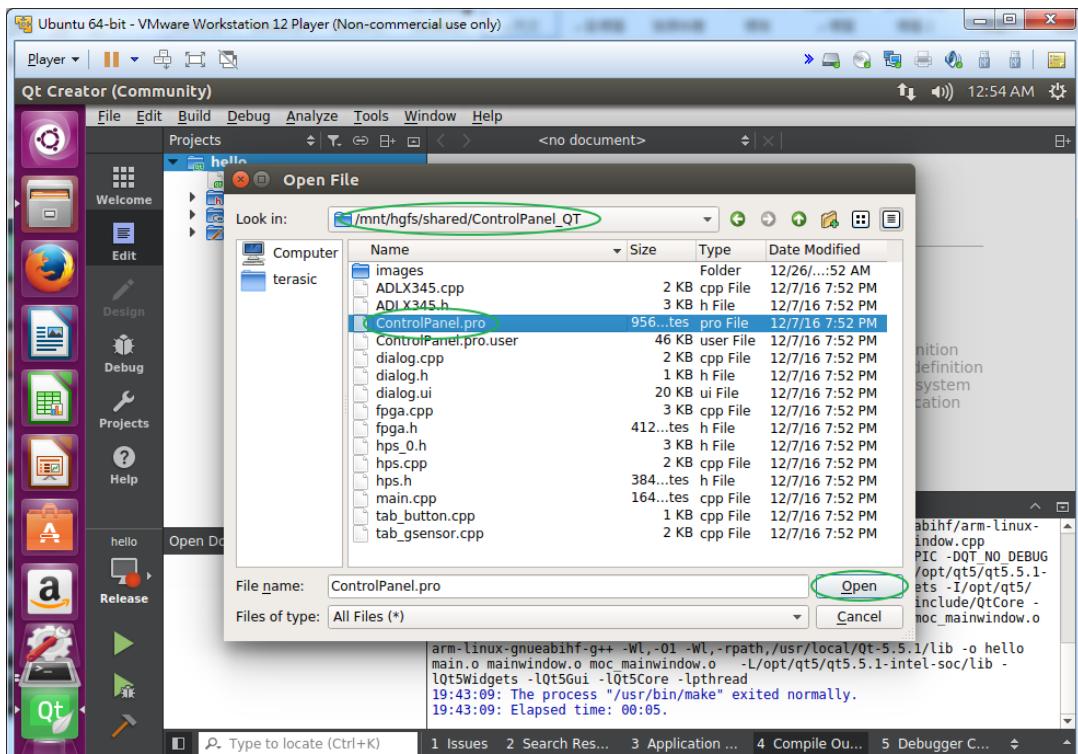
## 8.3 Build Control Panel Qt Project

Now, we can launch the Qt Creator, and select the menu item "Files→Open File or Project..." as shown in **Figure 8-12**.



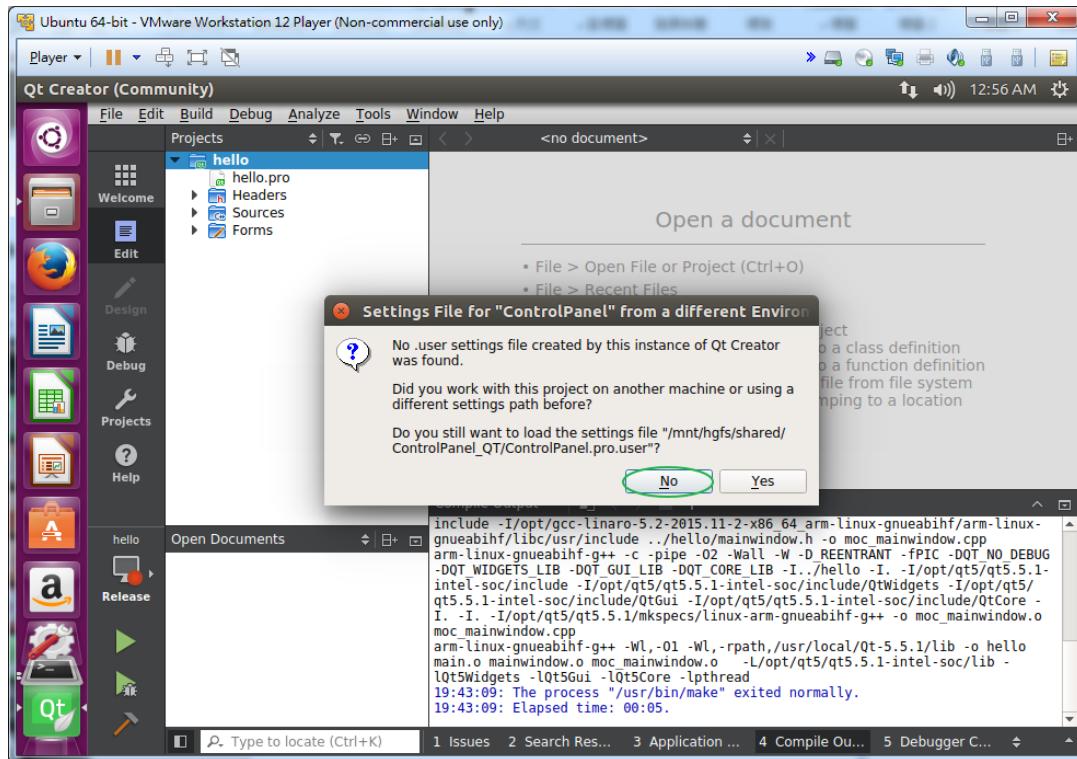
**Figure 8-12 Launch Open Project Dialog**

In the **Open File** dialog, go to the directory "/mnt/hgfs/shared/ControlPanel\_QT", select "ControlPanel.pro", and click "Open" as shown in **Figure 8-13**.



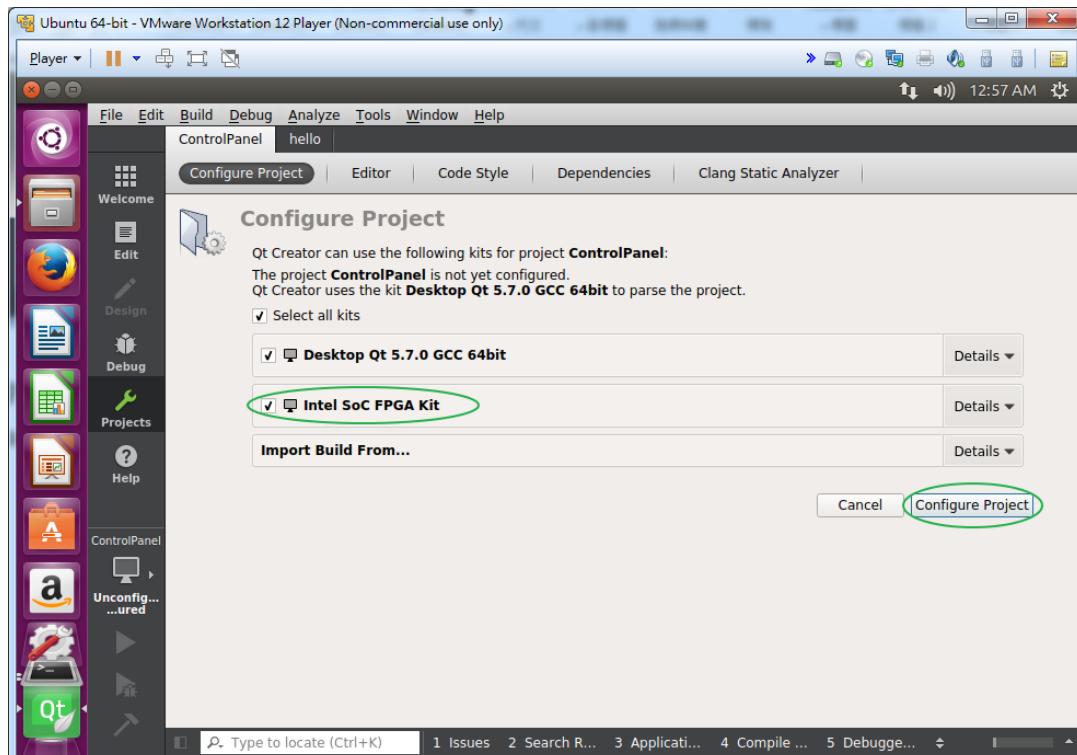
**Figure 8-13 Select Control Panel Project – ControlPanel.pro**

If the **Settings File** dialog appears as shown in **Figure 8-14**, please click "No" to proceed.



**Figure 8-14 Query .user Setting**

If Configure Project appears as shown in **Figure 8-15**, please check "Intel SoC FPGA Kit" and click "Configure Project".



**Figure 8-15 Configure Project**

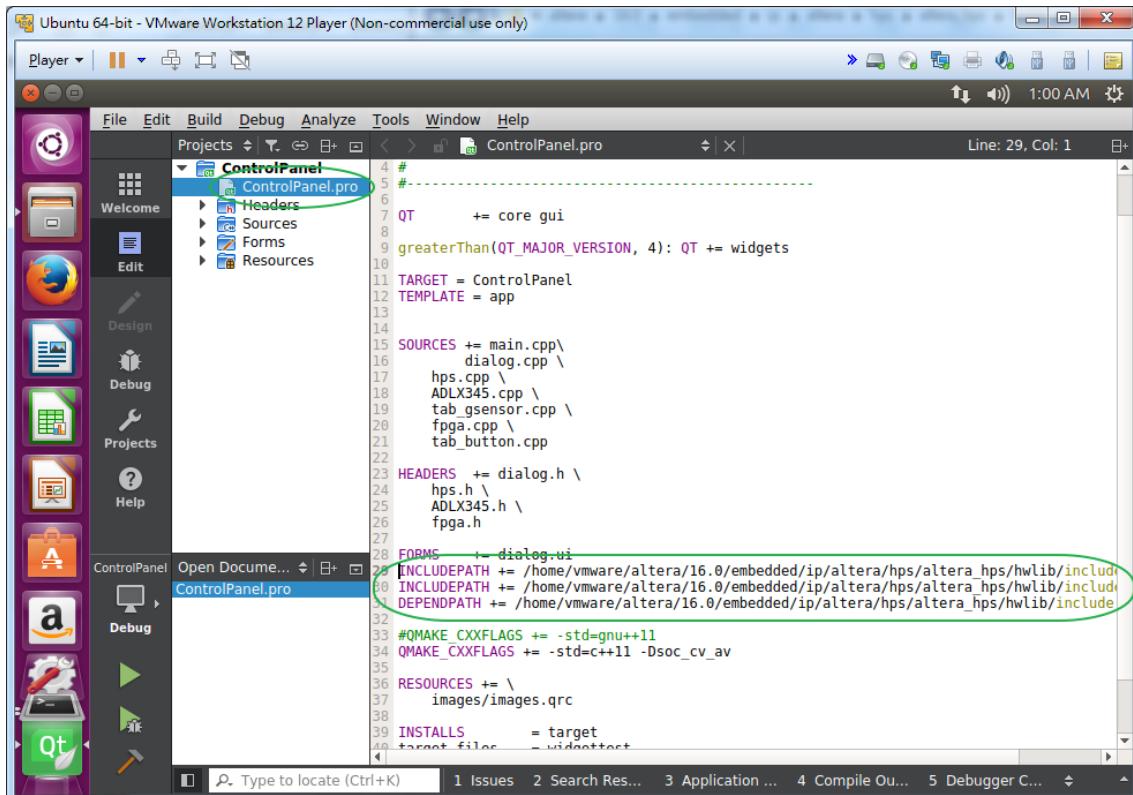
Now, to check if the include path is correct, please follow the instructions below:

First click on "Edit" icon on the left and double click "ControlPanel.pro" as shown in **Figure 8-16**, please make sure the **INCLUDEPATH** includes the correct path:

```
/home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hplib/include  
/home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hplib/include/soc_cv_av
```

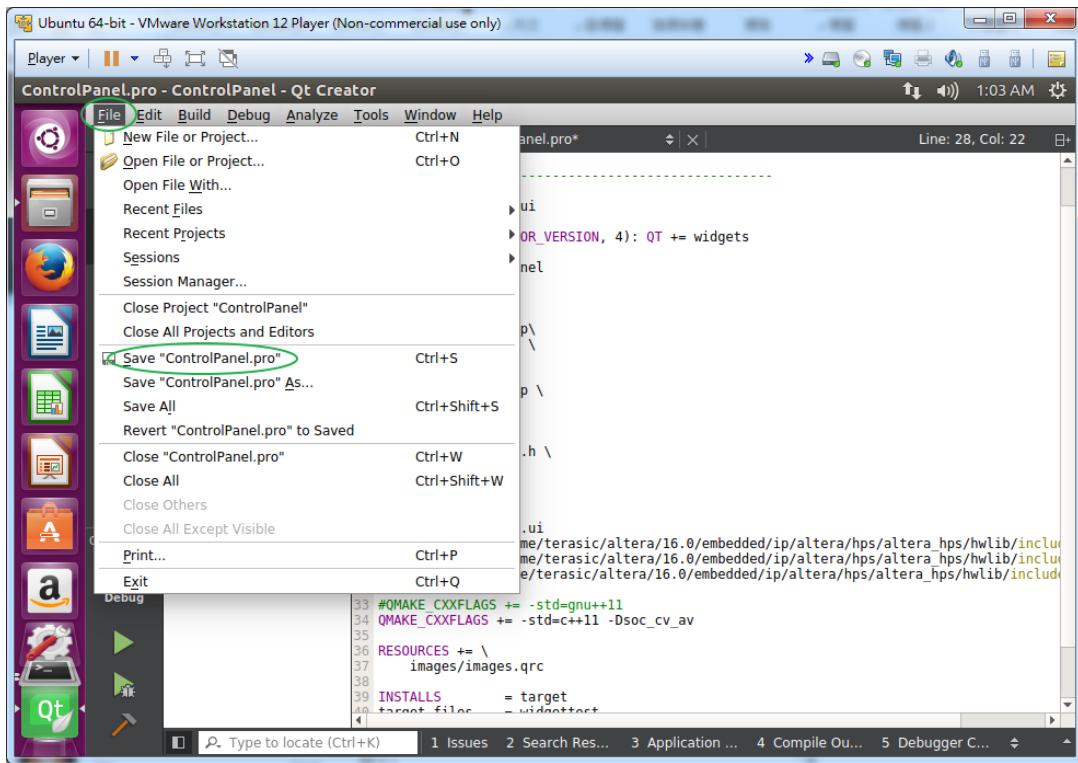
(Note, in the path string, the "terasic" should be replaced with your linux user name.)

If the file "ControlPanel.pro" is modified, please correct the path and select the menu item "File→Save "ControlPanel.pro" as shown in **Figure 8-17**.



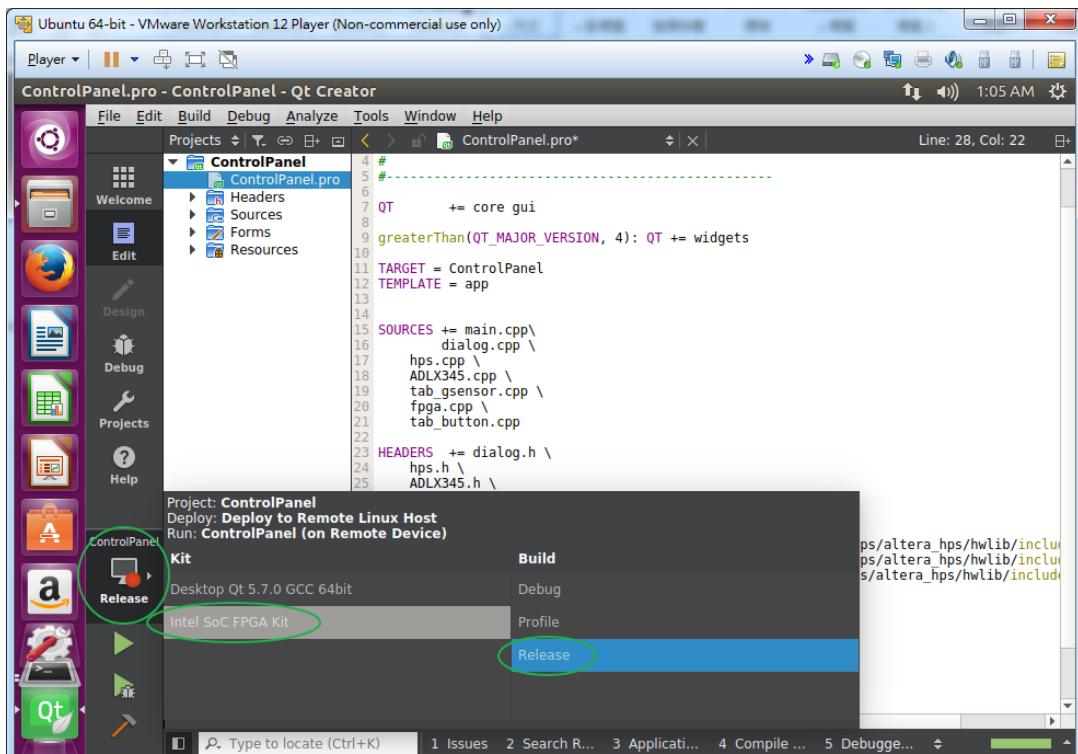
```
Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)  
File Edit Build Debug Analyze Tools Window Help  
Projects ControlPanel ControlPanel.pro  
Welcome Edit Design Debug Projects Help  
ControlPanel Open Document... ControlPanel.pro  
Qt  
  
4 #  
5 #  
6 QT += core gui  
7 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
8  
9 TARGET = ControlPanel  
10 TEMPLATE = app  
11  
12 SOURCES += main.cpp \  
13 dialog.cpp \  
14 hps.cpp \  
15 ADXL345.cpp \  
16 tab_gsensor.cpp \  
17 fpga.cpp \  
18 tab_button.cpp  
19  
20 FORMS += dialog.ui  
21 HEADERS += dialog.h \  
22 hps.h \  
23 ADXL345.h \  
24 fpga.h  
25  
26  
27  
28 DEPENDPATH += /home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hplib/include  
29 INCLUDEPATH += /home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hplib/include  
30 DEPENDPATH += /home/terasic/altera/16.0/embedded/ip/altera/hps/altera_hps/hplib/include  
31  
32  
33 #QMAKE_CXXFLAGS += -std=gnu++11  
34 QMAKE_CXXFLAGS += -std=c++11 -Dsoc_cv_av  
35  
36 RESOURCES += \  
37 images/images.qrc  
38  
39 INSTALLS += target  
40 target_file = target
```

**Figure 8-16 Check Include Path in ControlPanel.pro**



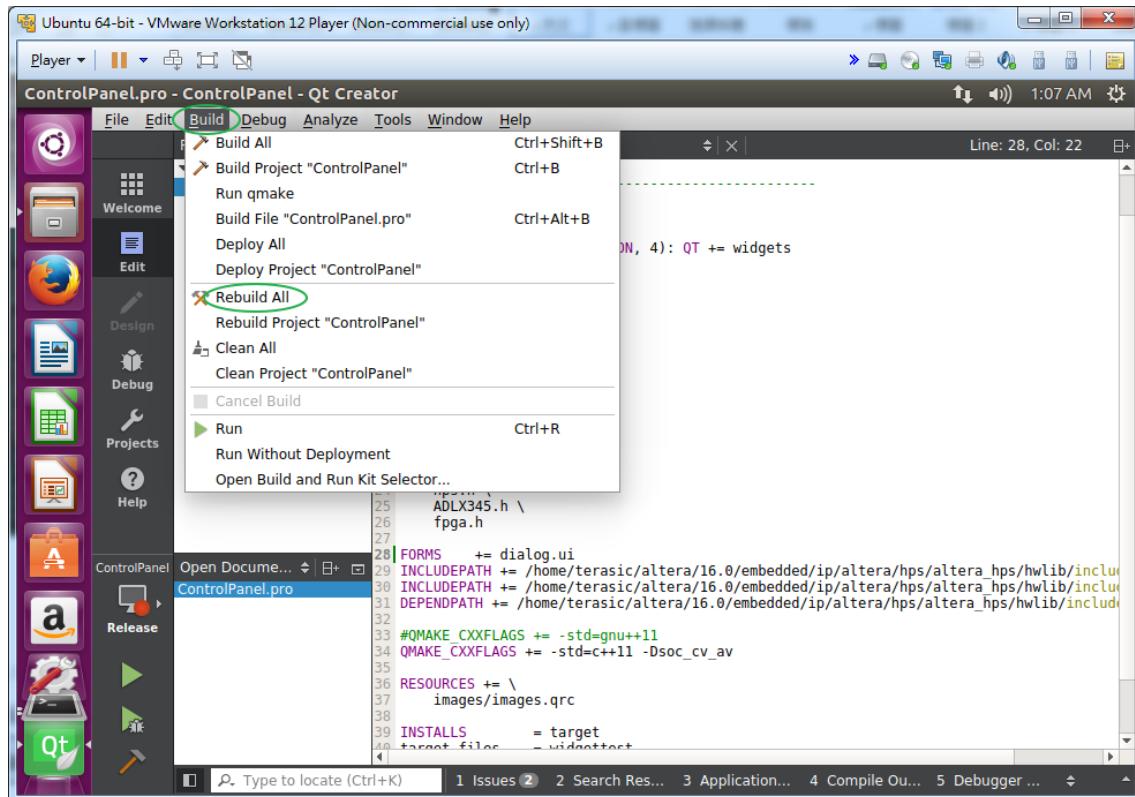
**Figure 8-17 Save ControlPanel.pro**

Click on the "Release" icon, and select "Intel Soc FPGA Kit" and "Release" Build as shown in **Figure 8-18**.



**Figure 8-18 Setting Release Kit and Build**

Then, select menu item "Build→Rebuild All", as shown in **Figure 8-19**, to build the Control Panel Project.



**Figure 8-19 Build Control Panel**

If the build is successful, the ControlPanel execution file is generated in the below folder as shown in **Figure 8-20**.

```
/mnt/hgfs/shared/build-ControlPanel-Intel_SoC_FPGA_Kit-Release
```

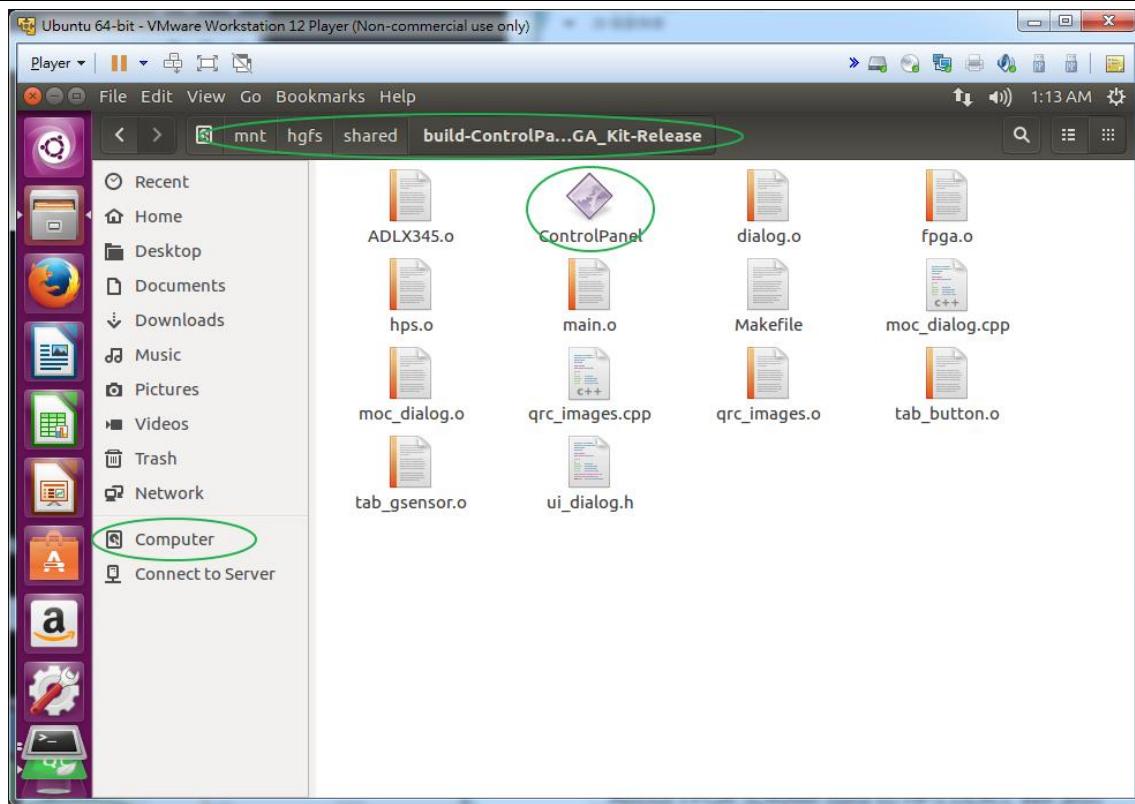


Figure 8-20 Generated Control Panel Execution File

## 8.4 Execute Control Panel Program

To run the "ControlPanel" execution file on the DE10-Nano FPGA development board, we need to copy it to the Linux on Intel SoC FPGA Board. This is similar to what we have done in Section 6.3 "Execute Hello Program" where we use Linux "scp" command to remote copy.

After "ControlPanel" execution file has been remotely copied to the "/home/root" directory of Linux running on Intel SoC FPGA Board, double click "ControlPanel" icon and click "Execute" button to launch the Control Panel in the LXDE Desktop of Intel SoC FPGA board. If you can see the ControlPanel windows as shown in **Figure 8-21**, it means you had successfully built the Control Panel Qt Project.



Figure 8-21 Screenshot of Control Panel

## 8.5 More on the Control Panel QT Project

### ■ Control FPGA LED

Control Panel controls the FPGA Qsys components through the memory-mapped method. The device driver "/dev/mem" is used to access the physical address space which is mapped to Qsys address space.

Below shows how to turn on the 8 LEDs, which is controlled by Qsys PIO Controller, whose address is 0x10040 in the Qsys system. Based on "/dev/mem" device driver and "mmap" function, we can calculate the LED address **led\_base**. With the address, we can directly write the value of the address to control the PIO Controller. Constant ALT\_LWFPGASLVS\_OFST is defined "socal/hsp.h"

```
#include "socal/hps.h"

#define HW_REGS_BASE ( ALT_STM_OFST )
#define HW_REGS_SPAN ( 0x04000000 )
```

```

#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )

// Controller Base Address in QSYS
#define FPGA_LED_PIO_BASE 0x10000

int file;
void *virtual_base;
uint8_t *led_base;

file = open( "/dev/mem", ( O_RDWR | O_SYNC ) );
virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE ),
MAP_SHARED, file, HW_REGS_BASE );
led_base= (uint8_t *)virtual_base + ( ( unsigned long )( ALT_LWFPGASLVS_OFST +
FPGA_LED_PIO_BASE ) & ( unsigned long)( HW_REGS_MASK ) );

*(uint32_t *)m_led_base = 0xFF; // turn on 8 led

```

## ■ Control HPS LED

The way to control HPS LEDs is similar to the way to control FPGA LED. The register files of both PIO controllers are different. Macro `alt_setbits_word` is used to set value to the specific address. The HPS LED is controlled by GPIO1 GPIO Controller.

Constant `ALT_GPIO1_SWPORTA_DDR_ADDR` is used to define the direction register address of GPIO1 Controller, and constant `USER_IO_DIR` is used to define the direction pin bits-mask associated to the HPS LED.

Constant `ALT_GPIO1_SWPORTA_DR_ADDR` defines the data register address of GPIO1 Controller, and constant `BIT_LED` is used to define the pin bits-mask associated to the HPS LED. Both `ALT_GPIO1_SWPORTA_DDR_ADDR` and `ALT_GPIO1_SWPORTA_DR_ADDR` are defined in "socal/hps.h". Macro `alt_setbits_word` is defined in "socal/socal.h"

```

#include "socal/socal.h"
#include "socal/hps.h"

#define HW_REGS_BASE ( ALT_STM_OFST )
#define HW_REGS_SPAN ( 0x04000000 )

```

```

#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )

#define USER_IO_DIR      (0x01000000)
#define BIT_LED          (0x01000000)

int file
void *virtual_base;
uint8_t *led_base;

file = open( "/dev/mem", ( O_RDWR | O_SYNC ) );
virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE ),
MAP_SHARED, file, HW_REGS_BASE );
// configure LED as output pin
alt_setbits_word((void *) ( (char *)virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DDR_ADDR ) & ( uint32_t )( HW_REGS_MASK ) ) ),
USER_IO_DIR );
// turn on LED
alt_setbits_word((void *) ( (char *)virtual_base +
( ( uint32_t )( ALT_GPIO1_SWPORTA_DR_ADDR ) & ( uint32_t )( HW_REGS_MASK ) ) ),
BIT_LED );

```

## ■ FPGA Class

All of FPGA control functions are encapsulated in the **FPGA** class and it is implemented in fpga.c and fpga.h. Below shows the public functions of this class:

```

bool LedSet(int mask);
bool KeyRead(uint32_t *mask);
bool SwitchRead(uint32_t *mask);

```

## ■ HPS Class

All of HPS control functions are encapsulated in the **HPS** class and it is implemented in hps.cpp and hps.h. Below shows the public functions of this class:

```

bool LedSet(bool bOn);
bool IsButtonPressed();
bool GsensorQuery(int16_t *X, int16_t *Y, int16_t *Z);

```