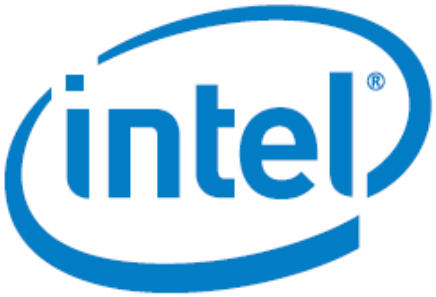


Intel® Galileo Development Kit for IoT

Tutorial Guide

Table of Contents

Document / Revision History

Date	Change Description	Version
20-Feb-2014	First release at MWC	1.0

Reference Documents

Document Name	Description
Intel® Galileo DevKit QSG	Intel® Galileo Development Kit for IoT Quick Start Guide.
Galileo Datasheet (329681-003US)	Detail of Galileo board and specifications.

Terminology / Definitions

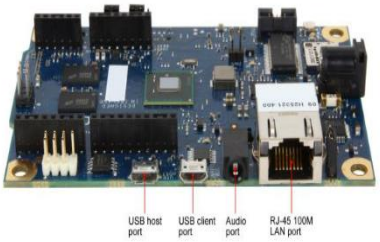
Term	Change Description
DevKit	Intel® Galileo Development Kit for IoT
ICC	Intel® C/C++ Compiler Toolchain
Host	The machine running the DevKit Live USB image
Target	Intel® Galileo platform

1 Introduction

The DevKit is an out-of-the box evaluation and development platform based on the Intel® Galileo board. This manual describes the technical details and the process of booting the DevKit Live image and the guide for application developers.

1.1 Component list

Disclaimer: Please note that pictures here are just for reference, the content and parts may vary.

No.	Item	Description	Qty
1.	Quick Start Guide	Intel® Galileo DevKit Quick Guide - Paper copy	1
2.		Intel® Galileo Development board. 400MHz 32-bit Intel® Pentium instruction set architecture (ISA)-compatible processor A full sized mini-PCI Express* slot, 100Mb Ethernet port, Micro-SD slot, RS-232 serial port, USB Host port, USB Client port, and 8MByte NOR flash come standard on the board.	1
3.		Micro SD card – 8GB	1
4.		USB Flash Disk – 16GB: Pre-loaded with live image for the Host	1
5		Power Supply – 5VDC, 3.0A	1
6		Wi-Fi card with Extender	1

7		Antennas	2
9		Serial Cable (3.5mm Stereo jack to DB9)	1
9		USB to 9pin-D cable	1
10		USB OTG adapter	1
11		LCD Module For Arduino 20 X 4	1
12		Sony eye webcam to provide video and audio feed via USB	1

1.2 DevKit Live USB Image

The login details for the Live USB image are:

Root user:

Username: root

Password: root

Ordinary user:

Username: user

Password: devkit

The login details for your Galileo target are:

Root user:

Username: root

Password: There is no password

2 DevKit Setup

Establish the target and host connections as shown it the figure below:

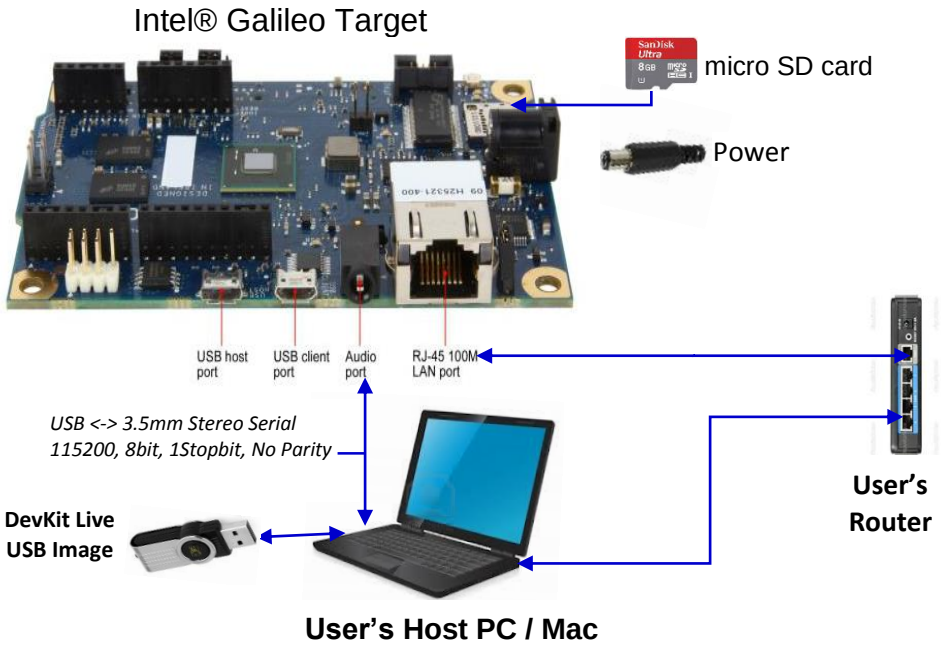


Figure 1: DevKit Development System and Target connections

2.1 Connections:

1. Insert the micro SD card supplied with the DevKit in the SD Card slot of Galileo.
2. Connect Ethernet cable between Galileo board and your internet router.
3. Plug-in serial cable between your Host PC (USB end) and 3.5mm stereo jack male pin in to Galileo board audio type serial connector close to Ethernet port.
4. Connect power cable to Galileo board and switch ON the power.

2.2 Booting DevKit Live USB on PC:

NOTE: The BIOS configuration are all PC specific, please refer to your PC manufacturer guide for details on specific BIOS settings.

1. Insert the provided DevKit Live USB key into the USB port of the Host PC.
2. Power on the computer and enter into BIOS menu.
3. Ensure that the legacy BIOS is enabled.
4. Select – **USB key as the first boot device** in the boot order.
5. Save your settings and exit the BIOS menu, reboot the Host PC.

6. Your Host PC will boot from the USB key and you will be presented with the following menu. For most modern computers the first (default) option should be sufficient (kernel 3.10). If you have older hardware or experience boot failures, please boot the 3.2 kernel (486 or 686-pae depending on your hardware).
7. Select the option corresponding to your CPU architecture and **Press Enter**.

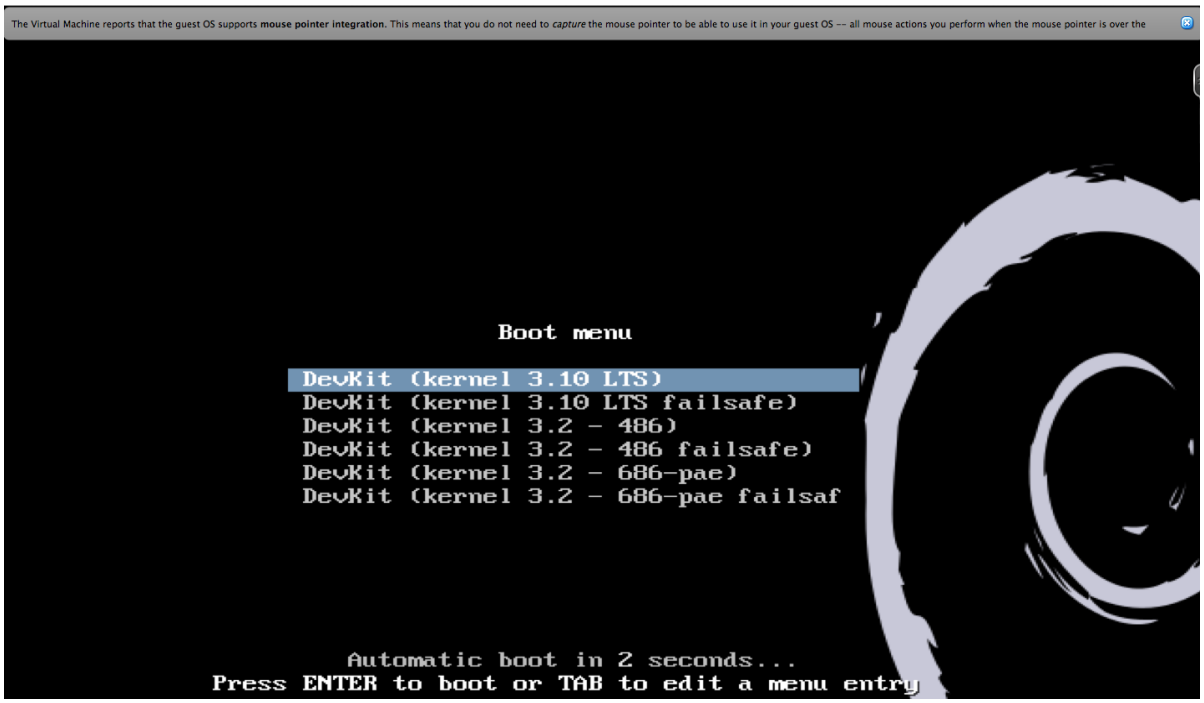


Figure 2: Menu on Boot from Live USB Key

2.3 Booting DevKit Live USB on Mac:

NOTE: Because Mac UEFI firmware does not implement the 'Legacy BIOS' extension. The DevKit live image needs to be run in a virtualized environment.

1. Install 'VirtualBox' and 'Oracle VM VirtualBox Extension Pack' on your Mac OS X machine by following the instructions on the Virtual Box website: https://www.virtualbox.org/wiki/Downloads
2. Insert the USB key into a USB slot on your Mac
3. Open the Terminal application and switch to the root user using the '**su'** command and enter the root password.

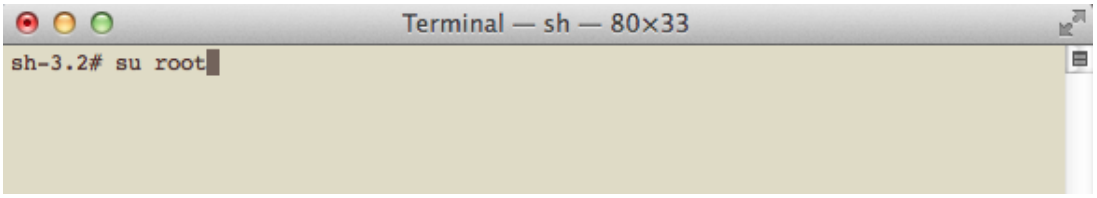


Figure 3: Super user terminal on Mac

4. Use the '**diskutil list'** command to find out the device node of your USB key and note it down (e.g '**/dev/disk4**' on this screenshot)

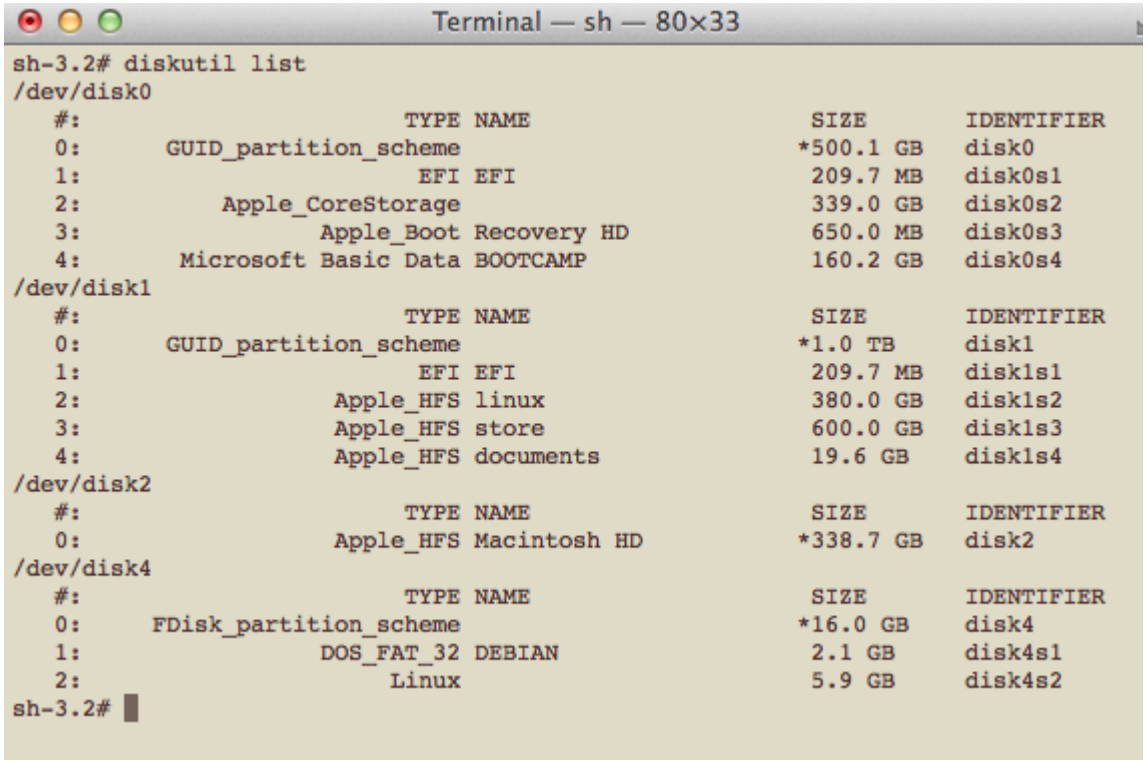


Figure 4: Device list on Mac

5. Use the '**diskutil unmountdisk'** command along with the device node captured in the previous step to unmount all partitions found on the USB key e.g.

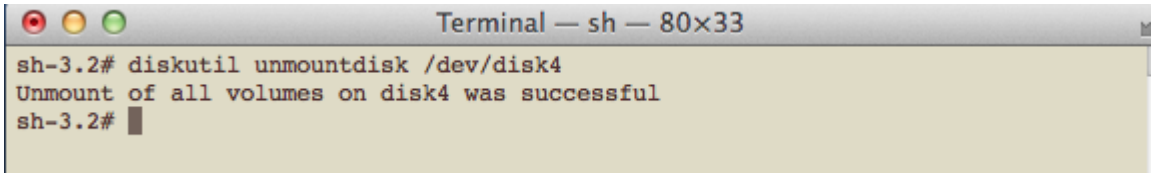


Figure 5: Unmount disk on Mac

6. Use the '**chmod'** command along with the device node captured in the previous step to make the USB key device accessible by a non-root user

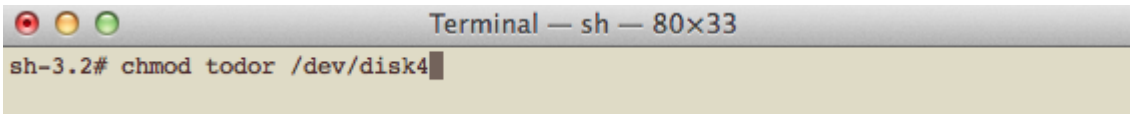


Figure 6: Make disk accessible on Mac

7. Create a VirtualBox virtual disk file corresponding to the DevKit USB key using the VBoxManage command as shown below. Please note you will need to change the path where you wish the file to be stored and use the device node captured in step 10 above

```
VBoxManage internalcommands createrawvmdk -filename /Users/todorminchev/Desktop/devkit.vmdk -
rawdisk /dev/disk4
```

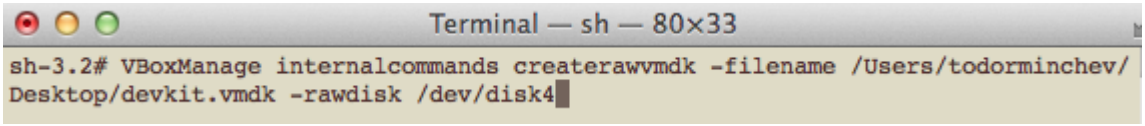


Figure 7: Create Virtual disk on Mac

NOTE : Back up your virtual disk file (e.g. devkit.vmdk) to a safe location. You might need to refer to it in the future in case that your virtual machine gets corrupted.

8. You are now ready to create your virtual machine. Use the 'diskutil unmountdisk' command to unmount the USB key again (please see step 5 above).

9. Open **VirtualBox -> Select ‘New’ -> Type ‘Linux’ -> Version ‘Debian’ -> Enter a name for your virtual machine -> Click ‘Continue’**

Figure 8: Virtual disk OS specifications on Mac

10. Assign memory to your virtual machine according to the amount of memory available on your host machine (recommended 2GB or more) -> Click ‘Continue’

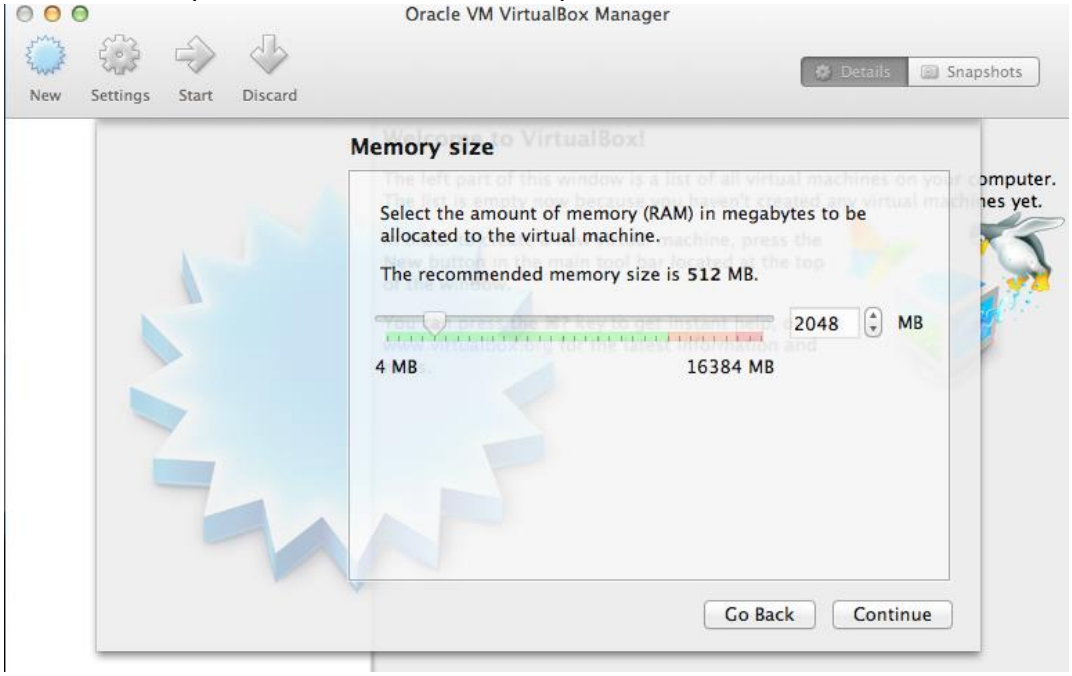


Figure 9: Virtual disk Memory specifications on Mac

11. Select **'Use an existing virtual hard drive file**' checkbox and assign the virtual machine disk file created in step 7 above to your virtual machine. -> Click '**Create'**

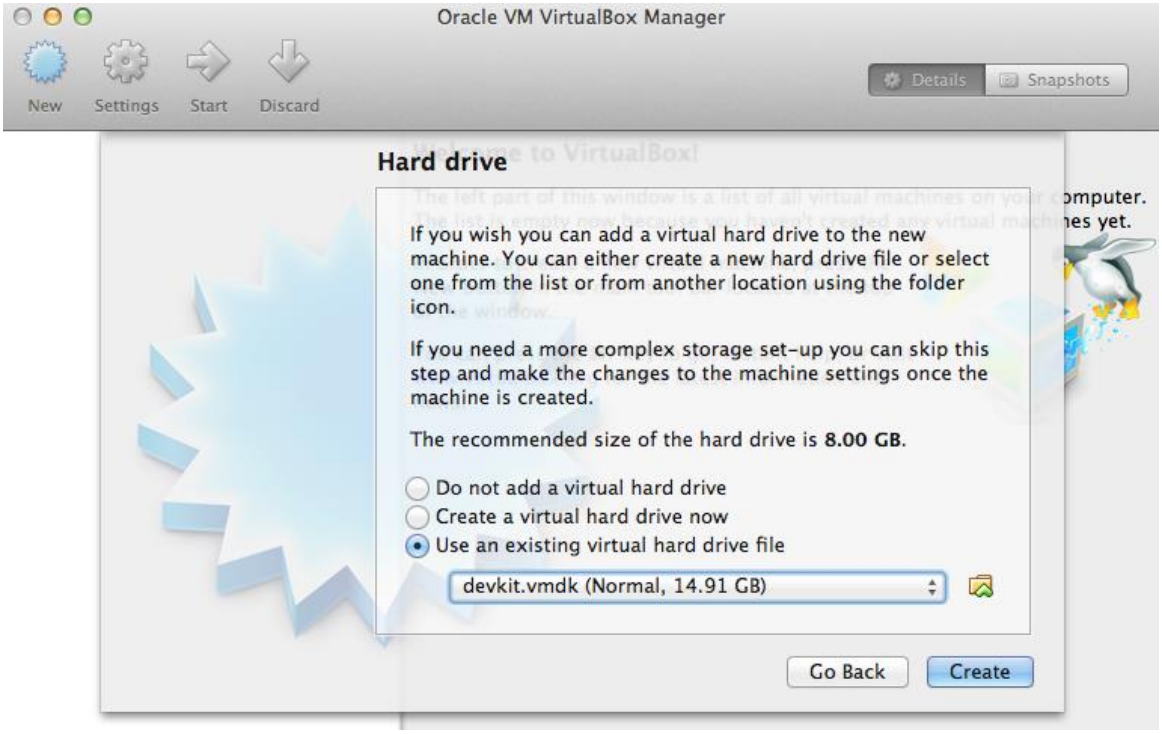


Figure 10: Virtual disk drive specifications on Mac

12. Create a filter for the USB -> Serial adapter by right clicking on your virtual machine name -> Select **'Settings'** -> Click the '**Ports'** button -> Select the '**USB' tab -> Click the plus button -> Select your USB adapter** from the list of available devices

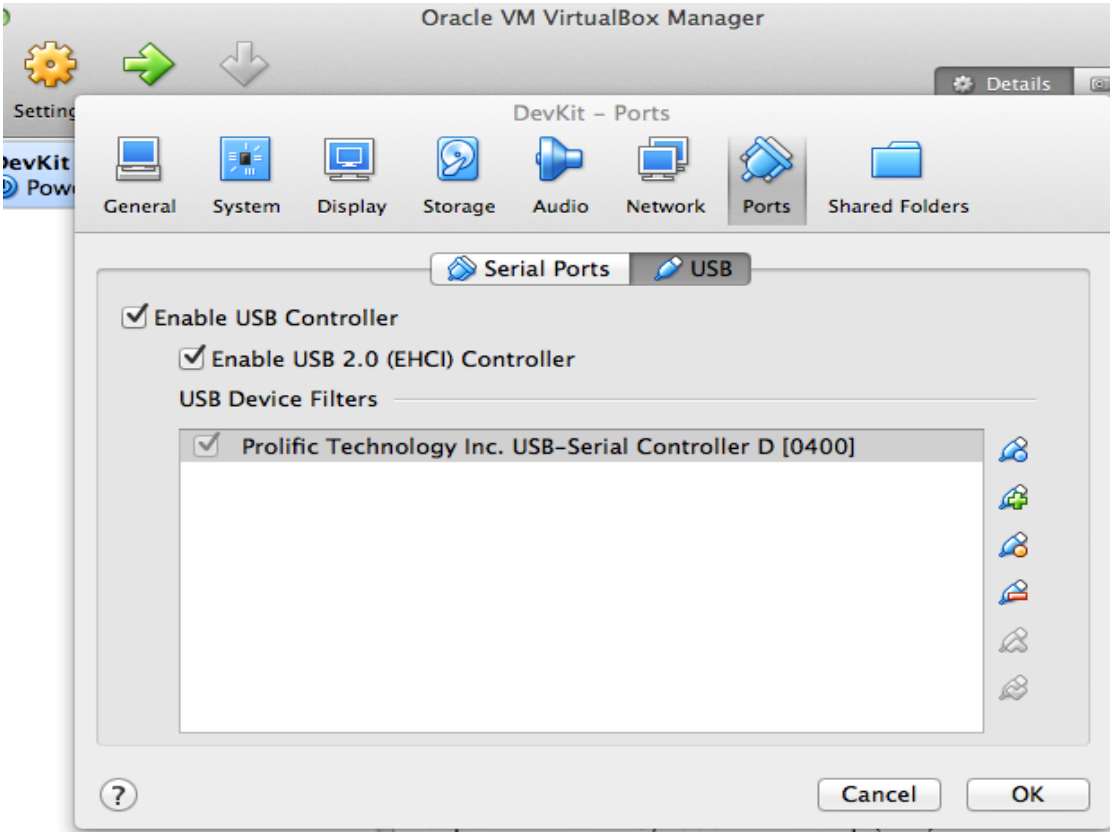


Figure 11: VirtualBox Serial-USB configuration on Mac

13. Double click your USB adapter and clear all fields except the **Vendor ID and Product ID** -> Set **Remote to 'Any' -> Click 'OK'** to close the **USB menu -> Click 'OK**' to close the setting menu.

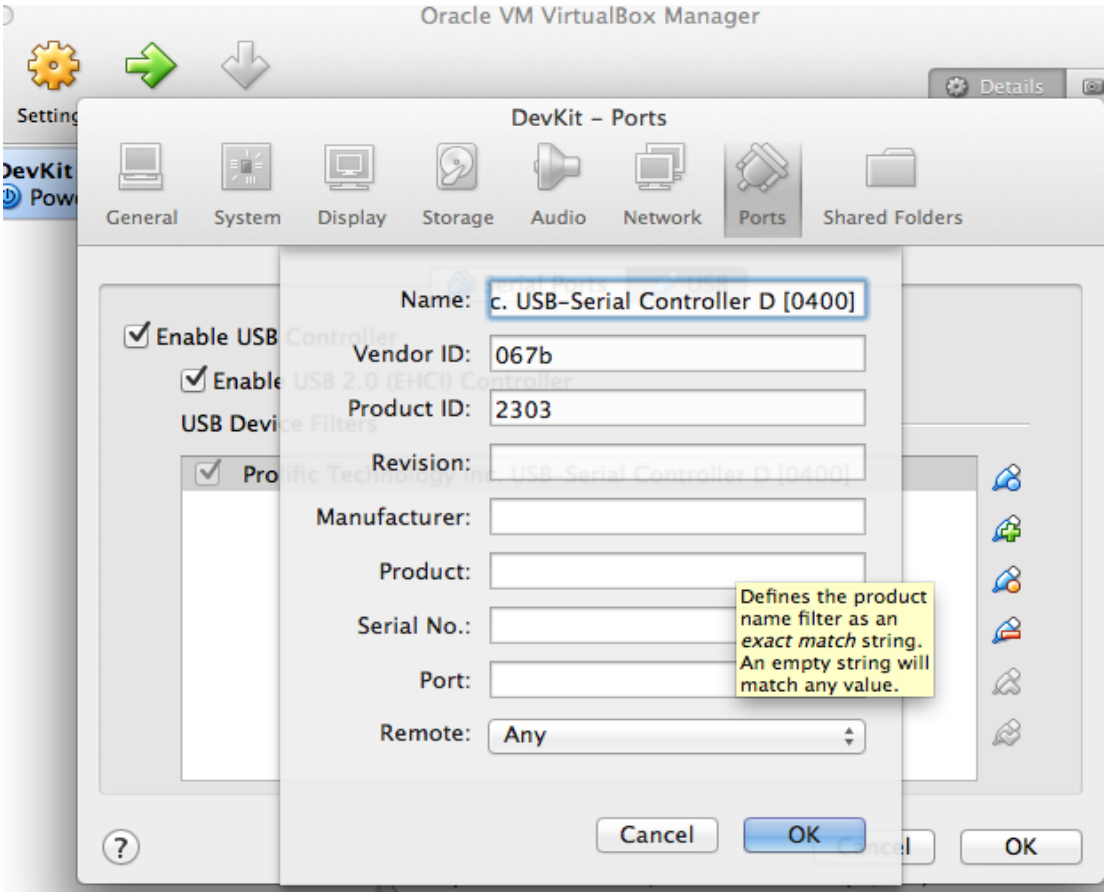


Figure 12: VirtualBox Serial-USB configuration on Mac

14. Your virtual machine is now ready! Click 'Start' to boot it for the first time.

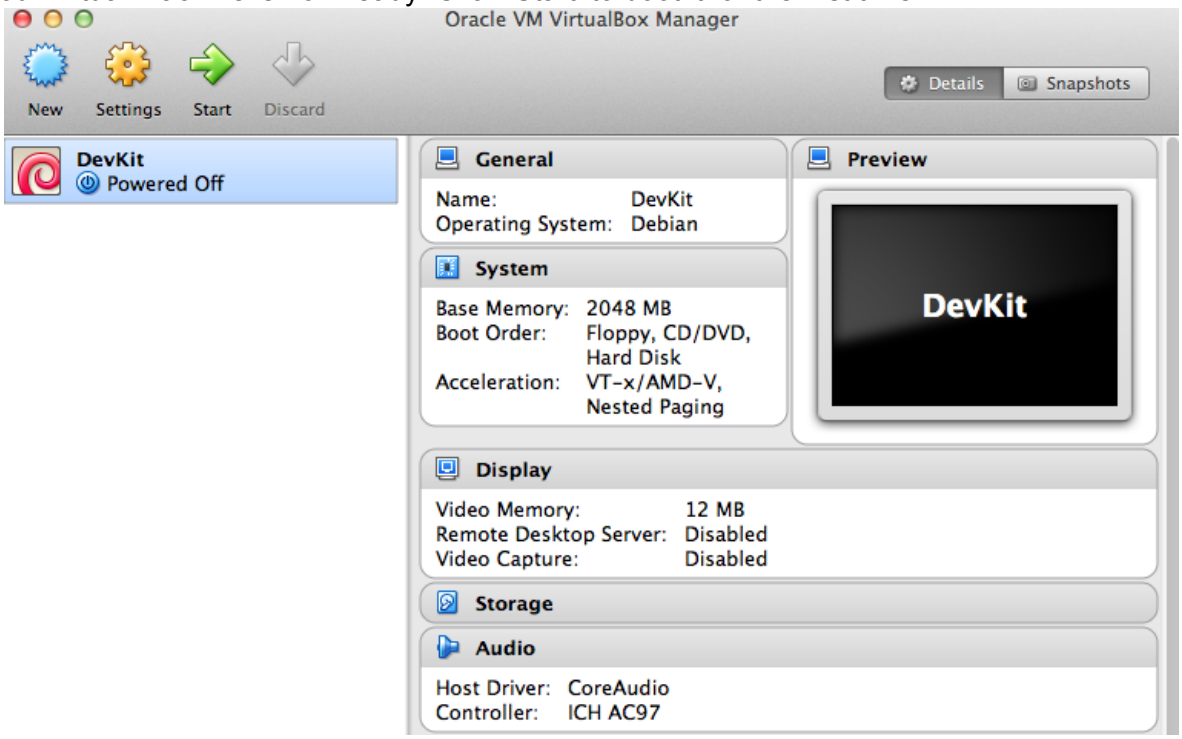


Figure 13: VirtualBox Status on Mac

15. Your machine will boot from the USB key and you will be presented with the following menu. For most modern computers the first (default) option should be sufficient (kernel 3.10). If you have older hardware or experience boot failures, please boot the 3.2 kernel (486 or 686-pae depending on your hardware).

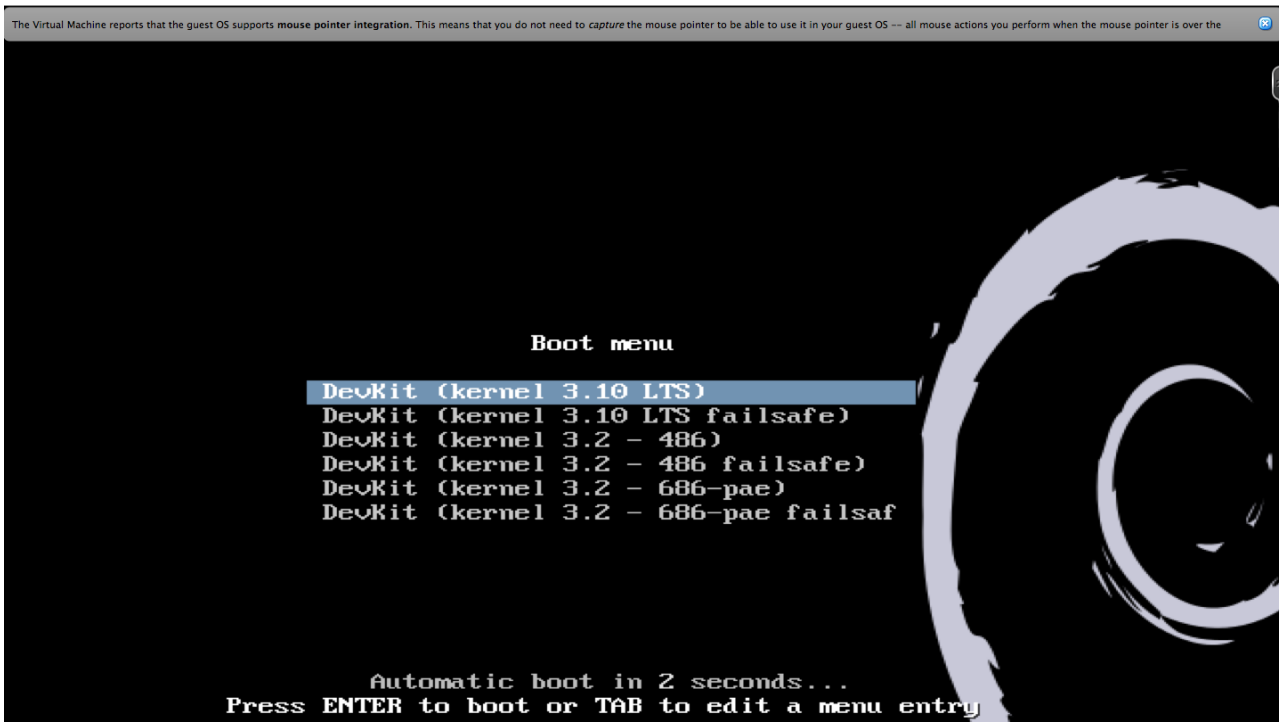


Figure 14: Boot Menu on Mac

2.4 Capturing Target IP address:

1. Connect Galileo target and your Host PC to the same network to verify that you have network connectivity between these two.
2. Use a terminal emulator (e.g. **minicom** / teraterm / putty) & connect to the Galileo target console with configuration '115200 Bit/s, 8 Bit, 1 Stop and No parity'. (Please refer to the Galileo release documentation for details on how to access the Galileo serial console).
3. Login as '**root**' user (*root user don't have password*) and find target IP address using command ***ifconfig*** as below. Note down the IP address at this stage.

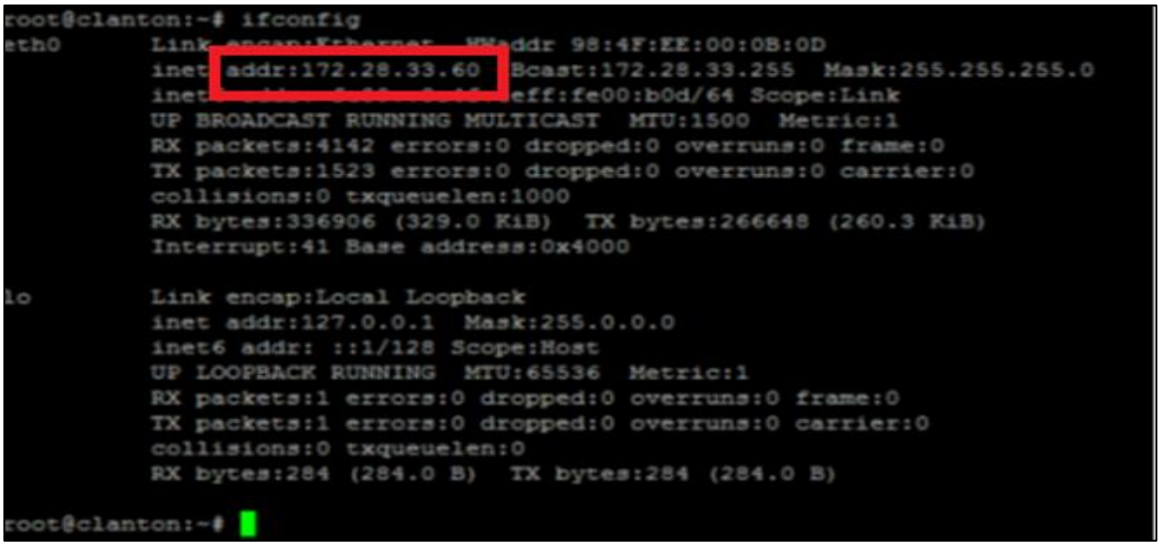


Figure 15: Development System Desktop

3 Setup Eclipse DevKit IDE

3.1 Configure Remote Target Connection

1. After successful boot-up from Live USB, the development system will show the desktop.
2. Launch the DevKit IDE by double clicking the desktop 'Eclipse DevKit IDE' icon on the host PC.

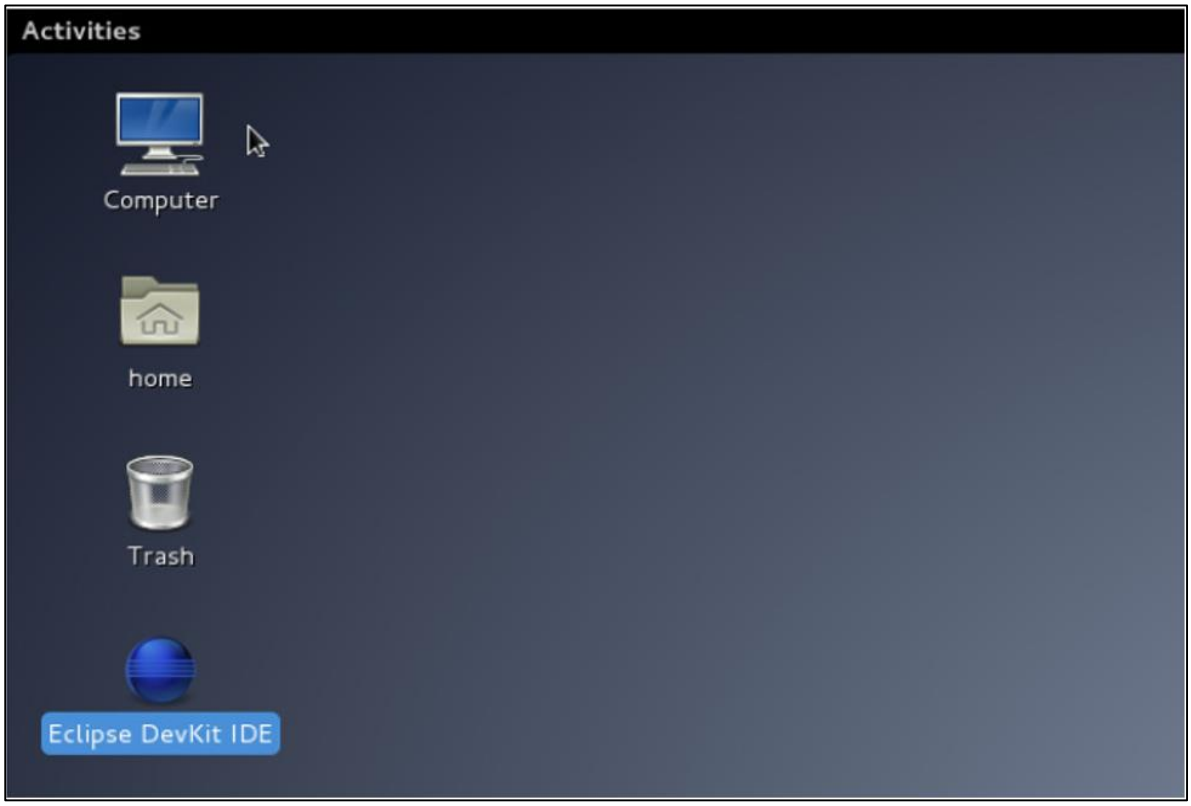


Figure 16: Development System Desktop

3. Inserting the Galileo target IP address captured in the previous step.
   a) Select menu **Window->Open Perspective->Other->Remote System Explorer**
   b) Right click the '**galileo'** connection and select **Properties -> Host**
   c) First time, it will ask for the root password, there is no password for root, hence simply press OK.

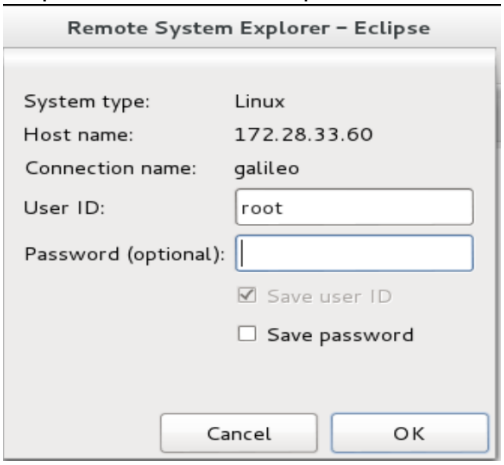


Figure 17: Root Password Dialog

d) Enter **IP address captured earlier** in the Host Name field.

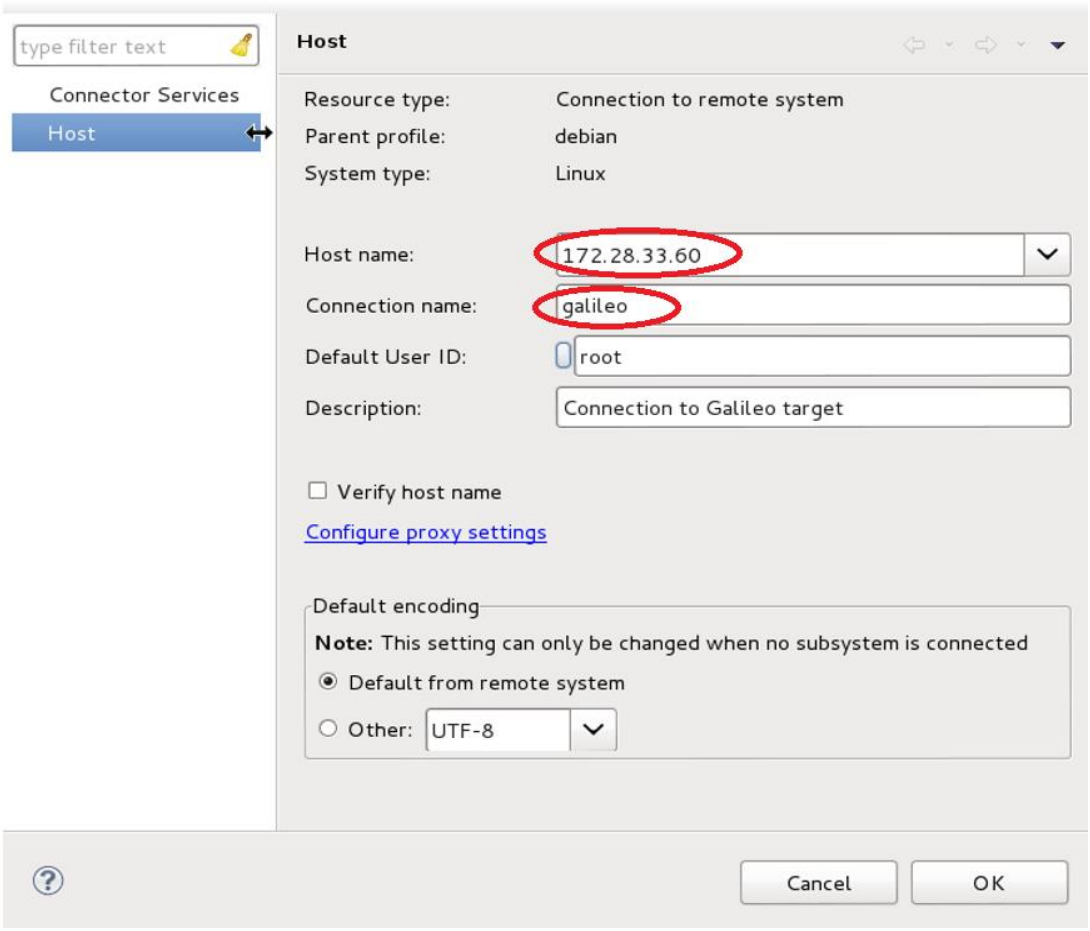


Figure 18: Connection to remote target

3.2 Toolchain Selection

The DevKit Development provides **GNU** and Intel C/C++ Compiler (**ICC**) Toolchain. NOTE: The build environment cannot be dynamically changed.

NOTE: Please note that the build environment cannot be dynamically changed i.e. if you project was created to use the GNU compiler, you will NOT be able to dynamically convert it to build with the ICC.

1. To select required Toolchain for our project, click the **Window tab-> Preferences-> Yocto Project ADT->Select GNU** or **ICC** in the Cross development profiles drop down menu.

2. Click **Apply** to change build environment and '**OK'** to close the Window.

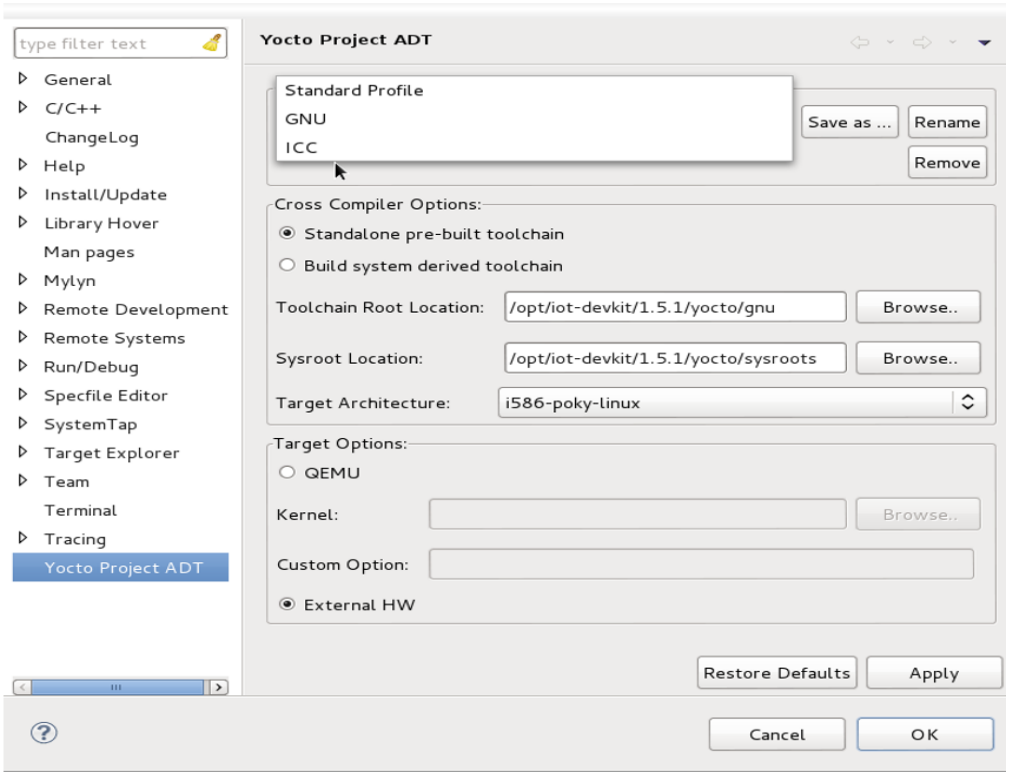


Figure-19: Connection to remote target

4 ‘Hello World’ Project

After selection of appropriate required tool chain, now you are ready to create your first project

4.1 Create and build project using GNU

1. Select **GNU** Toolchain as described earlier in ‘**Toolchain Selection**’ section.
2. Ensure that are on default C/C++ perspective view by selecting menu **Window->Open Perspective->Other->C/C++ (default)** and press **OK.**
3. From menu, select **File** → **New** → **C Project** → **Yocto Project ADT Autotools Project** → **Hello World ANSI C Autotools Project**
4. Enter a project name (e.g: **myGNUproject1**) → **Next** → **Complete** the required fields (Author, License etc.) → Click **Finish**
   NOTE: Please use only alphanumeric characters in the project name field and no spaces

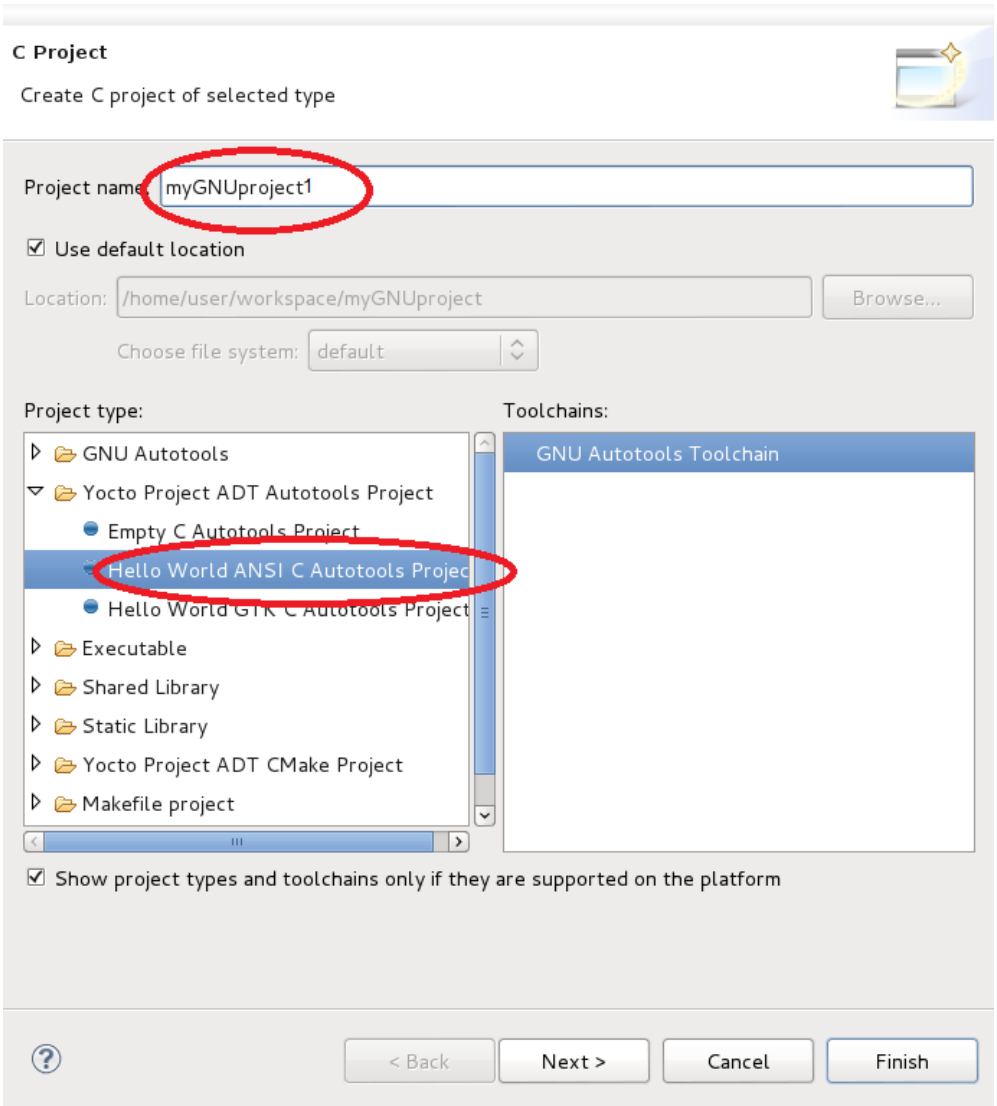


Figure-20: Project creation using GCC Toolchain

This will create a default “Hello World” project using the GNU Toolchain and sysroot for your Galileo target.

5. To build this created project for the first time **right click on project name** and select and click the option '**Build Project**'.

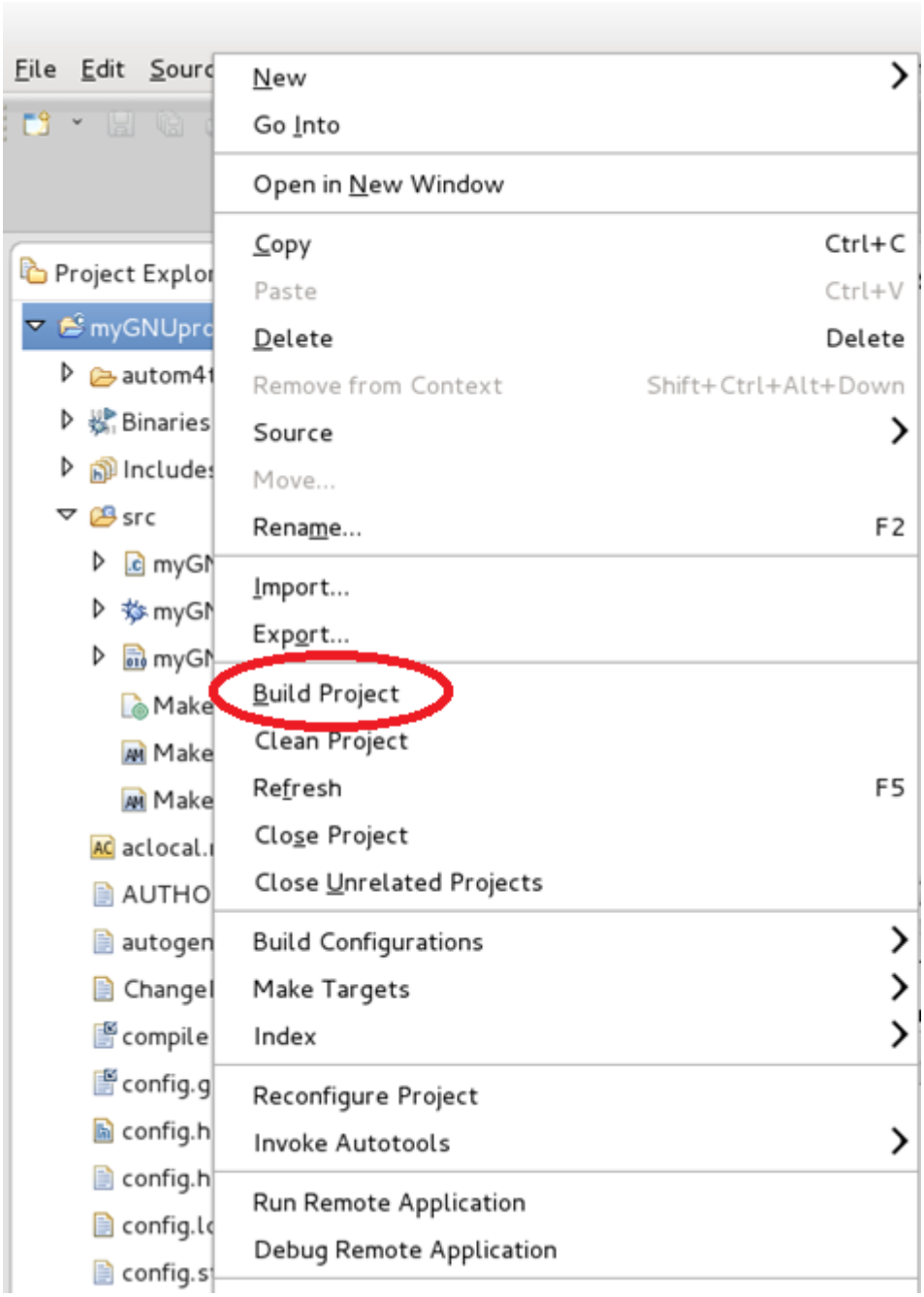


Figure-21: GCC project build

4.2 Create and build project using ICC

1. Select **ICC** Toolchain as described in '**Toolchain Selection'** section.
2. Ensure that are on default C/C++ perspective view by selecting menu **Window->Open Perspective->Other->C/C++ (default)** and press **OK.**
3. From menu, select **File** → **New** → **C Project** → **Yocto Project ADT Autotools Project** → **Hello World ANSI C Autotools Project**
4. Enter a project name (e.g: **myICCproject1**) → **Next** → **Complete** the required fields (Author, License etc.) → Click **Finish**

NOTE: Please use only alphanumeric characters in the project name field and no spaces

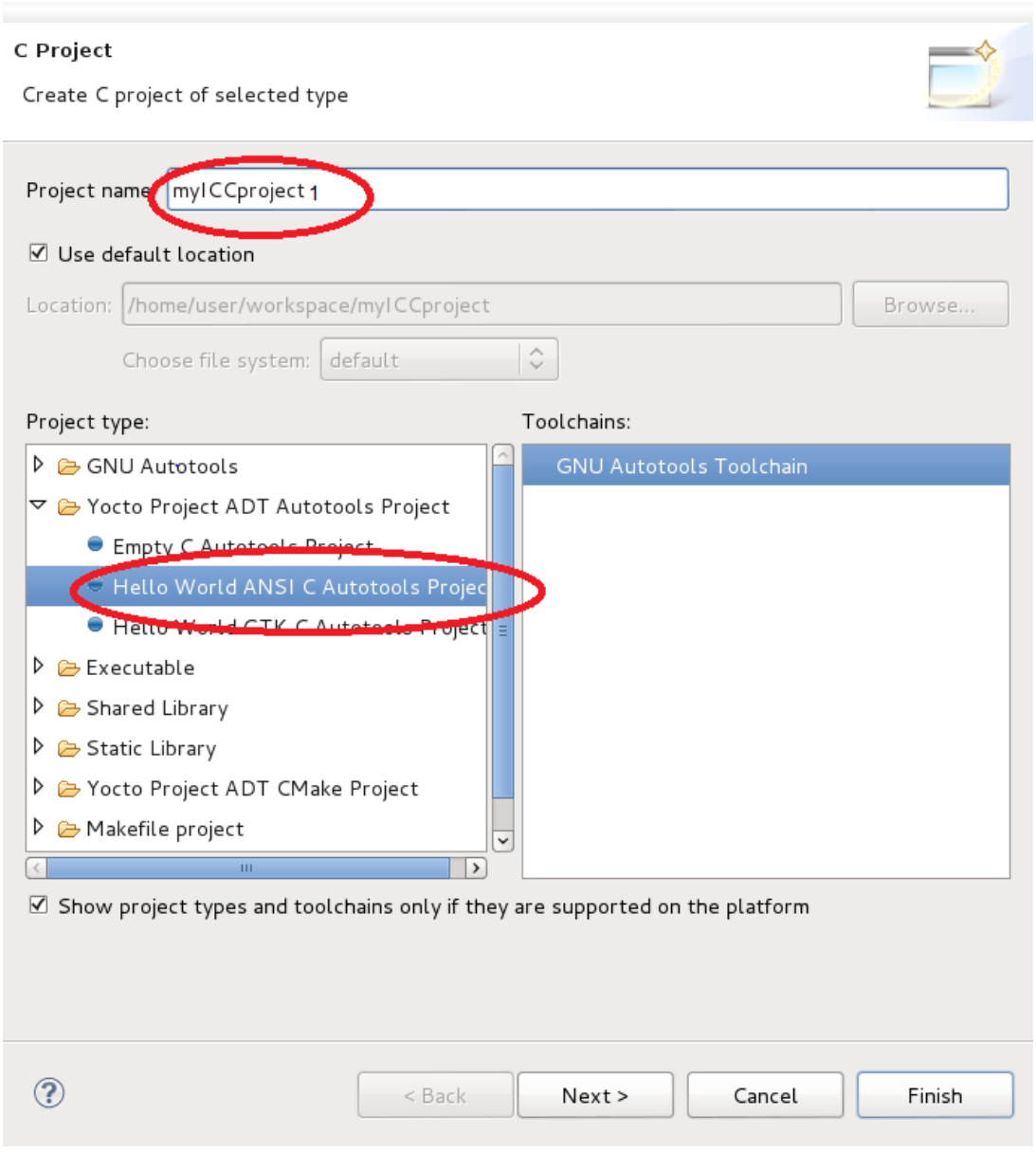


Figure-22: Project creation using ICC Toolchain

This will create a default "Hello World" project using the ICC Toolchain and sysroot for your Galileo target.

5. To build this created project for the first time **right click on project name** and select and click the option '**Build Project**'

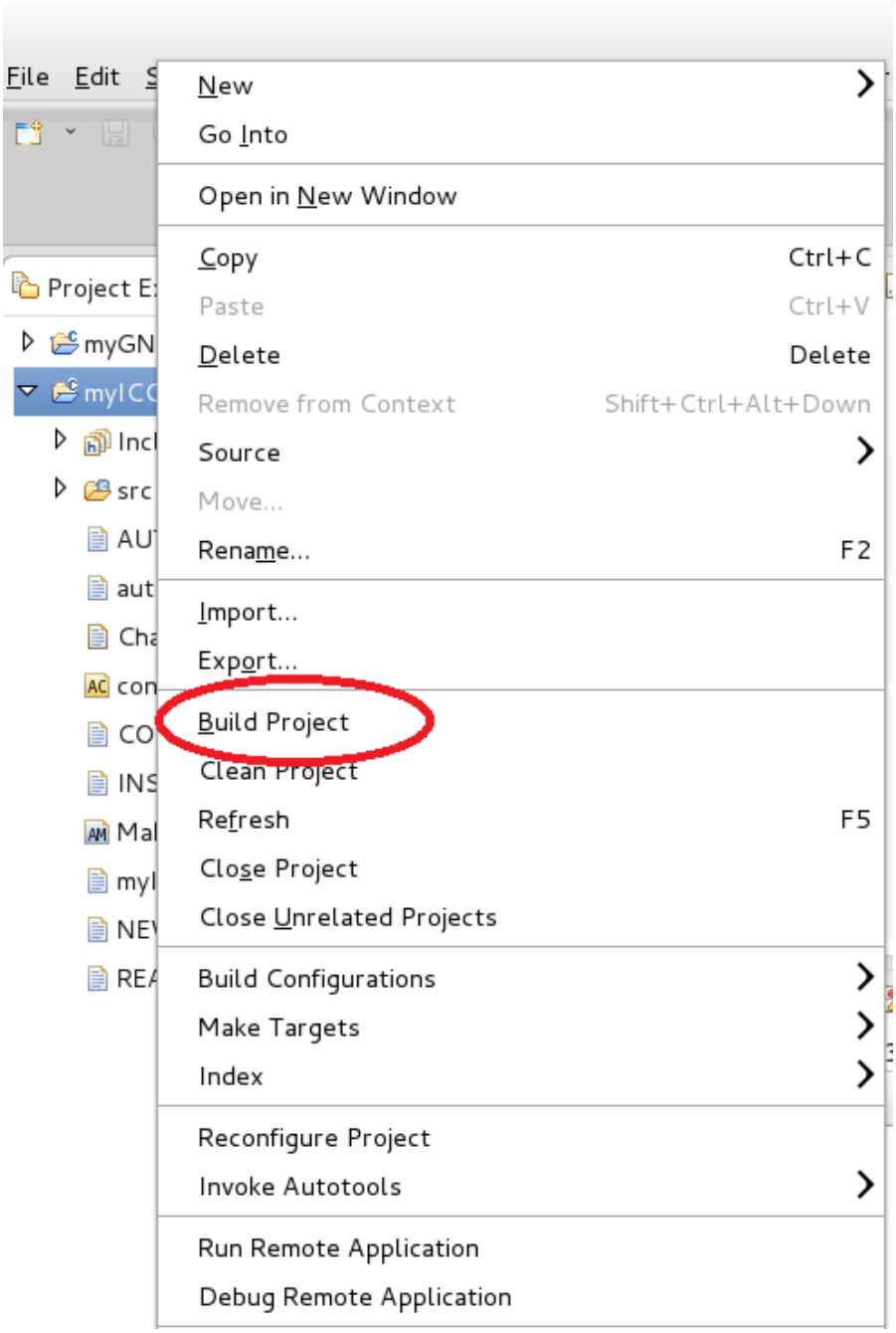


Figure-23: ICC project build

6. On successful compilation and build it will generate a binary file suitable to debug / run on Galileo target board.

4.3 Debug project

1. To transfer, run and debug your application remotely on your Galileo target, **right-click your project folder→ Debug As → Debug Configurations**.

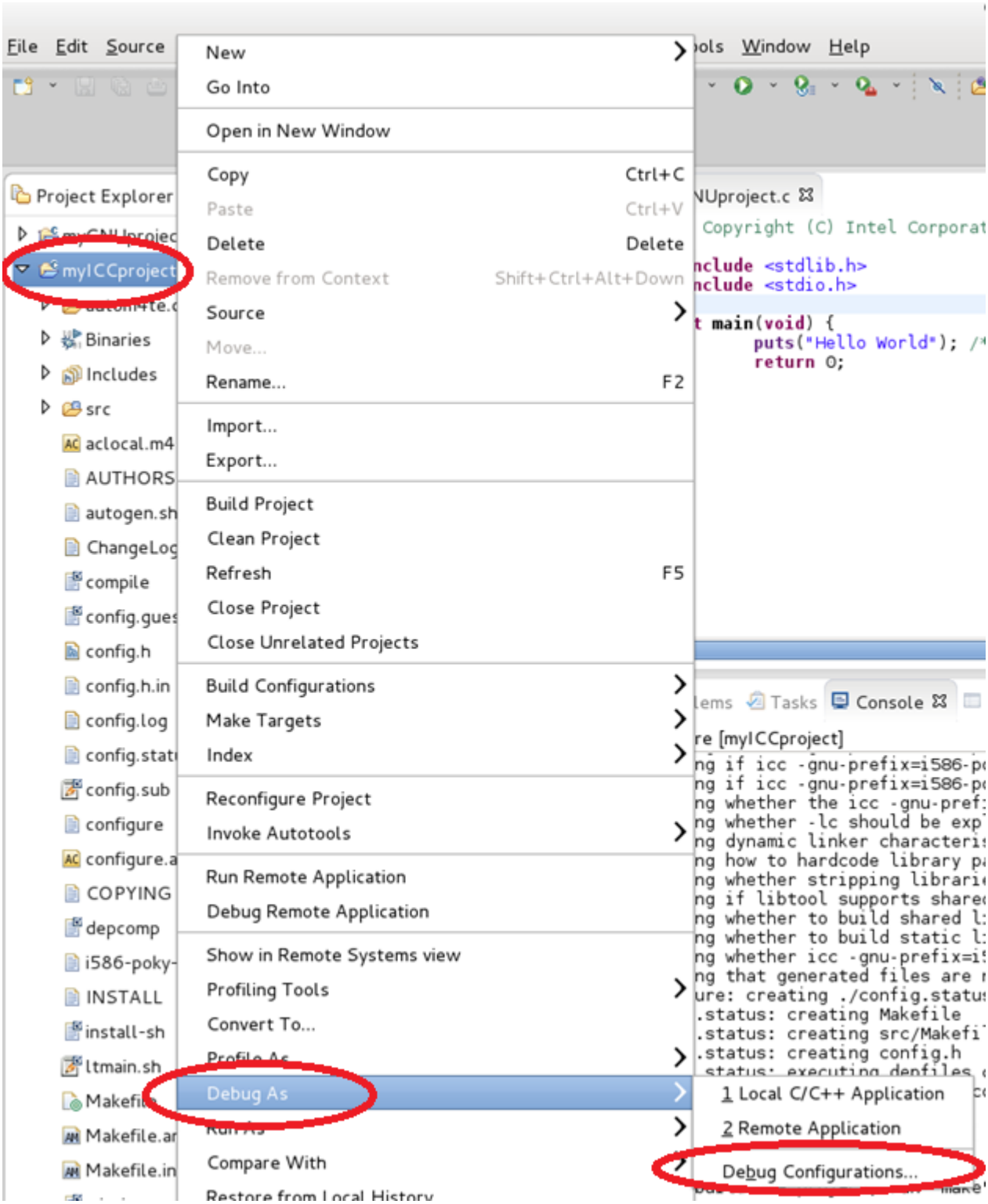


Figure-24: Debug project configuration

2. Select **'C/C++ Remote Application'** → Your project configuration name (e.g. **myICCproject_gdb_i586-poky-linux**) → Select **'galileo'** in the connection drop-down menu.

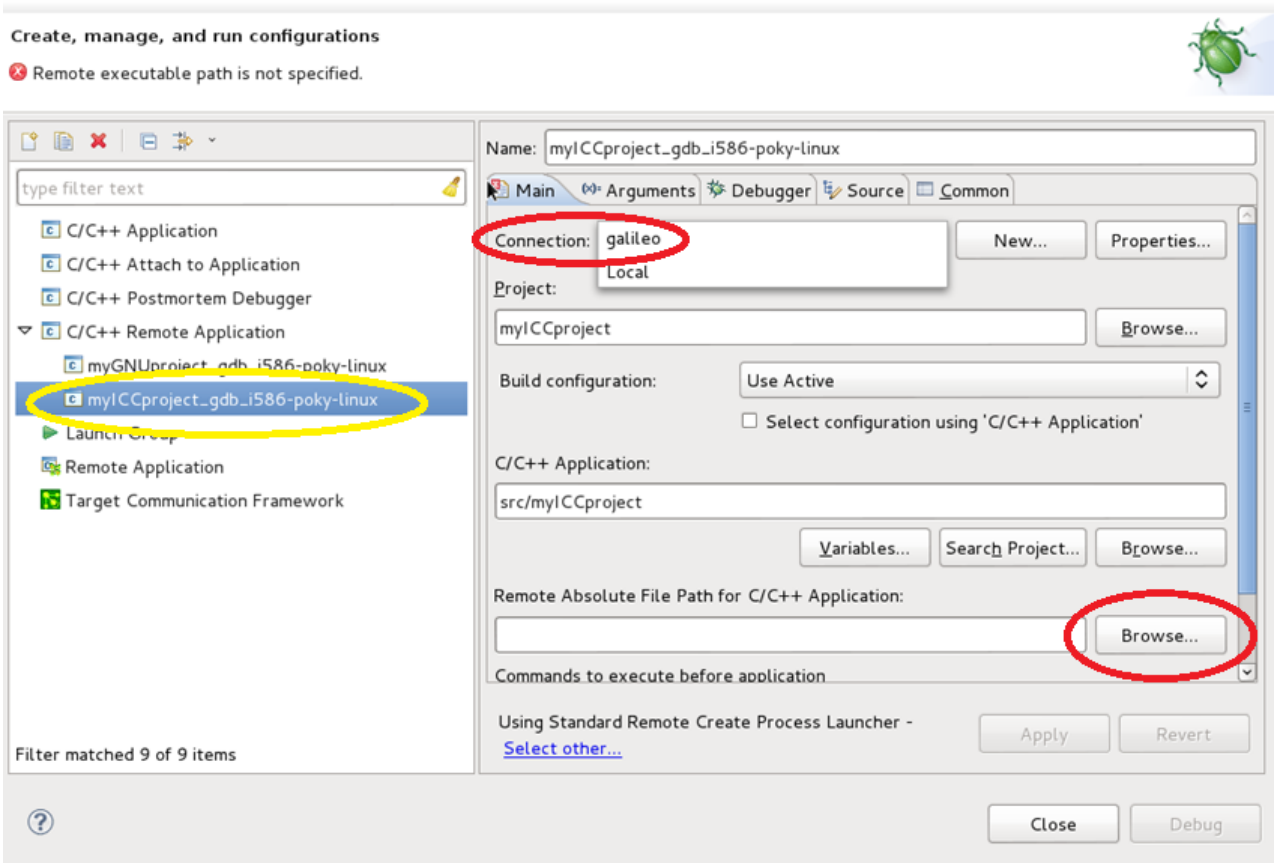


Figure-25: Debug configuration remote path setup

3. By clicking the 'Browse' button next to the 'Remote Absolute File Path for C/C++ Application'.
4. For the first time access to the Galileo board, the system will prompt you to enter root password optionally.

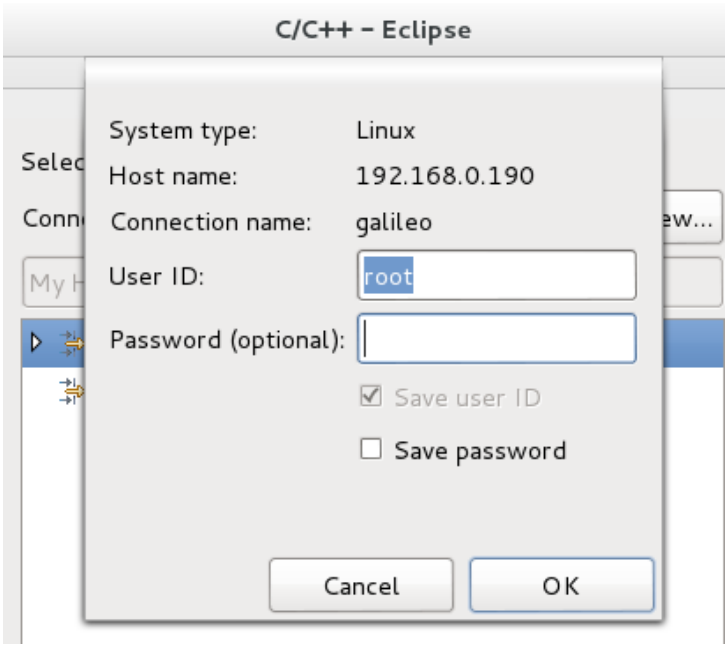


Figure-26: Optional root password to access remote target

5. Expand the **'Root'** tree and select where on the target you wish your executable to be saved (e.g. **/tmp/myICCproject**) → Click the OK button

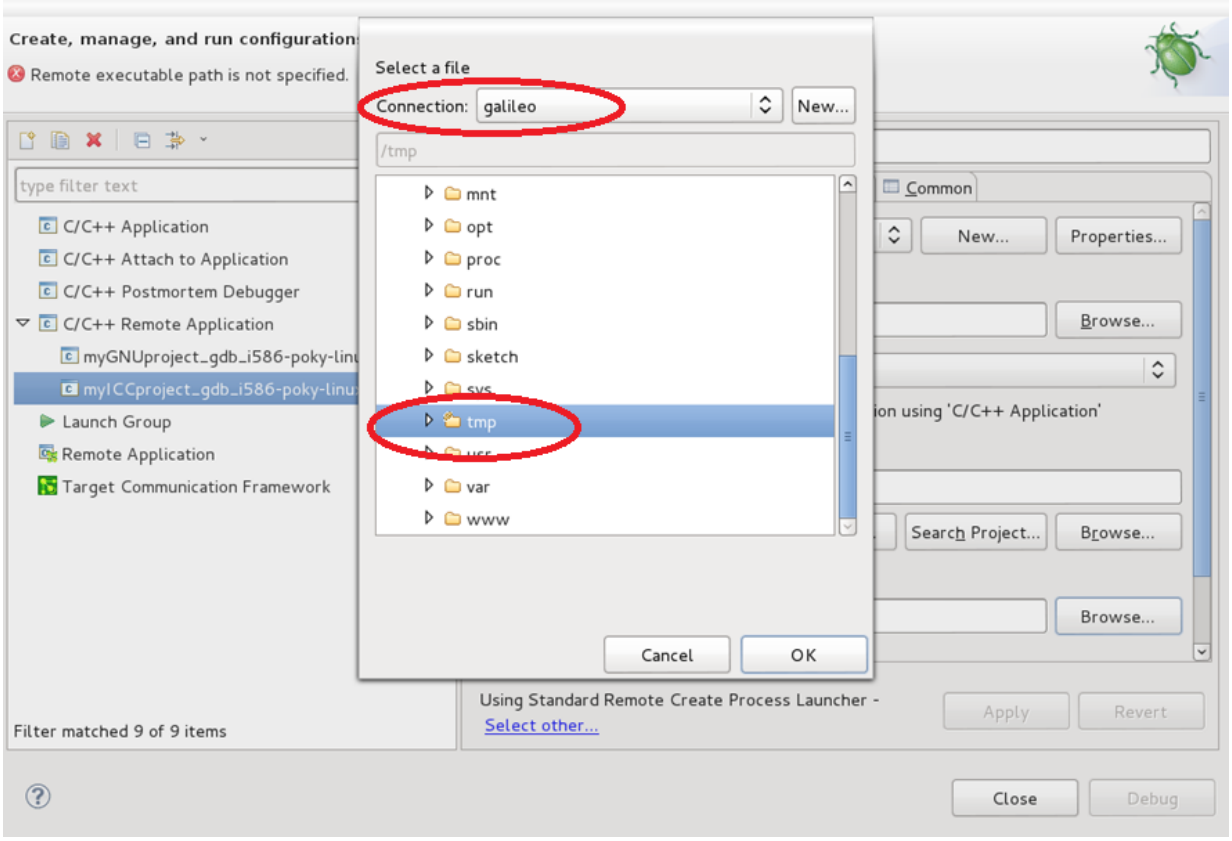


Figure-27: Destination of binary on remote target

6. In the **'Remote Absolute File Path for C/C++ Application'** field enter a file name for your binary on the target (e.g myICCproject)
7. In the **'Commands to execute before application'** use the chmod command to make your application executable on the target e.g. **chmod 755 /tmp/myICCproject**

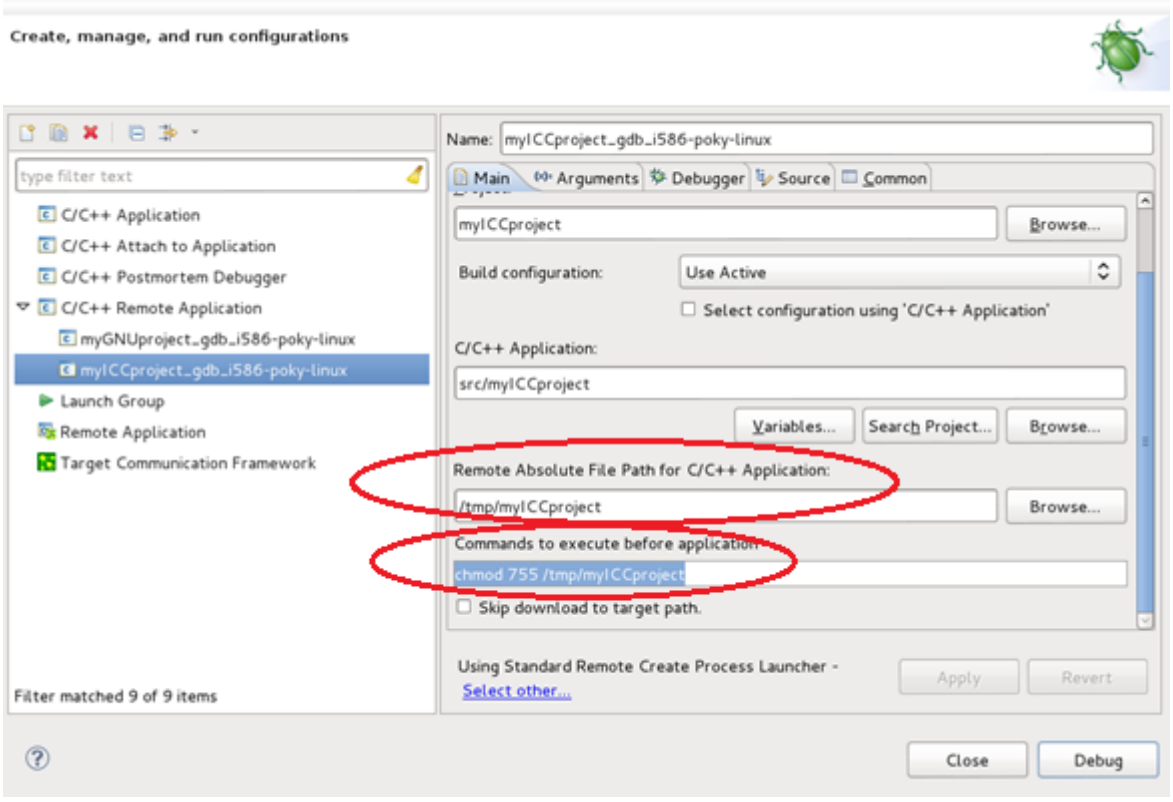


Figure-28: Command setup for binary on remote target

8. Click ‘Apply’ to apply the settings -> Click the 'Debug' button to transfer the application to the target, launch it and start the debugger. This will open the DevKit IDE debug perspective and insert a breakpoint at the main function. Use the control buttons on the toolbar to control the program execution (step into, step over etc.)

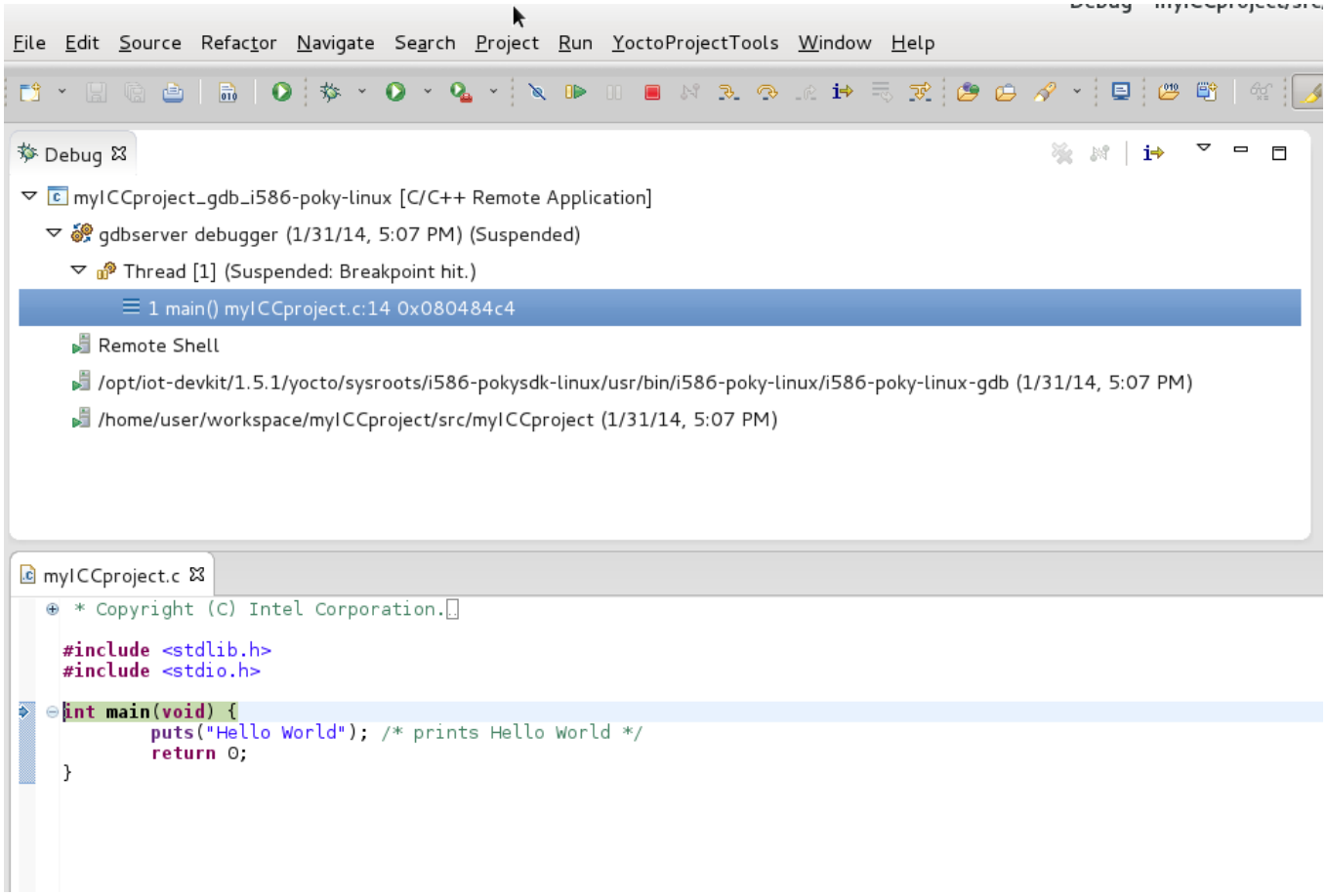


Figure-29: Code ready to debug / run on remote target

9. Modify your program as needed and use the build and debug steps described above for the rest of the development process. Once your program is finalized you can use the 'mv' command on the Galileo console to move the application binary to its final location e.g:
mv /tmp/my_project /usr/bin/my_final_app

4.4 Manually transfer binary to remote target

After your application is finalized you can transfer it to your Galileo target using the DevKit IDE.

1. Open the 'Remote System Explored' perspective **Window -> Open Perspective - > Remote System Explorer**
2. Locate the application binary in the 'Remote System Explorer' tab and copy it e.g. Expand the '**Local' tab** in the '**Remote System Explored'** window -> '**My Home' -> 'workspace' -> 'myICCproject' -> 'src' -> right click 'myICCproject' -> Click 'Copy'**

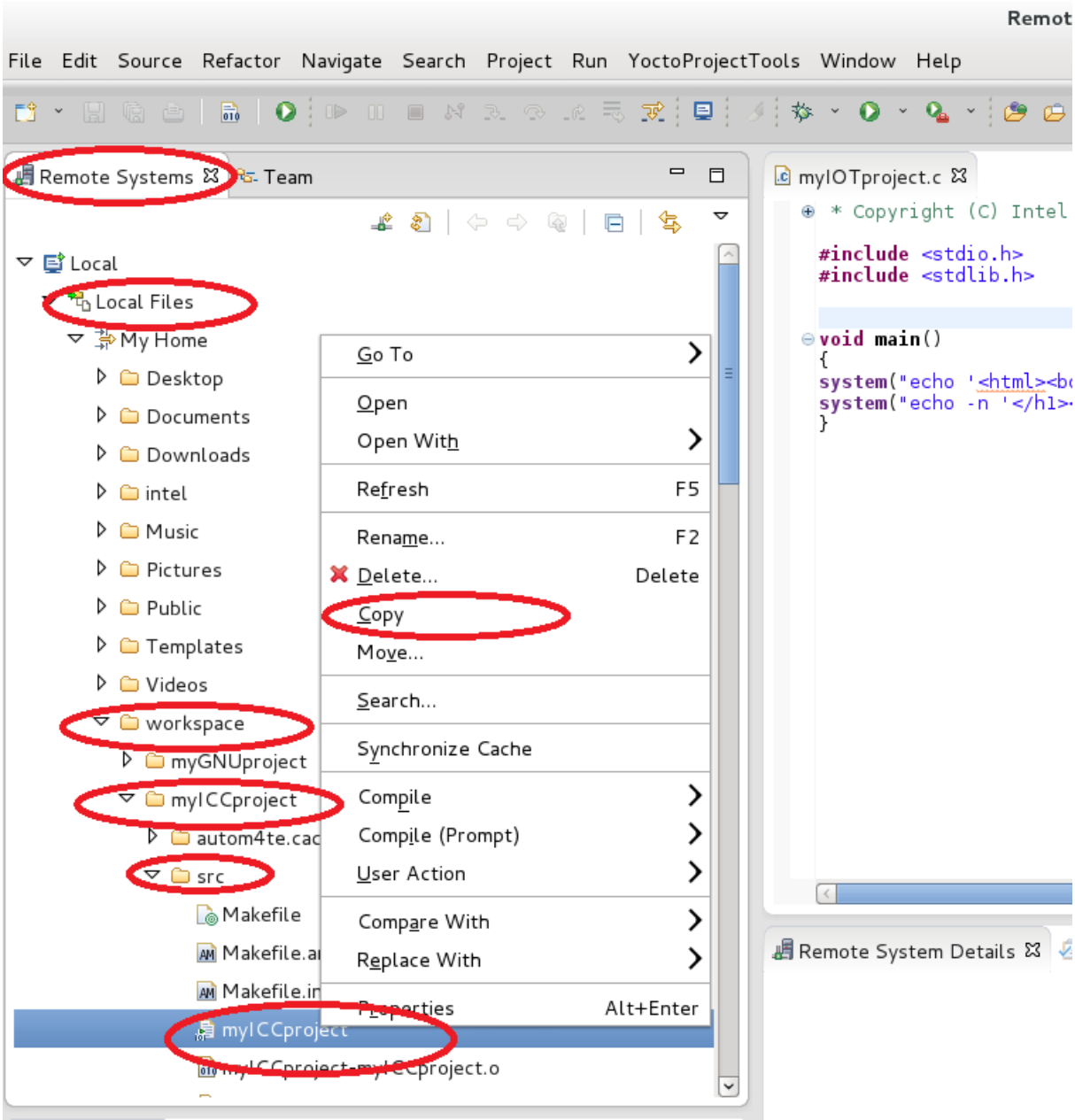


Figure-30: Copy code to remote target

3. Transfer the binary to its desired location on the Galileo target e.g. Expand the '**galileo'** tab in the '**Remote System Explored' window -> 'Sftp Files' -> 'Root' -> '/'-> right click 'tmp' -> Click 'Paste'**

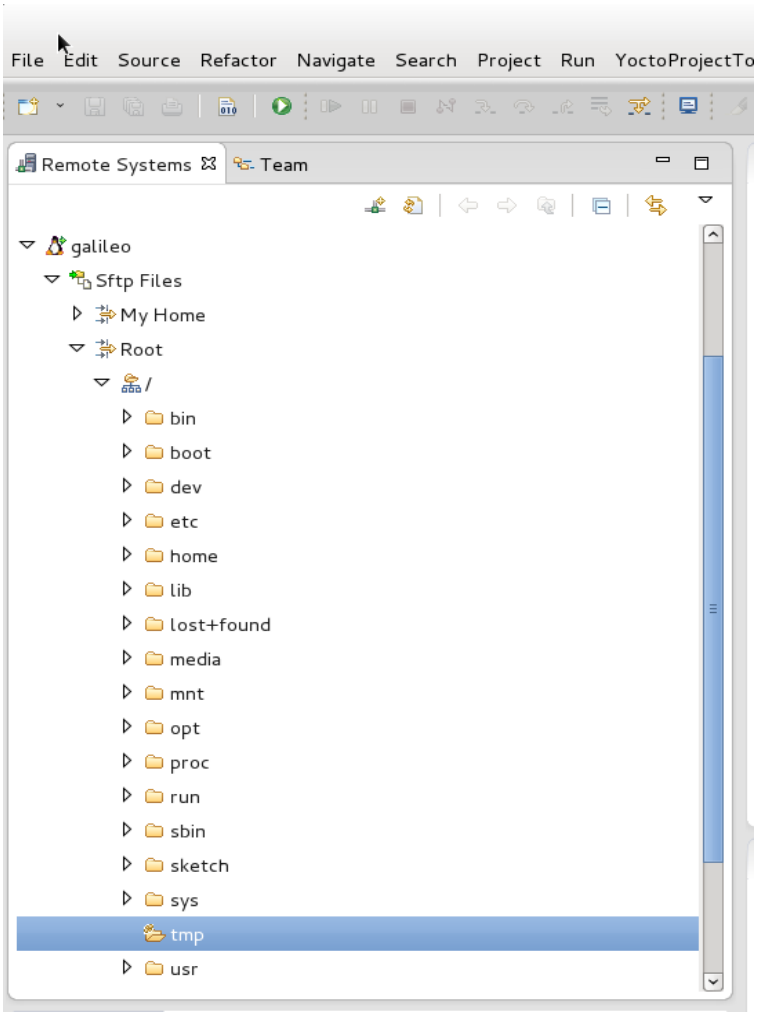


Figure-31: Destination for copy on remote target

4. Make the binary executable on your Galileo target: **Right click** the binary transferred in the previous step -> **Select 'Properties' -> Click 'Permissions'** -> Adjust the permissions as required -> **Click Apply -> Click OK** to close the dialog

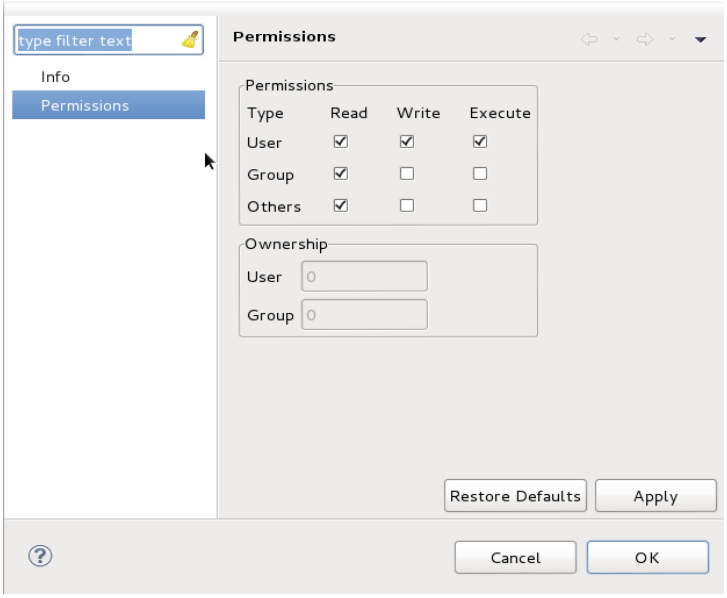
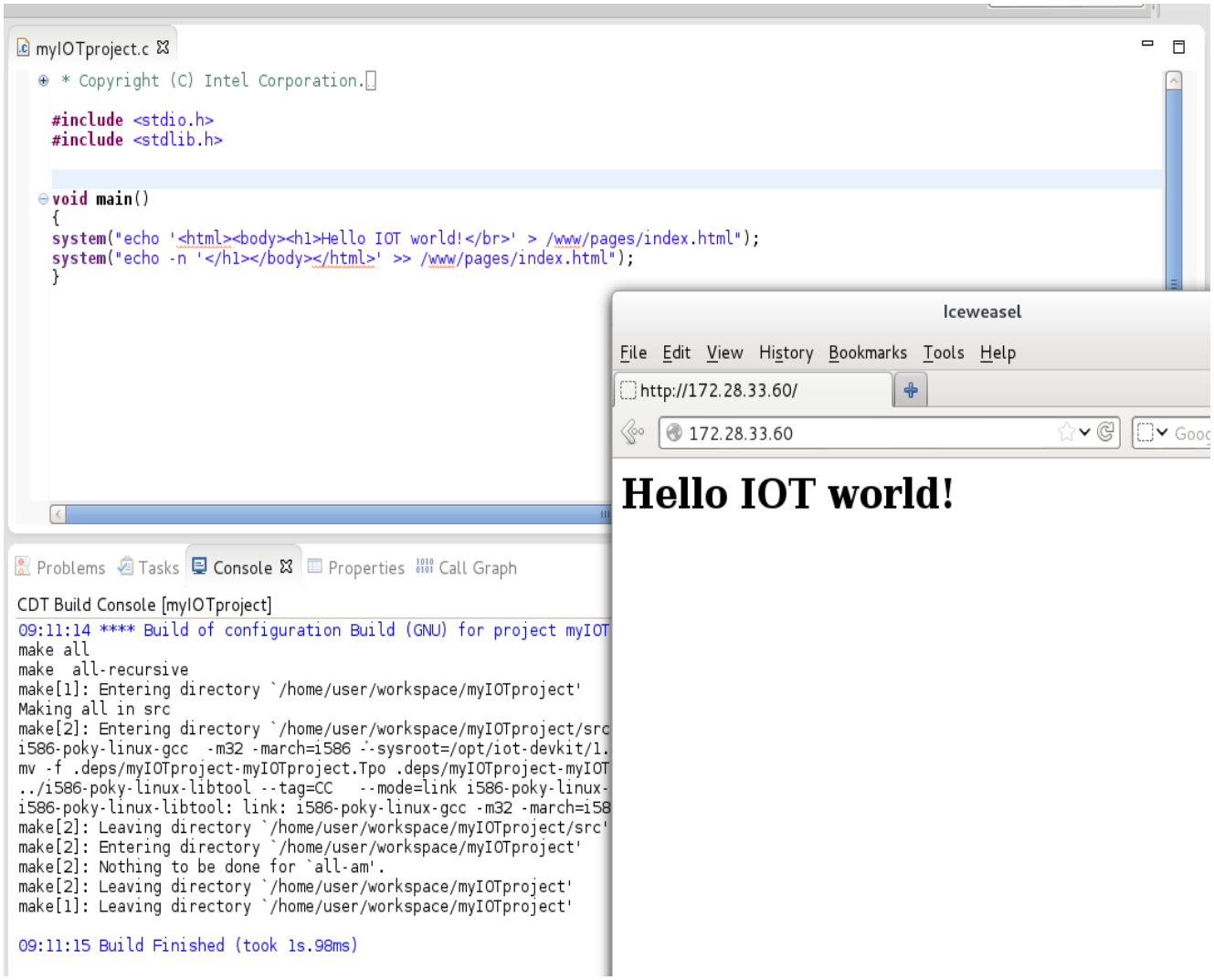


Figure-32: Destination for copy on remote target

5 Sample Applications

5.1 IoT Web Server

This application below shows an example of how you can utilize the DevKit development environment and the web server running on the Galileo target.



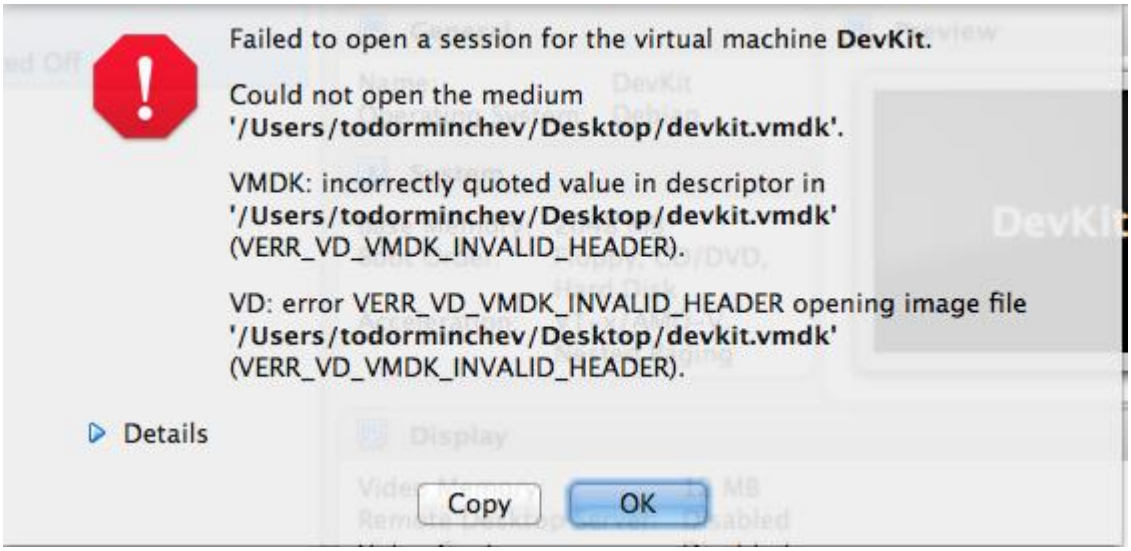
5.2 Other sample applications

Please refer to the provided source code of sample applications the SD card.

- LED Blink
- LCD Display
- Naui – Not A User Interface

6 Troubleshooting

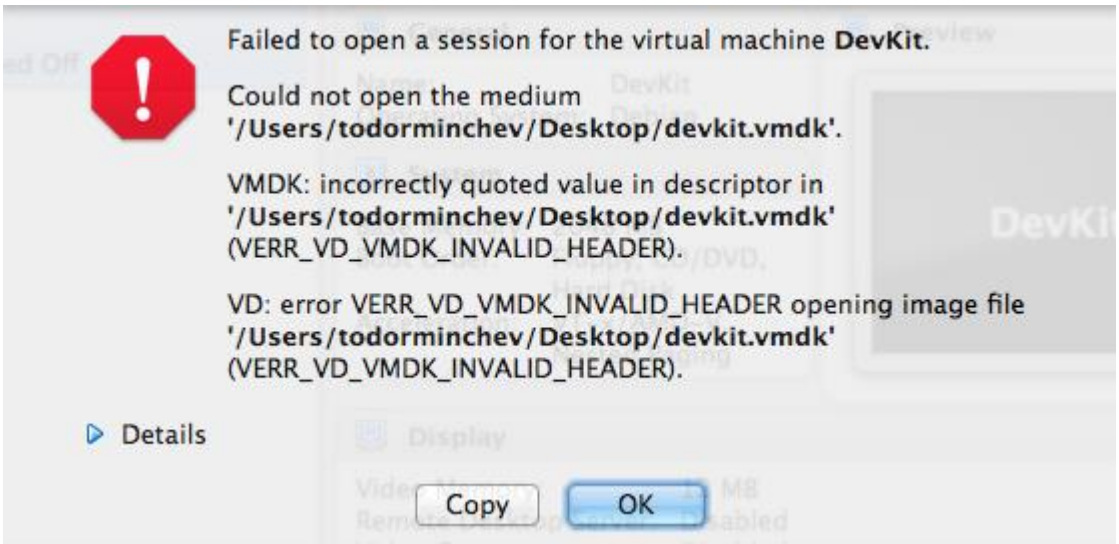
- You cannot create/start your Mac virtual machine and receive permission error messages.



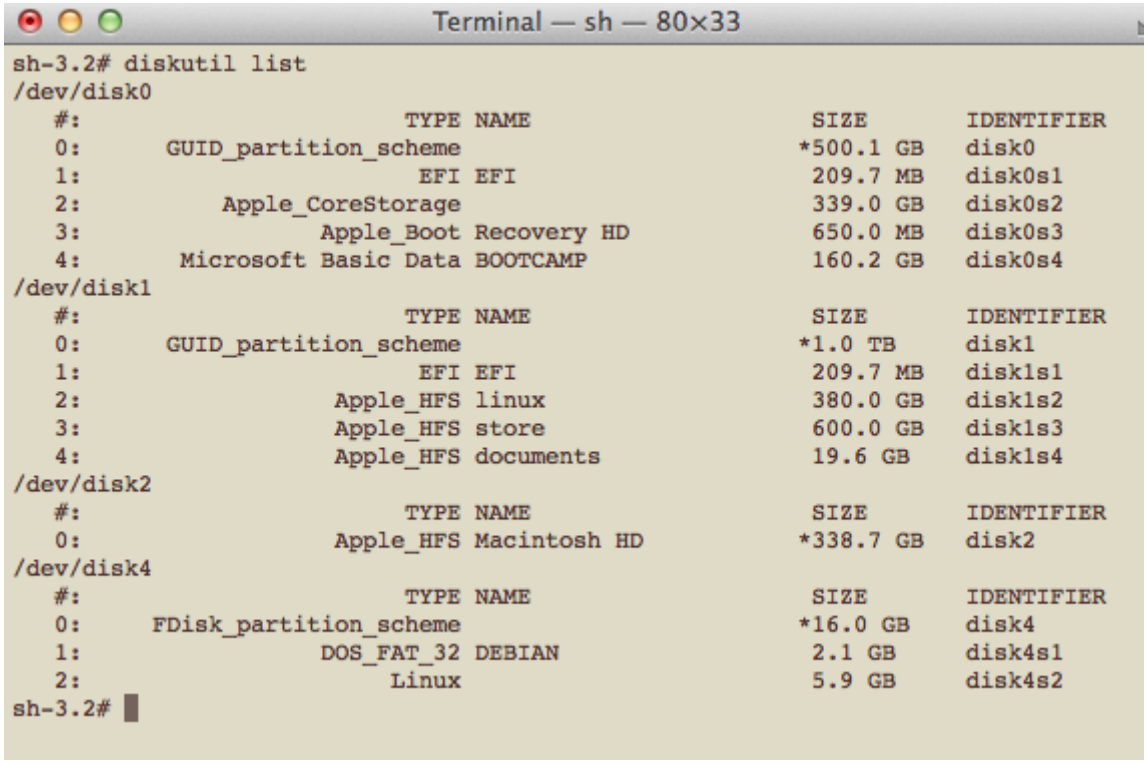
Solution: Please change the ownership of the USB device node e.g.

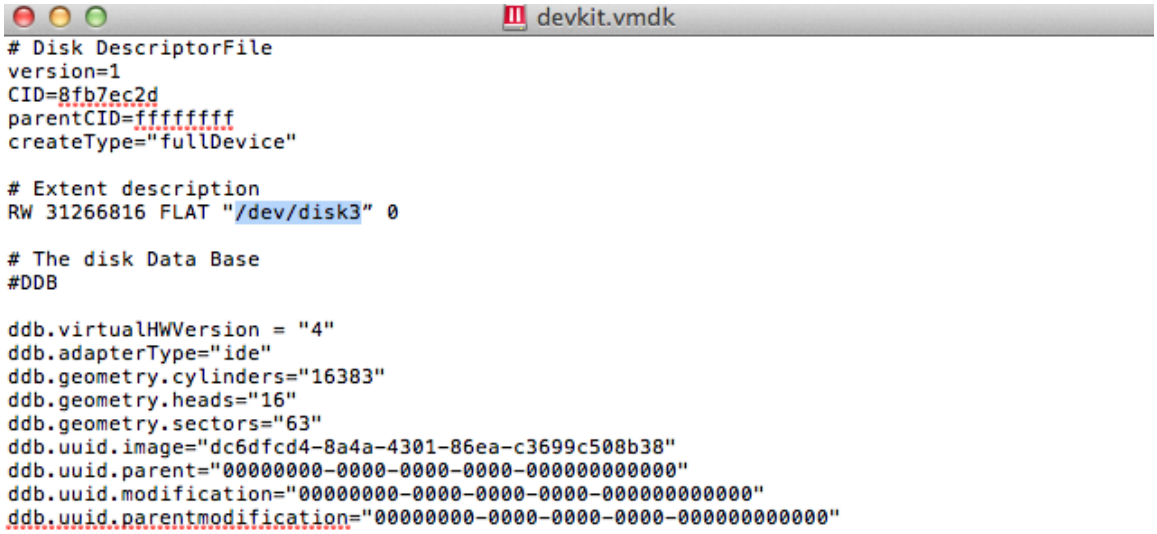
```
chown username  /dev/disk4
```

- You receive and invalid session error message when trying to start the VirtualBox virtual machine



Solution 1: Please use the 'diskutil list' command to verify that the device node of your USB key matched the device node in the virtual machine disk file:





Solution 2: Please use the '**diskutil unmountdisk'** command to unmount the USB key before starting the virtual machine