I Downloading and Setting Up the Cross Compiling Toolchain

- Create a rpi directory in your home directory by typing "mkdir rpi" while in your home directory.

- Go to the rpi directory with "cd rpi"

- and then type: git clone git://github.com/raspberrypi/tools.git

- This command will download (clone) Raspbian's official cross compiling toolchain from Github. The command will take a few minutes to complete its task.

- In a terminal window (on your Desktop Linux OS) type: cd ~/

  to point to the home directory, then type: nano.bashrc

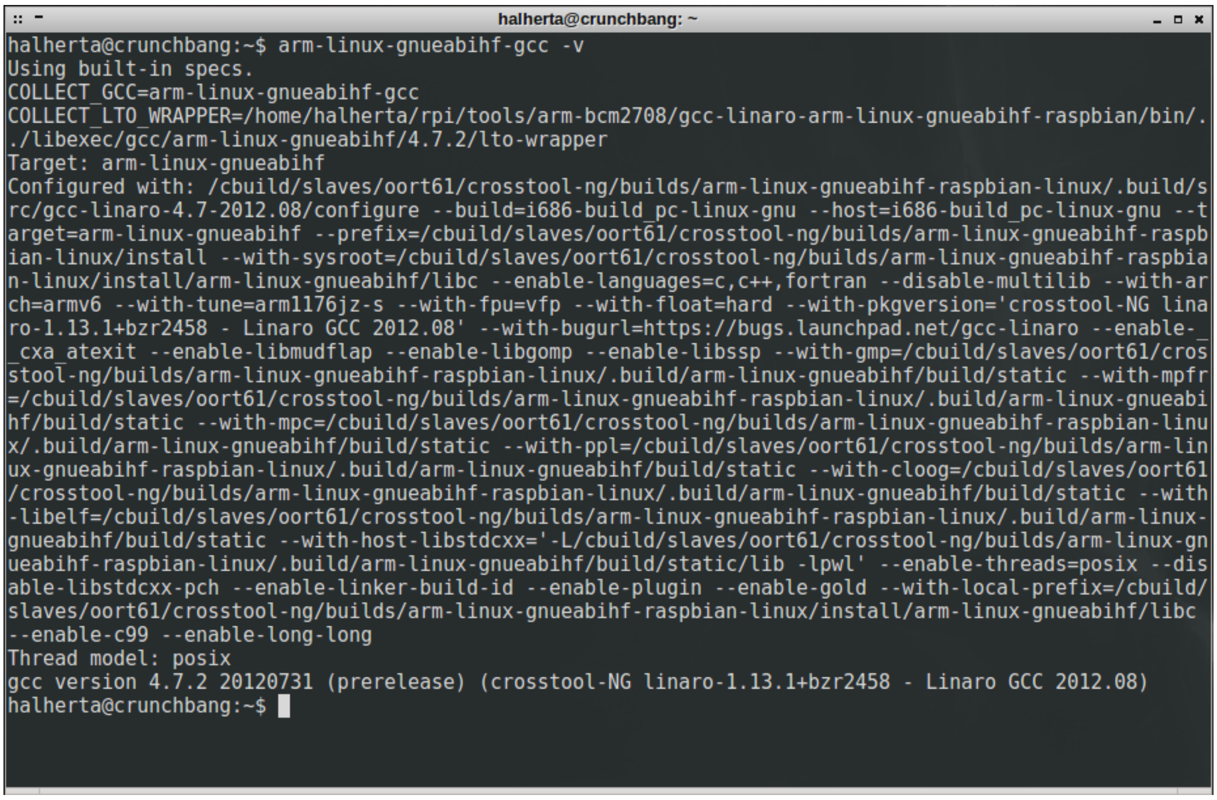
  This will open the .bashrc file in a command line based editor called nano. Go to the bottom of the .bashrc file using the down arrow key. Then type the following command:

```
$HOME/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin
```

- Then hit Ctrl+x to exit. You will be prompted to save changes. Say yes and hit "Enter".
- Then type: source .bashrc
- This allows the current open terminal to see the updated PATH variable. Alternatively one can close the terminal and open a new one.

- To test if you can access the cross compiling toolchain from the command line type: arm-linux-gnueabihf-gcc -v

  If you were successful, you should see output similar to that in Figure 1. Congratulations! you just installed Raspbian's official cross compiling toolchain on your Desktop PC / Virtual machine!



```
halherta@crunchbang: ~
halherta@crunchbang:~$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/home/halherta/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/.
./libexec/gcc/arm-linux-gnueabihf/4.7.2/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/.build/s
rc/gcc-linaro-4.7-2012.08/configure --build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --t
arget=arm-linux-gnueabihf --prefix=/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspb
ian-linux/install --with-sysroot=/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbia
n-linux/install/arm-linux-gnueabihf/libc --enable-languages=c,c++,fortran --disable-multilib --with-ar
ch=armv6 --with-tune=arm1176jz-s --with-fpu=vfp --with-float=hard --with-pkgversion='crosstool-NG lina
ro-1.13.1+bzr2458 - Linaro GCC 2012.08' --with-bugurl=https://bugs.launchpad.net/gcc-linaro --enable-_
_cxa_atexit --enable-libmudflap --enable-libgomp --enable-libssp --with-gmp=/cbuild/slaves/oort61/cros
stool-ng/builds/arm-linux-gnueabihf-raspbian-linux/.build/arm-linux-gnueabihf/build/static --with-mpfr
=/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/.build/arm-linux-gnueabi
hf/build/static --with-mpc=/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linu
x/.build/arm-linux-gnueabihf/build/static --with-ppl=/cbuild/slaves/oort61/crosstool-ng/builds/arm-lin
ux-gnueabihf-raspbian-linux/.build/arm-linux-gnueabihf/build/static --with-cloog=/cbuild/slaves/oort61
/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/.build/arm-linux-gnueabihf/build/static --with
-libelf=/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/.build/arm-linux-
gnueabihf/build/static --with-host-libstdcxx='-L/cbuild/slaves/oort61/crosstool-ng/builds/arm-linux-gn
ueabihf-raspbian-linux/.build/arm-linux-gnueabihf/build/static/lib -lpwl' --enable-threads=posix --dis
able-libstdcxx-pch --enable-linker-build-id --enable-plugin --enable-gold --with-local-prefix=/cbuild/
slaves/oort61/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/install/arm-linux-gnueabihf/libc
--enable-c99 --enable-long-long
Thread model: posix
gcc version 4.7.2 20120731 (prerelease) (crosstool-NG linaro-1.13.1+bzr2458 - Linaro GCC 2012.08)
halherta@crunchbang:~$
```

The Next step is to download and install Eclipse. Unfortunately as of this writing the apt-get repositories rarely contain the latest Eclipse build.. So we will not be using the apt-get package manager. Instead we will download the latest Eclipse IDE from the web using a web browser.

- Before we can run Eclipse, we need to ensure that we have a Java runtime environment installed; since Eclipse relies heavily on Java. We can do this with the following command: sudo apt-get install openjdk-7-jre

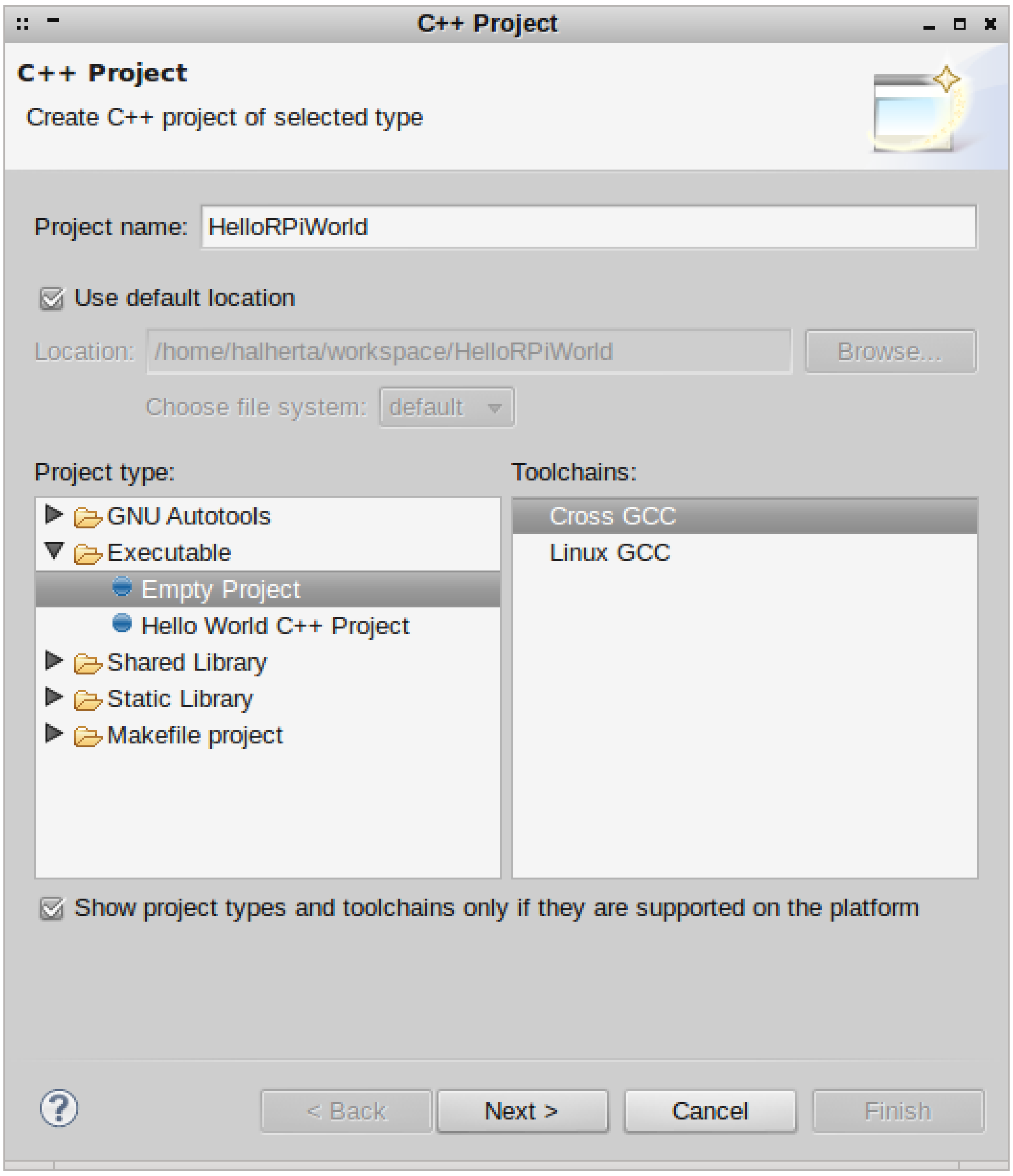
- Go to the link provided below and download the latest linux version of the Eclipse IDE for C/C++. Download the 32-bit version of the IDE if you're running a 32-bit Linux OS or download the 64-bit version of the IDE if you're running a 64-bit Linux OS.

Link: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplerr

- The next step is to extract the "eclipse" folder into our home directory from the eclipse-cpp-kepler-R-linux-gtk-x86_64.tar.gz compressed file. This can be easily done using File Manager and Archive Manager. Alternatively to perform the extraction from the command line, open a new console and navigate to the directory containing the .gz file (typically "~/Downloads/")then type: tar -xvzf eclipse-cpp-kepler-R-linux-gtk-x86_64.tar.gz -C ~/

II Downloading and Setting Up Eclipse

- Now go to File->New->C++ Project. This will open the "C++ Project" window shown in Figure 2.



C++ Project

Create C++ project of selected type

Project name: HelloRPiWorld

☑ Use default location

Location: /home/halherta/workspace/HelloRPiWorld

Choose file system: default

Project type:
- GNU Autotools
- Executable
  - Empty Project
  - Hello World C++ Project
- Shared Library
- Static Library
- Makefile project

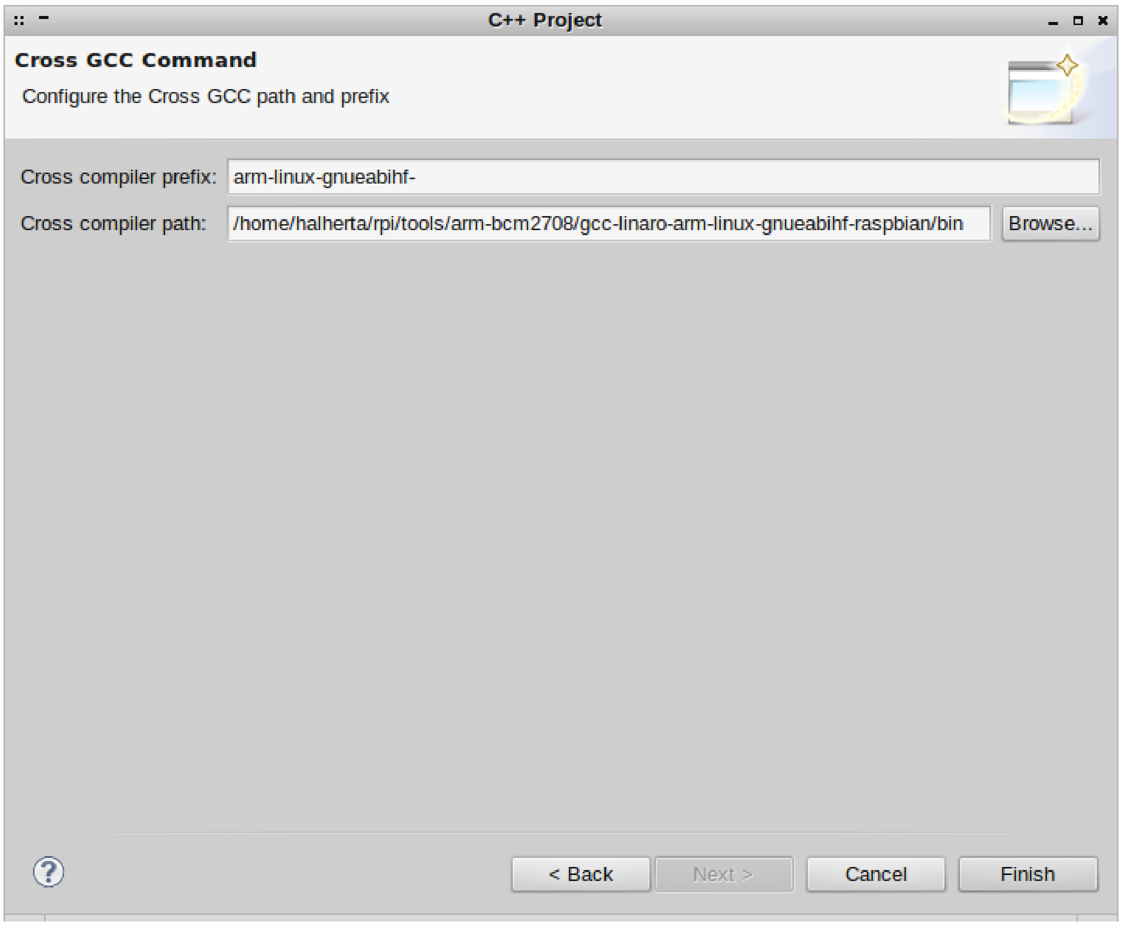
Toolchains:
- Cross GCC
- Linux GCC

☑ Show project types and toolchains only if they are supported on the platform

Figure 2. C++ Project Window

- Type "HelloRpiWorld" in the "Project Name" field. Under Project type select "Executable->Empty Project" Under "Toolchains" select "Cross GCC". Then click on "Next" This should take you to the "Select Configurations" Window. Accept the defaults and click "Next".
- This should then take you to the "Cross GCC command" Window (Figure 3).
  - In the "Cross compiler prefix" field type: "arm-linux-gnueabihf-".

- In the “Cross compiler path” field type the full path to the toolchain: “/home/**"username"**/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/”
- Click Finish. This step informs Eclipse of the location of the cross compiling toolchain and its standard C/C++ libraries and headers.



Cross compiler prefix: arm-linux-gnueabihf-

Cross compiler path: /home/halherta/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin

Figure 3. Configure the Cross GCC path and prefix

- . Click on “File->New Source File”.

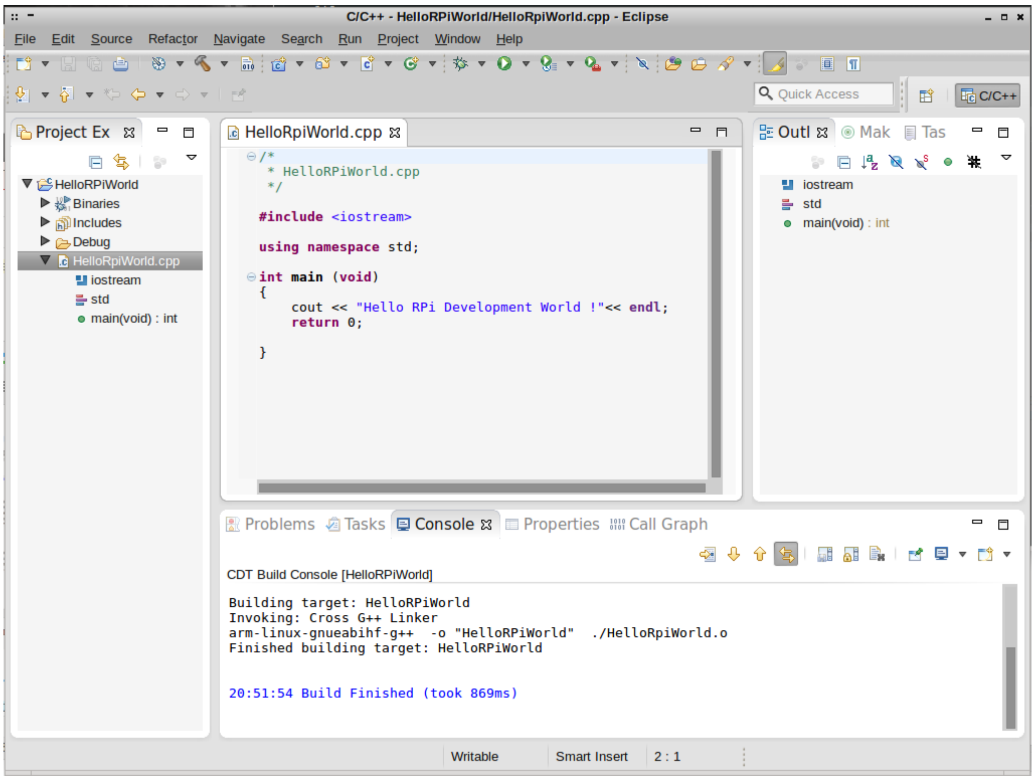
- The “Source folder” field should already have the project name “HelloRPiWorld” displayed in it. In the “Source file” field type “HelloRPiWorld.cpp” and click Finish. This will add a HelloRPiWorld.cpp source file to our project. In the source file copy the following C++ code:

```
/*
 * HelloRPiWorld.cpp
 */

#include <iostream>

```

```
using namespace std;

int main  (void)
{
        cout  <<  "Hello  RPi  Development  World  !"<<  endl;
        return 0;
}


```

- Save your project by clicking on “File->Save”.

To build the project click on “Project->Build Project”. To clean the project click on “Project->Clean”. Take note of the various options under the “Clean dialog box” that enable multiple projects to be clean at once as well as automatic rebuild. The output of the compilation/build process should be available for viewing in the “console” tab. Figure 5 shows the Eclipse workspace with the HelloWorld program successfully cross compiled for the Raspberry Pi!!!



```
/*
 * HelloRPiWorld.cpp
 */

#include <iostream>

using namespace std;

int main (void)
{
    cout << "Hello RPi Development World !"<< endl;
    return 0;

}
```

CDT Build Console [HelloRPiWorld]

```
Building target: HelloRPiWorld
Invoking: Cross G++ Linker
arm-linux-gnueabihf-g++  -o "HelloRPiWorld"  ./HelloRpiWorld.o
Finished building target: HelloRPiWorld


20:51:54 Build Finished (took 869ms)
```

Figure 5. HelloWorld program successfully compiled for Raspberry Pi.!!!

III Deploying the Binary file onto the Raspberry Pi

To deploy the binary on the Raspberry Pi, make sure that the RPi board is powered and connected to your network. The next step is click on "Window->Show View->Other". This will open a show view window. In this window select the "Remote Systems" folder and then select "Remote Systems Details" and press OK (Figure 6).

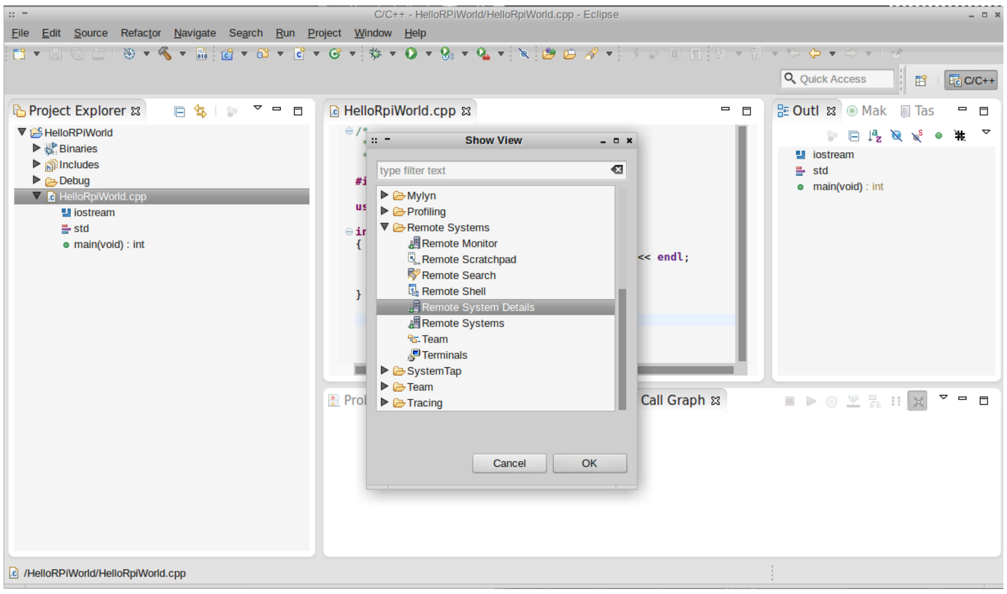
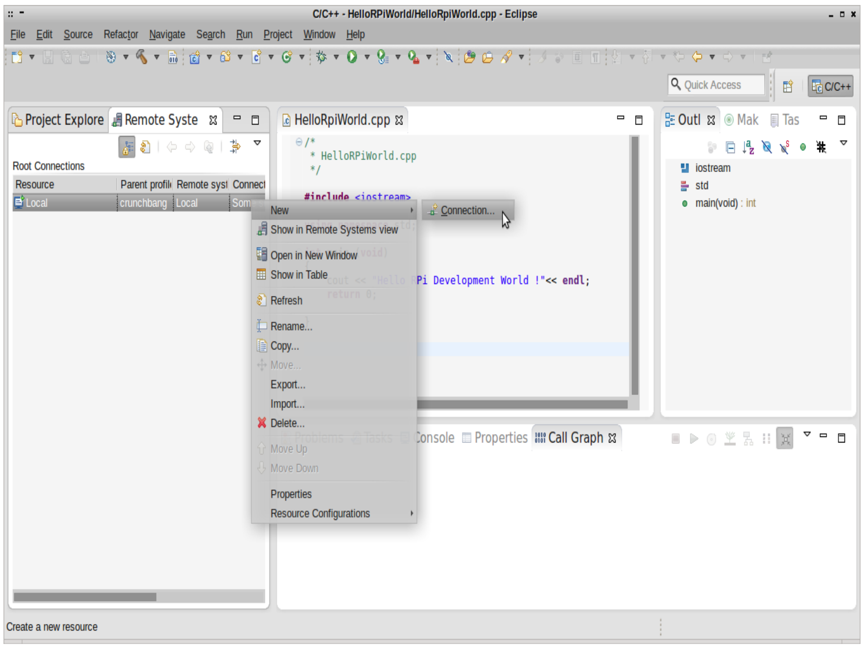


Figure 6. Remote system view

- If the "Remote Systems Details" shows up in the bottom of the workspace, drag it such that its tab shares the same region as the project explorer tab as shown in Figure 7.



```
/*
 * HelloRPiWorld.cpp
 */
#include <iostream>
...Pi Development World !"<< endl;
```

Figure 7. Creating a New Connection

- Now right click in the “Remote Systems Detail” area and click on “New Connection”. In the “Select Remote System Type” window (Figure 8), select “SSH only” then click “Next”

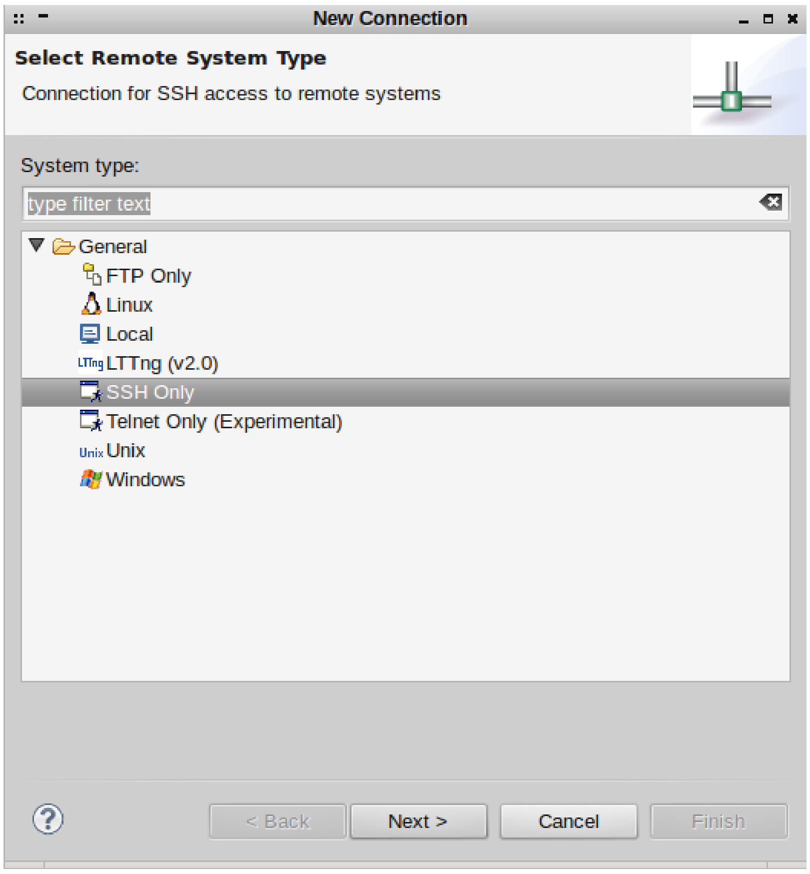
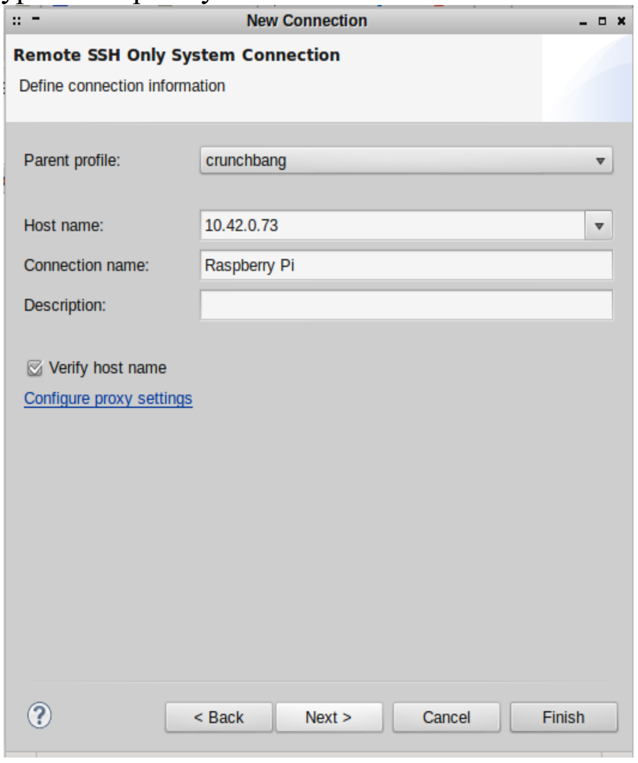


Figure 8. Select remote system type

In the “Remote SSH Only System Connection” (Figure 9) type in the IP address of the RPi in the “Host name” field and give the connection a valid name in the “Connection name” field. I typed “Raspberry Pi”. Then click “Finish”.


New Connection

Remote SSH Only System Connection

Define connection information

Parent profile: crunchbang

Host name: 10.42.0.73

Connection name: Raspberry Pi

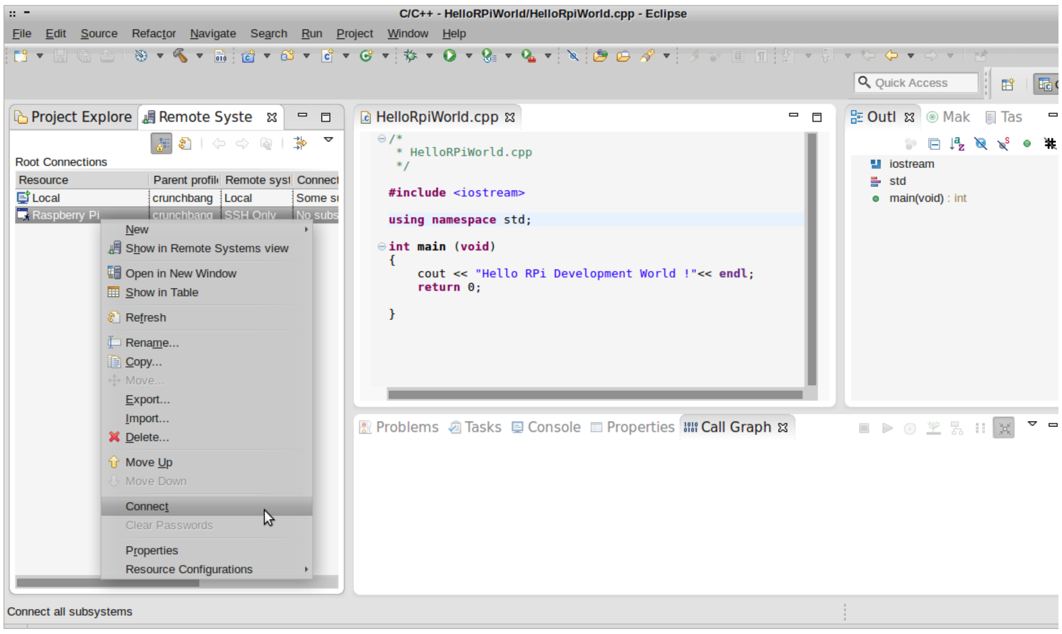
Description:

☑ Verify host name

Configure proxy settings

Figure 9. Remote SSH only system connection

- At this point a second device (Raspberry Pi) shows up in the “Remote Systems Detail” tab (Figure 10). Right click on the Raspberry Pi resource and click connect. You will be prompted to enter the username and password. Make sure that you utilize “username:pi” and “password:raspberry” (if you haven’t changed the default password). You may get a warning windows asking you if you are sure that you want to accept this connection. Affirm that you want to proceed with the connection. At this point you should be connected to your RPi from Eclipse.



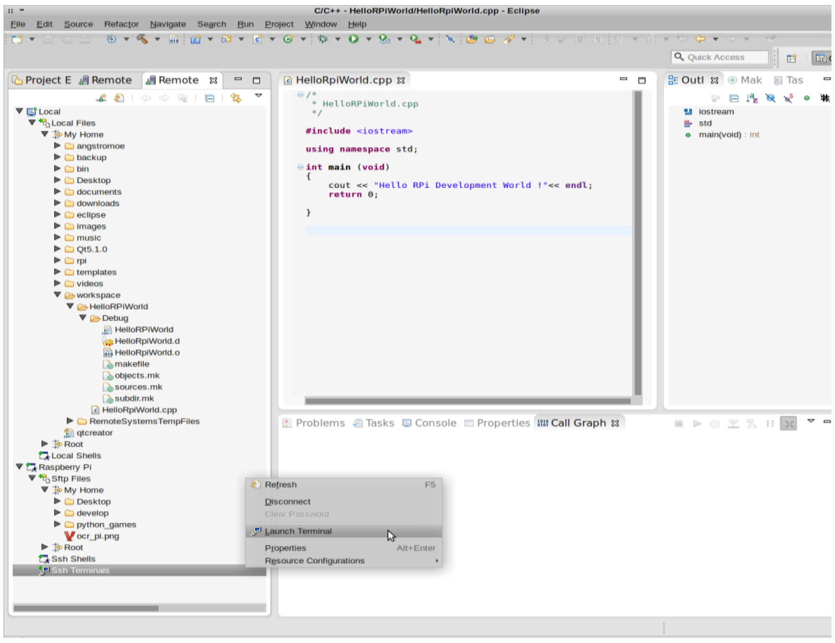
Resource	Parent profil	Remote syst	Connect
Local	crunchbang	Local	Some s
Raspberry Pi	crunchbang	SSH Only	No subs

```
/*
 * HelloRPiWorld.cpp
 */

#include <iostream>

using namespace std;

int main (void)
{
    cout << "Hello RPi Development World !"<< endl;
    return 0;
}
```

Connect all subsystems

Figure 10. Starting an SSH terminal in Eclipse

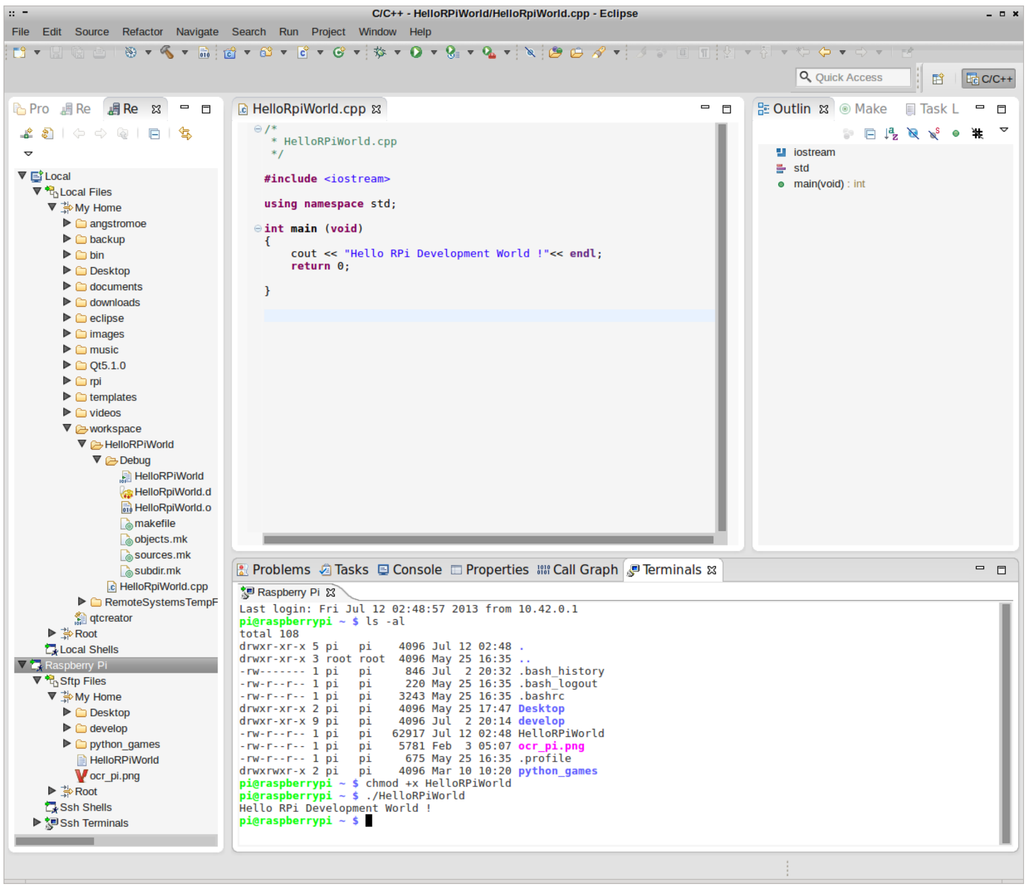
- Right click on the Raspberry Pi resource again in the “Remote Systems Detail” tab and click on the “Show in Remote Systems View”. This will open a “Remote Systems” tab in the same region of the workspace. In this tab, one can navigate the root file systems of both the Local Linux OS and that of the Raspbian/Raspberry Pi.
- Locate the HelloWorld Binary file on our local desktop PC and drag it to the home directory on the Raspberry Pi (You could also right click on the binary and click on “Copy” and then right click on the home folder in the Raspberry Pi and click “Paste”. This effectively copies the binary file to the home directory of the Raspberry Pi.



```
/*
 * HelloRPiWorld.cpp
 */

#include <iostream>

using namespace std;

int main (void)
{
    cout << "Hello RPi Development World !"<< endl;
    return 0;
}
```

Figure 11. Launch Terminal

- Now right click on the SSH terminal icon under the Raspberry Pi icon (Figure 11) in the "Remote Systems" and select "Launch Terminal" . This should launch a terminal window in the bottom of the Eclipse workspace (Figure 12). This is basically a console / terminal window connected to the Raspberry Pi via SSH.



```
/*
 * HelloRPiWorld.cpp
 */

#include <iostream>

using namespace std;

int main (void)
{
    cout << "Hello RPi Development World !"<< endl;
    return 0;

}
```

```
Last login: Fri Jul 12 02:48:57 2013 from 10.42.0.1
pi@raspberrypi ~ $ ls -al
total 108
drwxr-xr-x 5 pi   pi    4096 Jul 12 02:48 .
drwxr-xr-x 3 root root  4096 May 25 16:35 ..
-rw------- 1 pi   pi     846 Jul  2 20:32 .bash_history
-rw-r--r-- 1 pi   pi     220 May 25 16:35 .bash_logout
-rw-r--r-- 1 pi   pi    3243 May 25 16:35 .bashrc
drwxr-xr-x 2 pi   pi    4096 May 25 17:47 Desktop
drwxr-xr-x 9 pi   pi    4096 Jul  2 20:14 develop
-rw-r--r-- 1 pi   pi   62917 Jul 12 02:48 HelloRPiWorld
-rw-r--r-- 1 pi   pi    5781 Feb  3 05:07 ocr_pi.png
-rw-r--r-- 1 pi   pi     675 May 25 16:35 .profile
drwxrwxr-x 2 pi   pi    4096 Mar 10 10:20 python_games
pi@raspberrypi ~ $ chmod +x HelloRPiWorld
pi@raspberrypi ~ $ ./HelloRPiWorld
Hello RPi Development World !
pi@raspberrypi ~ $ 
```

Figure 12. Running the HelloRPiWorld binary on the Raspberry Pi via an eclipse's SSH terminal

IV)How to run the project

download project from :

-https://github.com/AnassTeemo/serverpi (for the server)

-https://github.com/AnassTeemo/ApiRaspberry (for the API)

With the makefile :
To compile the entire project :make
To compile the crossroads: make carrefour
To compile the chase : make chenillard

Launch the server (piserver)
Go to a browser launch page : localhost:9090/webfiles/manipulation.html