

phyBOARD WEGA-AM335x Single Board Computer

Application Development User Manual



Product No	: PCL-051/PBA-CD-02
SOM PCB No	: 1397.0
CB PCB No	: 1405.0
Edition	: Feb 26, 2014

In this manual copyrighted products are not explicitly indicated. The absence of the trademark (™) and copyright (©) symbols does not imply that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

The information in this document has been carefully checked and is believed to be entirely reliable. However, PHYTEC Embedded Pvt. Ltd. assumes no responsibility for any inaccuracies. PHYTEC Embedded Pvt. Ltd. neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Embedded Pvt. Ltd. reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages that might result.

Additionally, PHYTEC Embedded Pvt. Ltd. offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Embedded Pvt. Ltd. further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

© Copyright 2014 PHYTEC Embedded Pvt. Ltd, Koramangala Bangalore.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may be made without the explicit written consent from PHYTEC Embedded Pvt. Ltd.

	India	Europe	North America
Address:	PHYTEC Embedded Pvt. Ltd. # 16/9C, 3rd Floor, 3rd Main, 8th Block, Opp: Police Station, Koramangala, Bangalore -560095 INDIA	PHYTEC Technologie Holding AG Robert-Koch-Str. 39 55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+91-80-40867046 Sales@phytec.in	+49 (800) 0749832 order@phytec.de	1 (800) 278-9913 sales@phytec.com
Web Site:	http://www.phytec.in	http://www.phytec.de	http://www.phytec.com

Table of Contents

Introduction.....	4
1. Application development using Eclipse IDE.....	4
1.1. Eclipse IDE Installation.....	4
1.2. Eclipse IDE Configuration for phyBOARD-WEGA-AM335x.....	5
1.2.1. Host Setup.....	5
For Linux:.....	5
For Windows:.....	5
1.2.2. Target Setup.....	6
1.3. Creating a New Project in Eclipse.....	6
1.3.1. How to open eclipse.....	6
For Linux:.....	6
For windows:-.....	7
1.3.2 Creating a New Project.....	8
1.3.3. Open new C source file.....	10
1.3.4. Write simple Hello Application.....	11
1.3.5. Build the project.....	13
1.4. Changing the Demo Application.....	14
1.4.1. Open Target Board using Minicom.....	16
1.5. Remote System Access using Eclipse.....	16
For Windows :.....	16
For Linux :.....	16
1.5.1. Create New Connection for Remote System login.....	18
1.5.2. Set the Host Name and IP.....	18
1.6. Debugging an example project.....	23
1.6.1. Starting the GDB server on the target.....	23
1.6.2. Configuring and starting the debugger in Eclipse.....	24
1.6.3. Setting a Breakpoint	27
1.6.4. Stepping and Watching Variable Contents.....	28
1.6.5. Stepping and Watching Variable Contents.....	30
1.6.6. Using the Memory Monitor.....	31

Introduction

This Reference Manual describes the phyBOARD-WEGA-AM335x for application development. First chapter describes the installation of eclipse and how to develop an application on phyBOARD-WEGA-AM335x using Eclipse IDE. Second chapter describes about how to write an application using console terminal. After completing this manual you will come to know how to use the Eclipse.

1. Application development using Eclipse IDE

During this chapter you will learn how to build your own C/C++ applications for the target with the help of Eclipse. We will start developing our own applications with the help of Eclipse. First we will take a look on the C programming language. At the end of this chapter we will explain how to execute your written programs automatically when booting the target.

1.1. Eclipse IDE Installation

Download the Eclipse IDE from the below links (Note: According to your system configuration) and install.

For Linux:

- Install java using below command:

```
$ sudo apt-get install openjdk-7-jdk openjdk-7-jre
```

- Download eclipse from below link:

<http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/junosr2>

For windows:

- Download and install the java for windows using below link:

<http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

- Download eclipse from below link:

<http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/junosr2>

1.2. Eclipse IDE Configuration for phyBOARD-WEGA-AM335x

1.2.1. Host Setup

Toolchain: For Compiling the Application we need the toolchain which you can easily download from the below link.

For Linux:

ftp://ftp.phytec.de/pub/Products/India/phyBOARD_WEGA-AM335x/Linux/PD14.0.0/tools/toolchain/arm-cortexa8-linux-gnueabi.tar.bz2

- Set ip Address

```
$ ifconfig eth0 192.168.1.12 up
```

- Set the gateway

```
# route add default gw 192.168.1.1
```

For Windows:

http://sourcery.mentor.com/public/gnu_toolchain/arm-none-linux-gnueabi/arm-2012.09-64-arm-none-linux-gnueabi.exe

- Click Start ► Control Panel ► open Network and Sharing Center
- From the Tasks menu on the left, choose Manage Network Connections
- Find and Right click on the active Local Area Connection and choose Properties
- Double-click on Internet Protocol Version 4 (TCP/IPv4)
- Click on Use the following IP address
- Enter a IP like 192.168.1.12
- Press Tab and the Subnet Mask section will populate with default numbers
- Enter gateway 192.168.1.1
- Hit Ok.

1.2.2. Target Setup

Connect the power adaptor, serial cable, ethernet cable to the phyBOARD-WEGA-AM335x Board & Boot the Board.

To see all the Communication interfaces present on the phyBOARD-WEGA-AM335x Board

```
# ifconfig -a
```

- Configure eth0.

```
# ifconfig eth0 192.168.1.11 up
```

where eth0 is the LAN interface.

Check whether eth0 is configured or not by using the below command.

```
# ifconfig -a
```

Note:

192.168.1.11 is not mandatory you can use any IP but it should be different from the server IP.

- Set the gateway

```
# route add default gw 192.168.1.1
```

To see the change in the gateway.

```
# route
```

1.3. Creating a New Project in Eclipse

In this section you will learn how to create a new project with Eclipse and how to configure the project for use with the GNU - C/C++ cross development toolchain. Click the Eclipse icon to start the application. You can find this icon where you have extracted the Eclipse IDE for C/C++ Developers.

1.3.1. How to open eclipse

For Linux:

- Go to the Location where you have downloaded eclipse, Extract it and run binary file `./eclipse`
- Confirm the [workspace directory](#) with OK

For windows:-

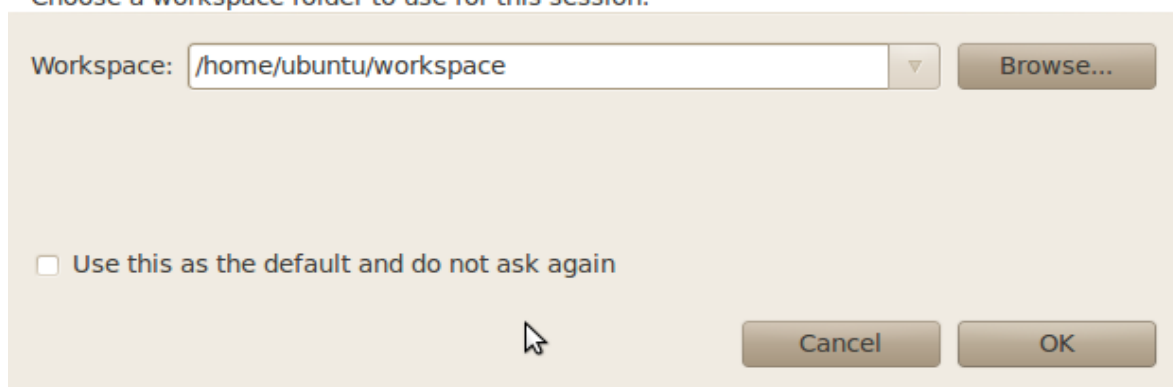
- Click the Eclipse icon to start the application. You can find this icon on your desktop.



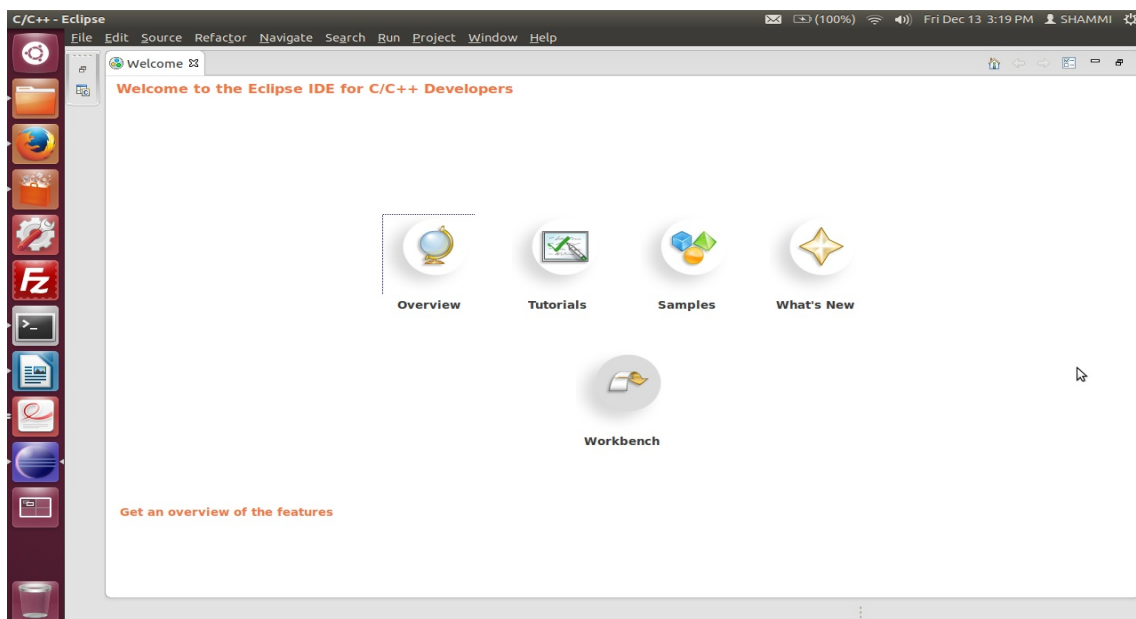
- Confirm the workspace directory with OK

Select a workspace

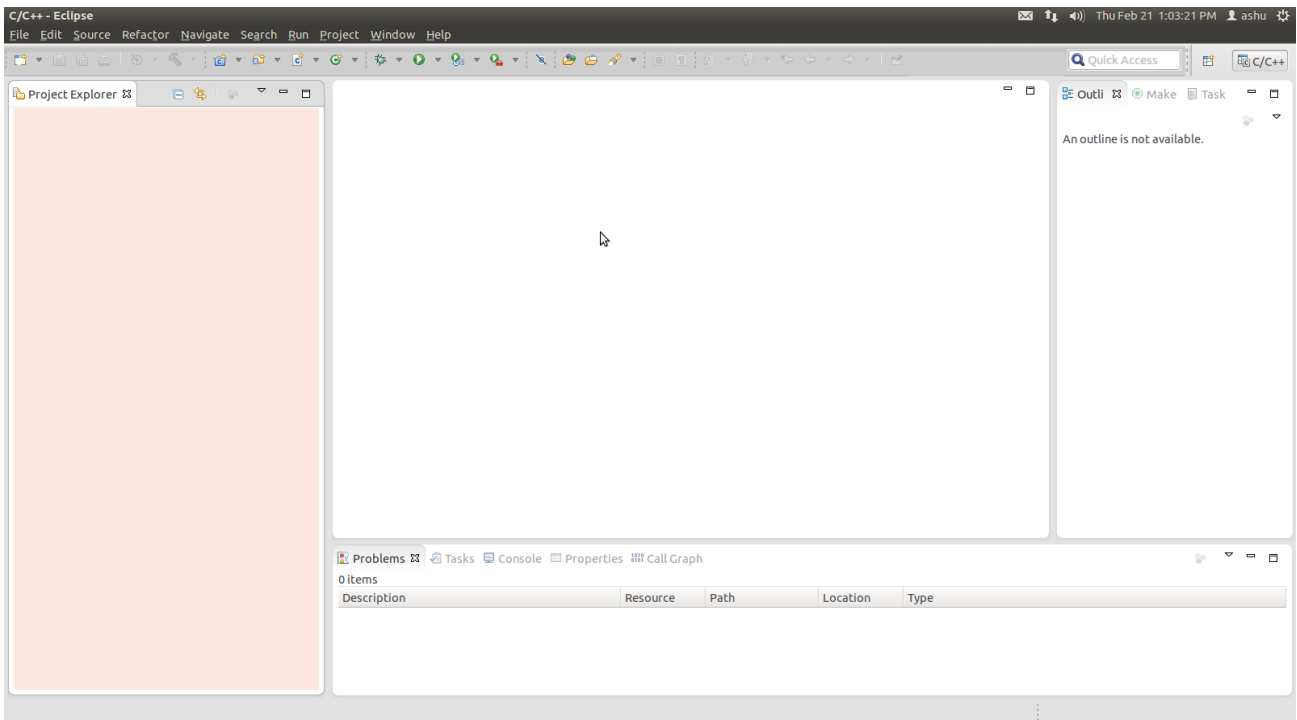
Eclipse SDK stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.



- Close the "Welcome to Eclipse" screen by clicking on the "workbench" button



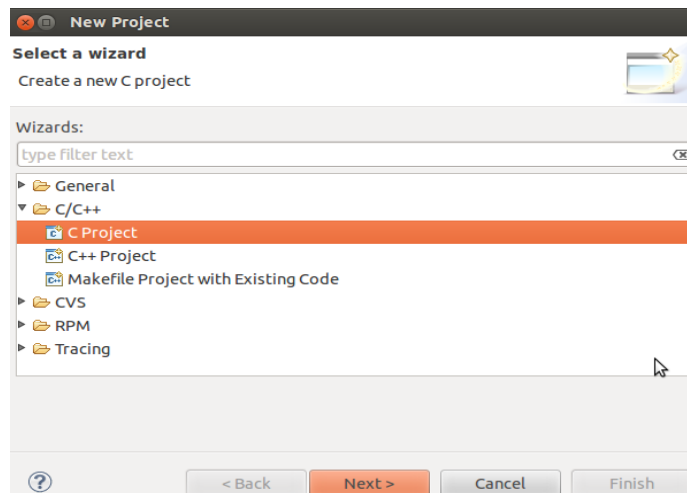
Now you can see the Eclipse Workbench as below:



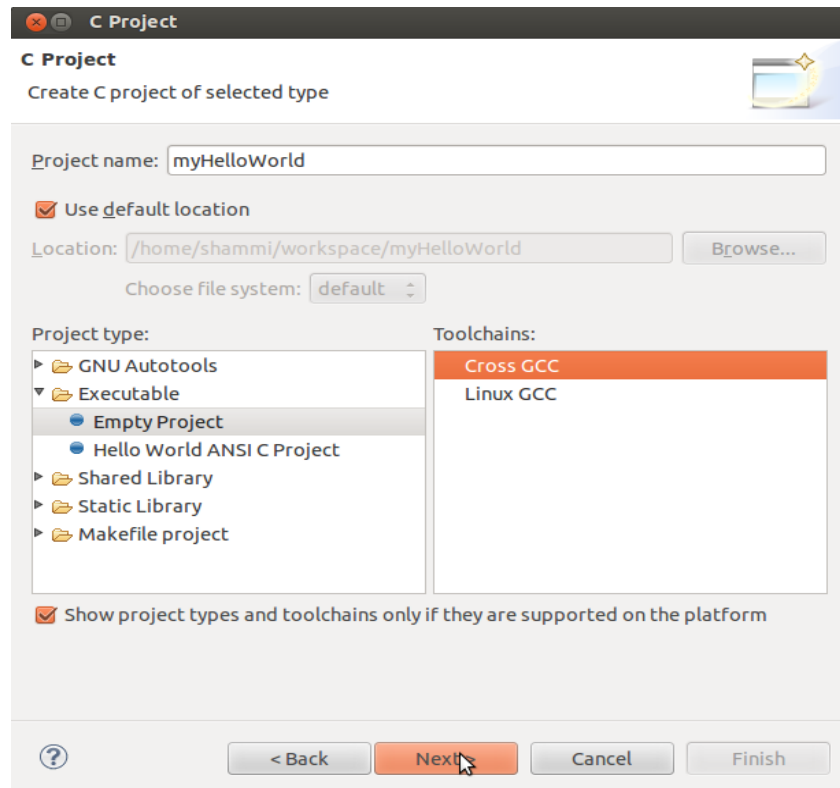
1.3.2 Creating a New Project

In this section we will learn how to create a new project with Eclipse and how to configure the project for use with the GNU - C/C++ cross development toolchain.

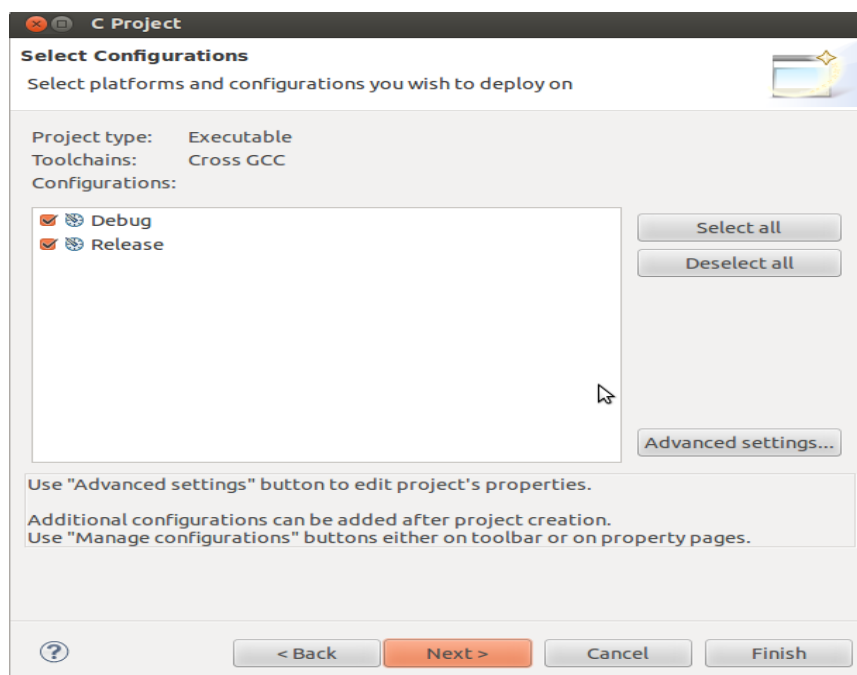
- Select **File** ► **New** ► **Project** from the menu bar. A new dialog will open.
- Select **C Project** and click **Next**



- Enter the project name **myHelloWorld** and Toolchain as **Cross GCC** then click **Next**

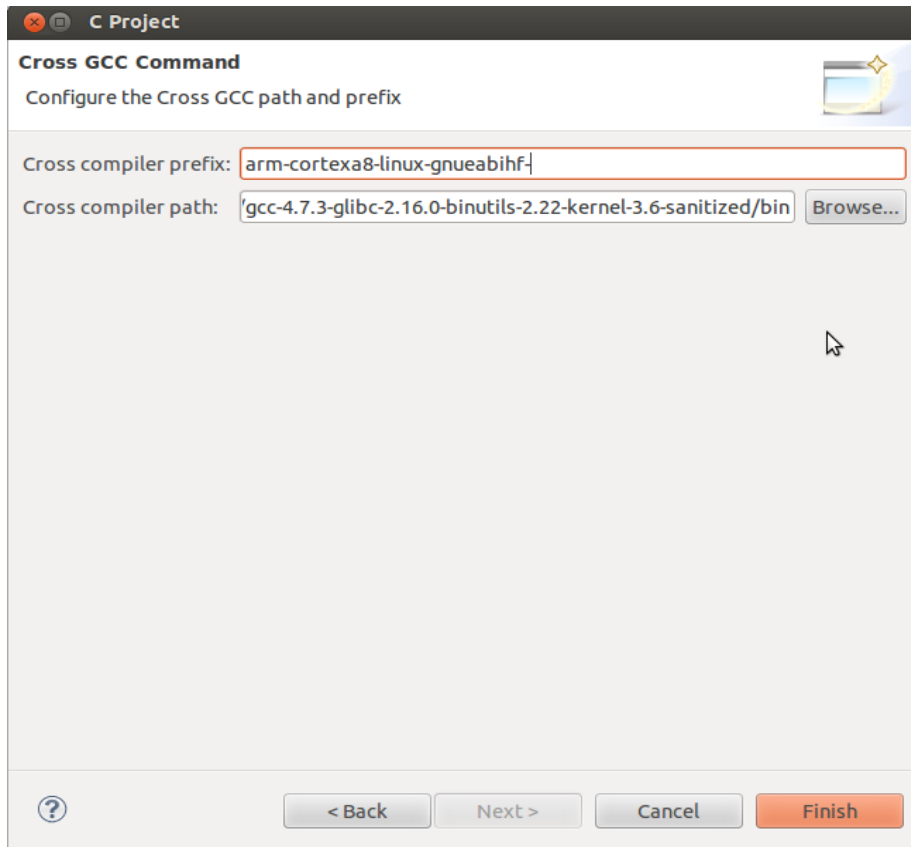


- Click **Next**



- **Set Toolchain Prefix & Path and Click **Finish****

Select the **Cross Compiler Prefix** as ***arm-cortexa8-linux-gnueabihf-*** and **Cross Compiler Path** as <path of toolchain bin>



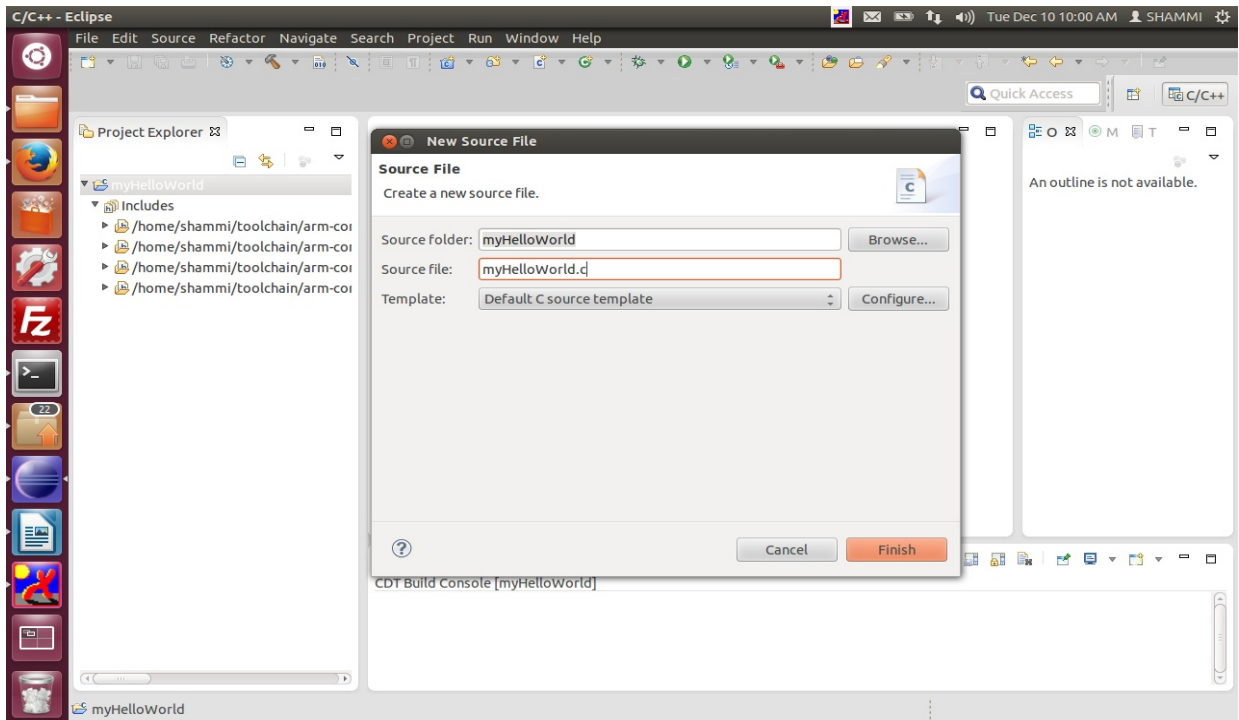
Note

For windows you have to select the arm-none-linux-gnueabi- and the appropriate path of the toolchain.

1.3.3. Open new C source file

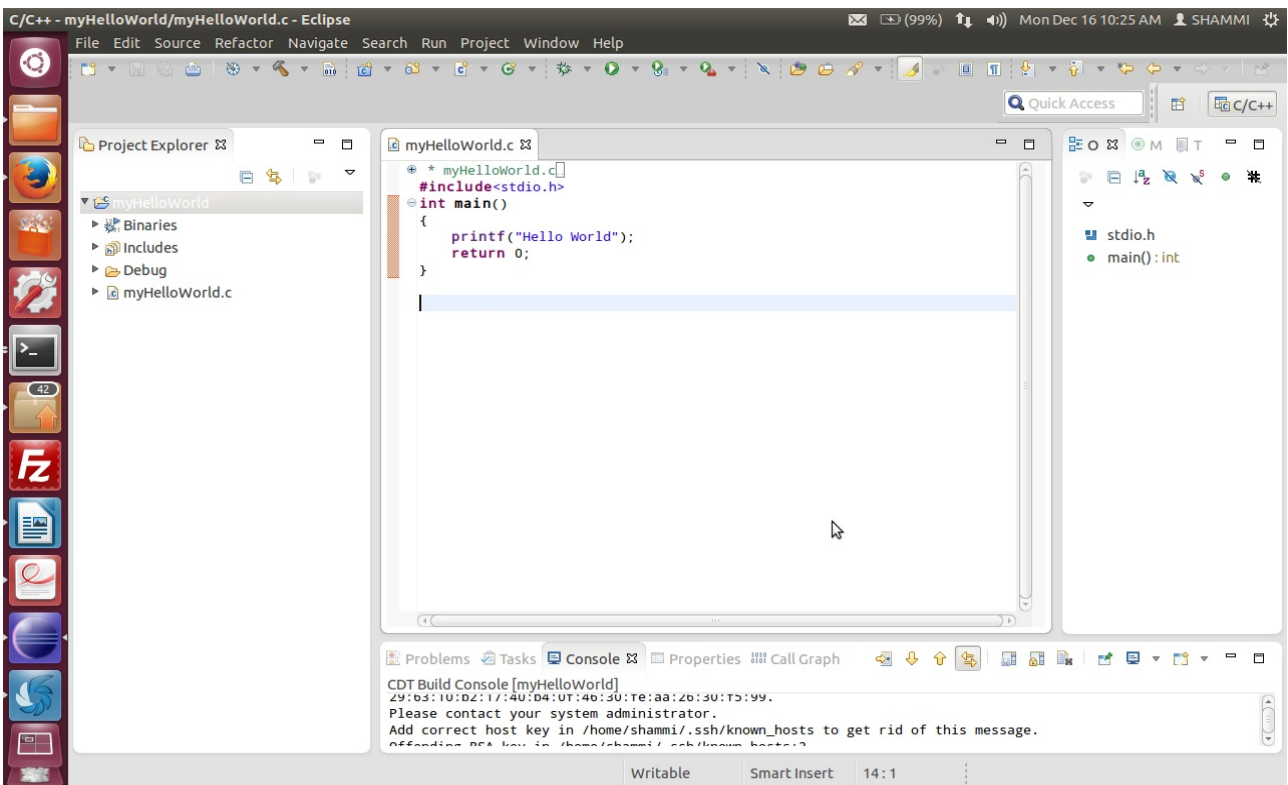
- Right-click on **myHelloWorld** project
- Select **File ► New ► Source file** from the menu bar

- In Source file write `myHelloWorld.c` and click on **Finish**.



1.3.4. Write simple Hello Application

Write a simple Hello Application in C.



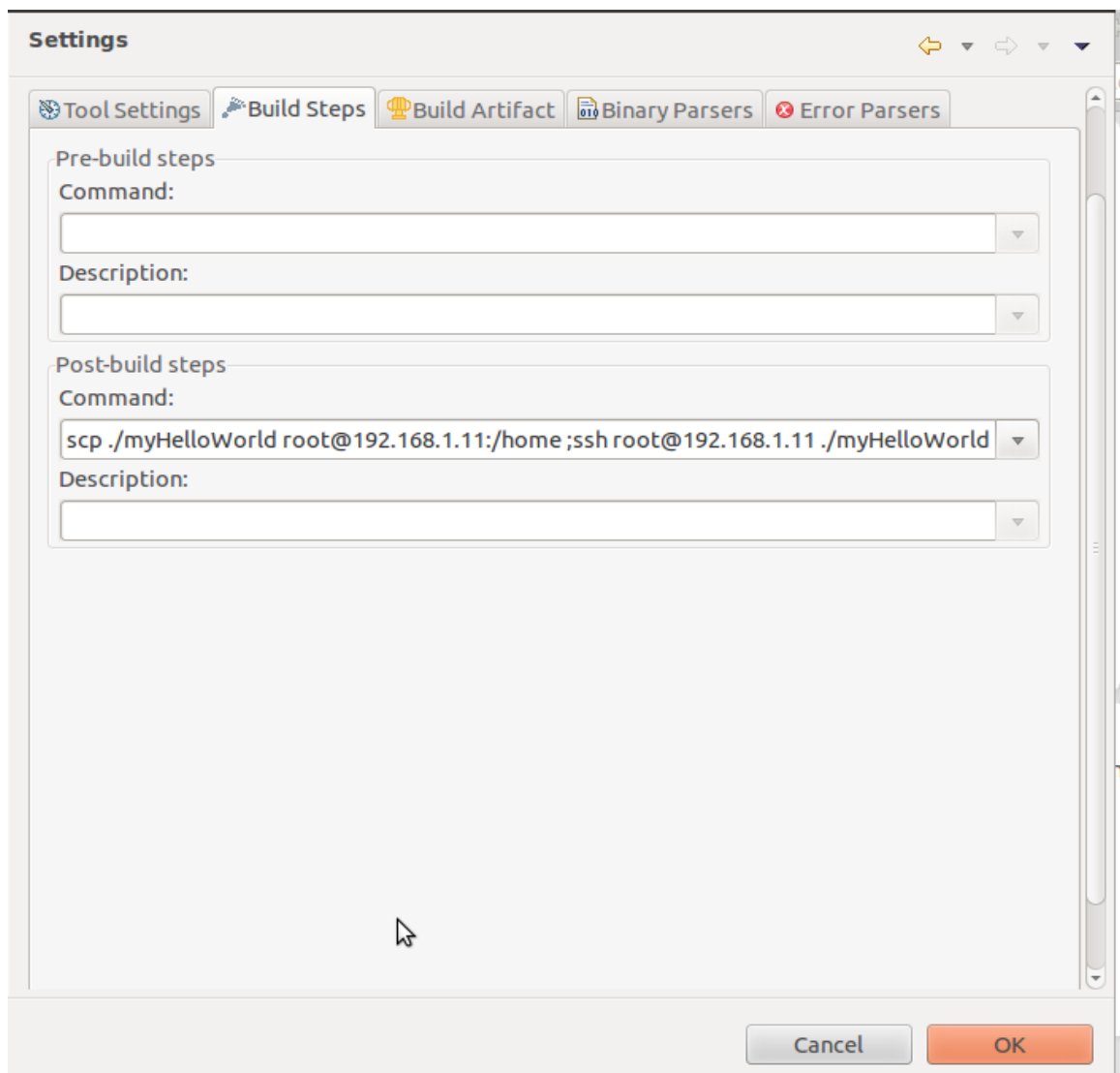
To compile your project for the phyBOARD-WEGA-AM335x instead, you will have to use the [GNU - C/C++ cross compiler](#).

Modify Post build steps:

- Right-click the [myHelloWorld](#) project and chose [Properties](#). The Properties dialog appears.
- Select [C/C++ Build](#) ► [Setting](#) ► Select the [Build Steps](#) tab

Enter the following command in the Post-build steps [Command](#) input field:

```
scp ./myHelloWorld root@192.168.1.11:/home ;ssh  
root@192.168.1.11 ./myHelloWorld
```



Note

First login manually using **ssh** as shown in snapshot below. Otherwise we will not be able to login in target because it requires secure connection.

```
naveen@linux:~$ ssh root@192.168.1.11
The authenticity of host '192.168.1.11 (192.168.1.11)' can't be established.
RSA key fingerprint is c8:1a:3c:2c:95:75:16:78:d4:b3:0e:88:42:2b:b1:e5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.11' (RSA) to the list of known hosts.
root@phyBOARD-WEGA-AM335x:~
```

Note

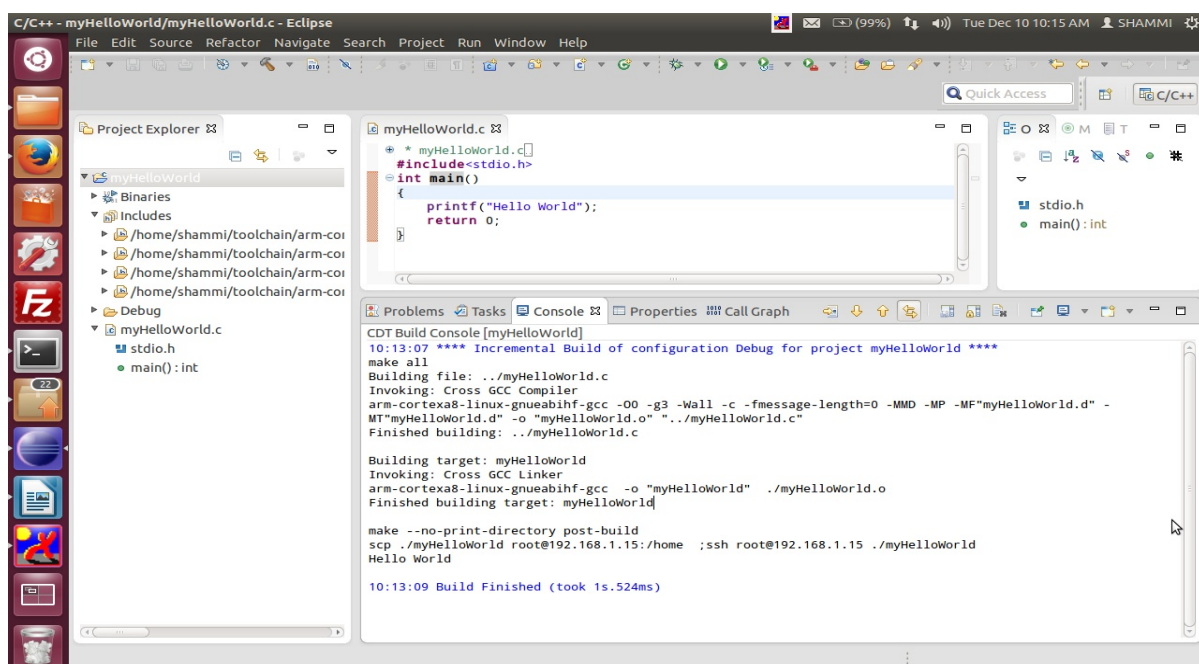
If you are using windows so you have to use **Winscp** (or) directly copy the binary into target board using pen-drive or sd-card.

- Click Apply
- Click OK

1.3.5. Build the project

- Select **Project** ► **Build project** from the menu bar
The project will be built.
- Select the **Console** tab.

if no errors occur while building the project, you will see the following output:



```
C/C++ - myHelloWorld/myHelloWorld.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
myHelloWorld
  Binaries
  Includes
  /home/shammi/toolchain/arm-coi
  /home/shammi/toolchain/arm-coi
  /home/shammi/toolchain/arm-coi
  /home/shammi/toolchain/arm-coi
  Debug
  myHelloWorld.c
  stdio.h
  main():int
myHelloWorld.c
#include <stdio.h>
int main()
{
    printf("Hello World");
    return 0;
}
Problems Tasks Console Properties Call Graph
CDT Build Console [myHelloWorld]
10:13:07 **** Incremental Build of configuration Debug for project myHelloWorld ****
make all
Building file: ../myHelloWorld.c
Invoking: Cross GCC Compiler
arm-cortexa8-linux-gnueabi-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"myHelloWorld.d" -
MT"myHelloWorld.o" -o "myHelloWorld.o" "../myHelloWorld.c"
Finished building: ../myHelloWorld.c

Building target: myHelloWorld
Invoking: Cross GCC Linker
arm-cortexa8-linux-gnueabi-gcc -o "myHelloWorld" ../myHelloWorld.o
Finished building target: myHelloWorld

make --no-print-directory post-build
scp ./myHelloWorld root@192.168.1.15:/home ; ssh root@192.168.1.15 ./myHelloWorld
Hello World

10:13:09 Build Finished (took 1s.524ms)
```

Note

If you are using Window machine then you need the make utils using the below link.

<ftp://ftp.equation.com/make/32/make.exe>

1.4. Changing the Demo Application

Now we will extend the *myHelloWorld* application. The extended *myHelloWorld* application will write an output to the first serial interface as well as to the standard output.

- Open Eclipse if it is not opened yet
- Double-click [myHelloWorld.c](#) in the myHelloWorld project

First include the following two additional header files:

```
#include <unistd.h>
#include <fcntl.h>
```

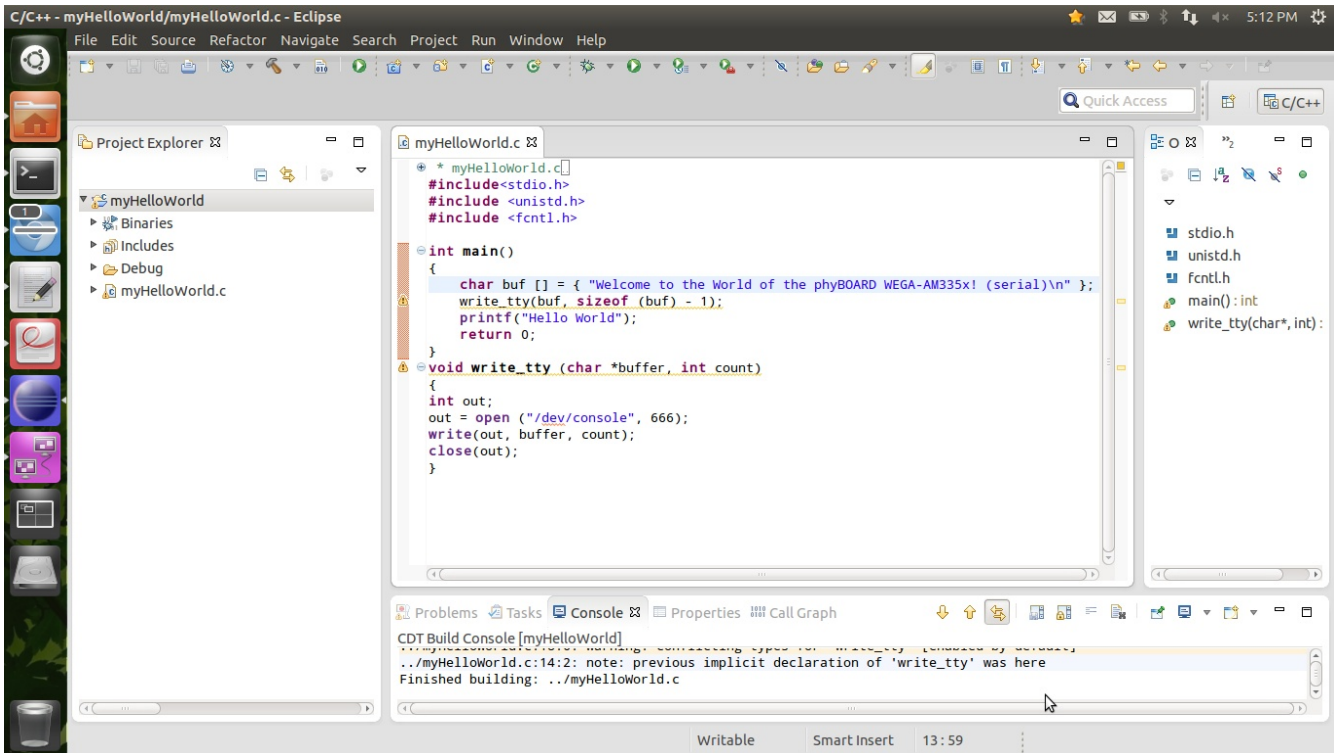
Then add the function `write_tty()`, which writes `n` bytes to the first serial interface (which, on the phyBOARD-WEGA-AM335x, is connected to the system console `/dev/console`):

```
void write_tty (char *buffer, int count)
{
    int out;
    out = open ("/dev/console", 666);
    write(out, buffer, count);
    close(out);
}
```

Enter the following two lines in the `main()` function to declare the buffer and call the `write_tty()` function.

```
char buf [] = { "Welcome to the World of the phyBOARD WEGA-AM335x! (serial)\n" };
write_tty(buf, sizeof (buf) - 1);
```

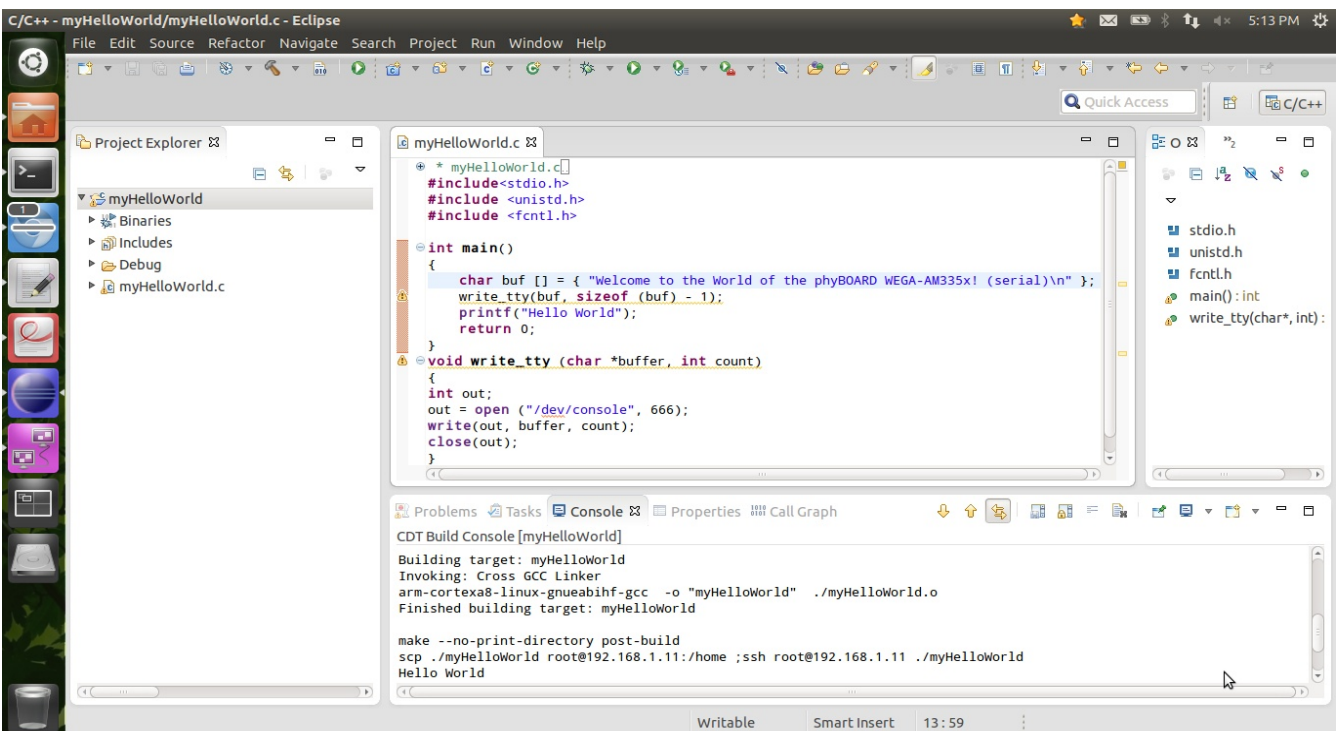
In the screenshot next page, you can see the complete program.



- Save your program after changing the code.
- Select Project ► Build project from the menu bar.

The project will be build...

The application will be compiled, build, copied to the target and executed.



1.4.1. Open Target Board using Minicom

- Open the terminal using minicom.
- Enter user name as `root` and press Enter then type `ls` to see all the file.

```
# ls
```

```
Type ./myHelloWorld to start the application
```

```
# ./myHelloWorld
```

```
Hello World
```

```
Welcome to the World of the phyBOARD WEGA-AM335x! (serial)
```

- `close` minicom.

In this section we had changed an existing application. We also learned how to access the serial interface.

First, we called the function `open()` on the device `/dev/console`. The return value of this function was a file descriptor, with the file descriptor you called the function `write()` to send `n` bytes to the device `/dev/console`.

After that, the file descriptor was closed with the function `close()`.

1.5. Remote System Access using Eclipse

For Windows :

You have to set the address manually
connect ethernet cable

Go to [network connections](#)

right click on "[Local area connection](#)" ► [properties](#) ► under [general tab](#)

double click on "[Internet Protocol\(TCP/IP\)](#)"

change the parameters.

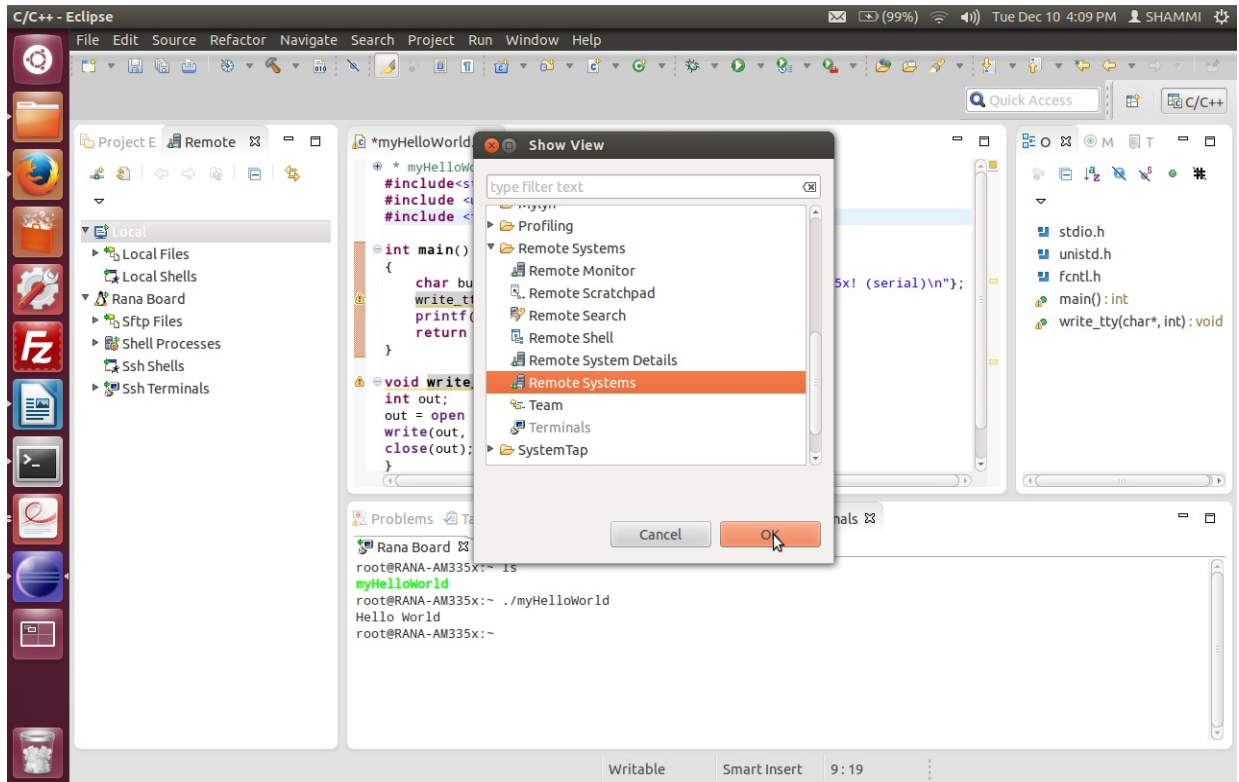
For above settings, refer section [1.2.1](#)

For Linux :

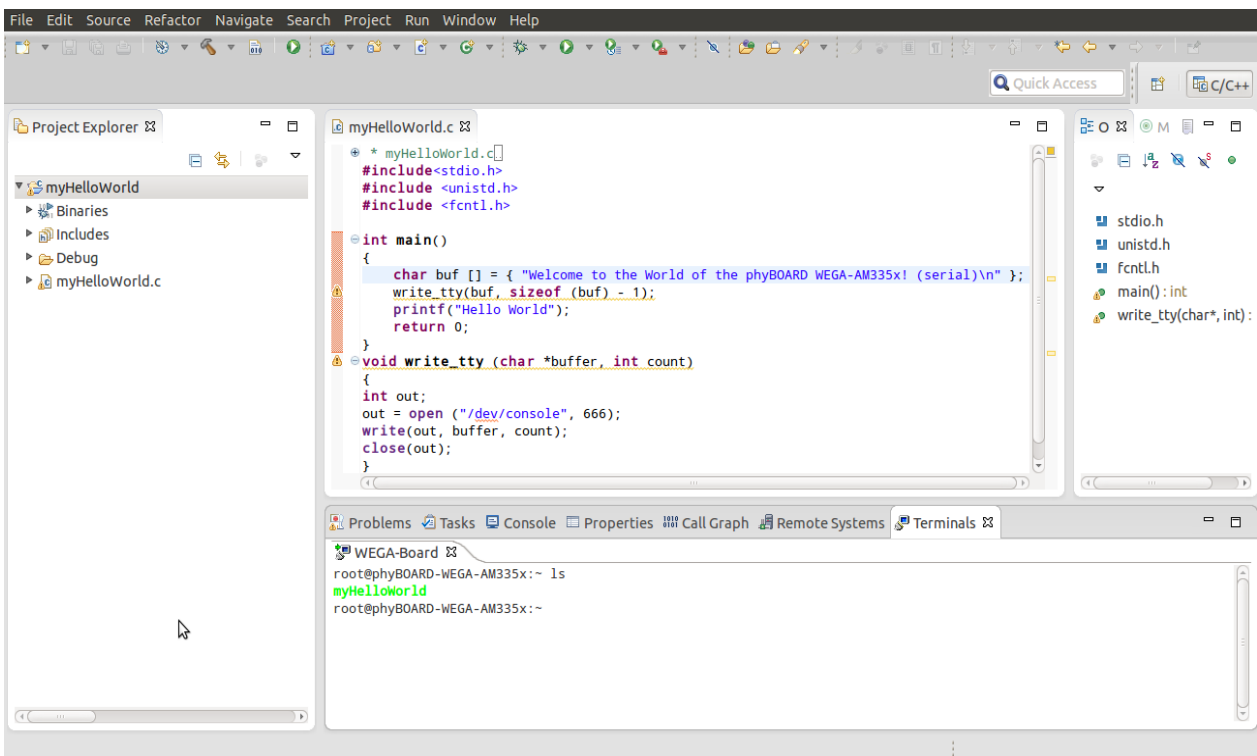
You have to set the address manually, for this refer section [1.2.1](#)

- Left-click the [Window](#) tab

[Show view](#) ► [other](#) ► [Remote Systems](#) and ok



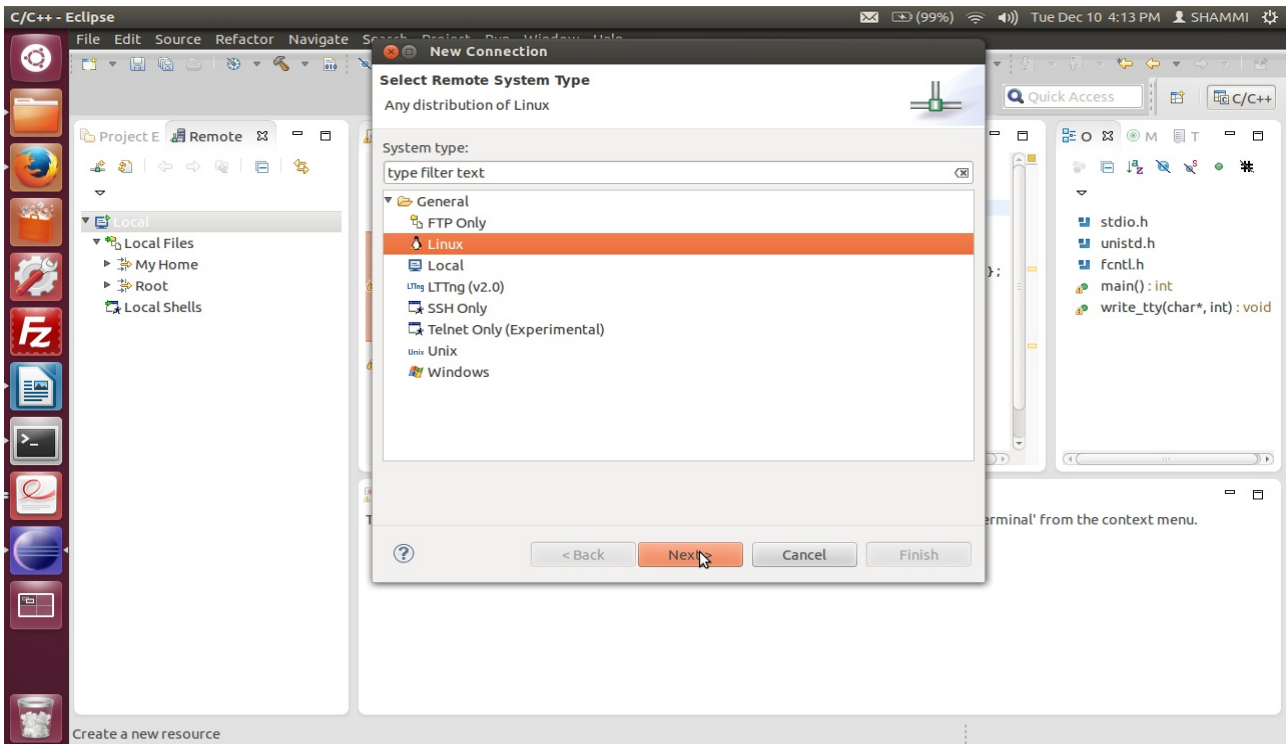
Now you are able to see the remote system page.



1.5.1. Create New Connection for Remote System login

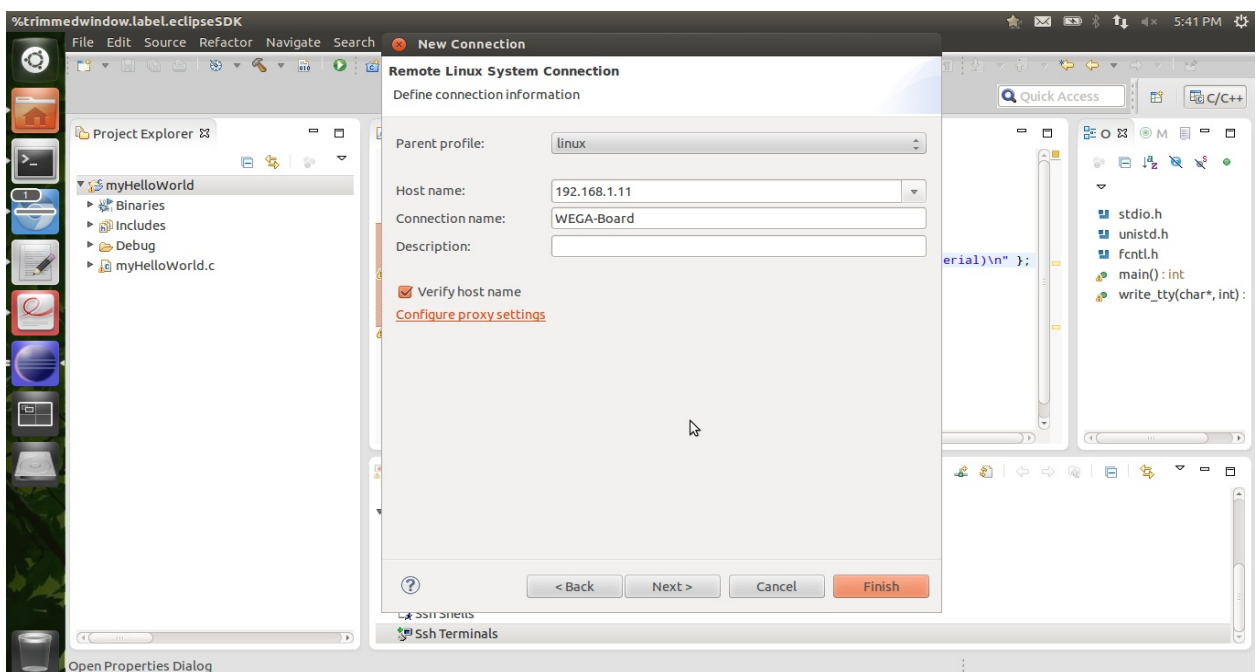
- Right Click on **Local** select new connection

select **linux**

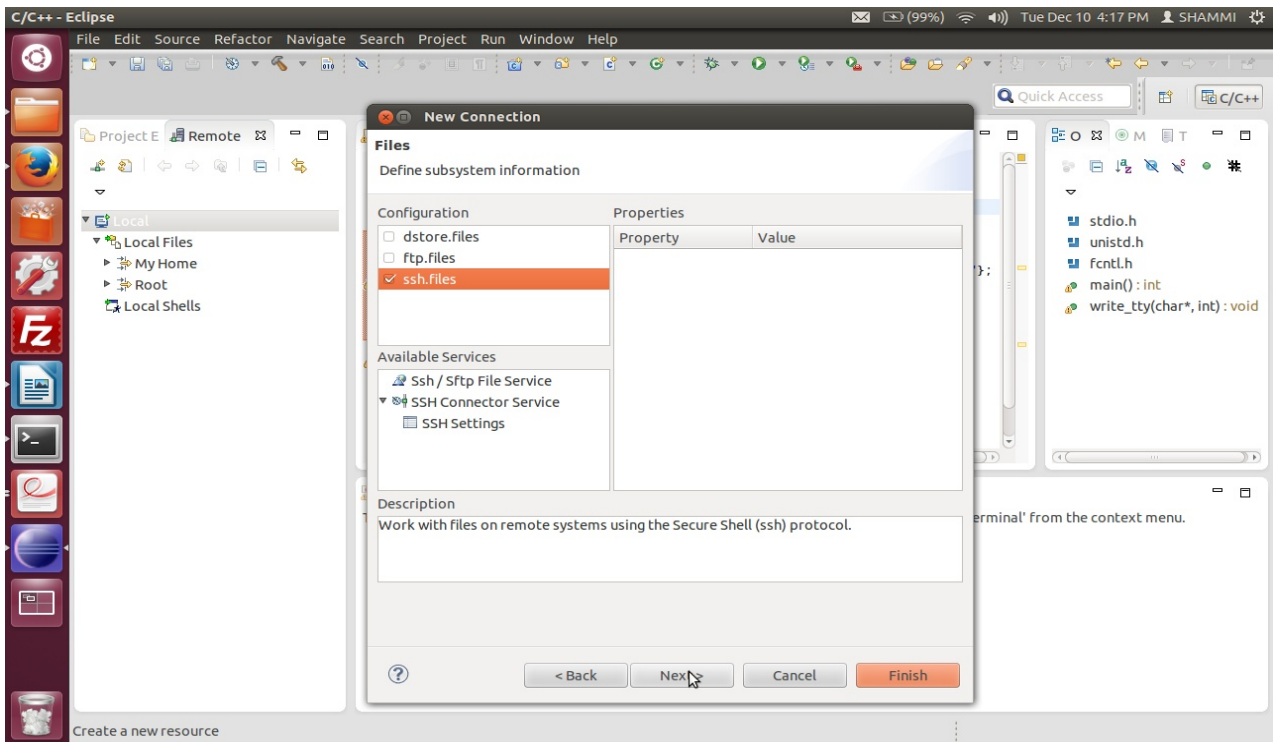


1.5.2. Set the Host Name and IP

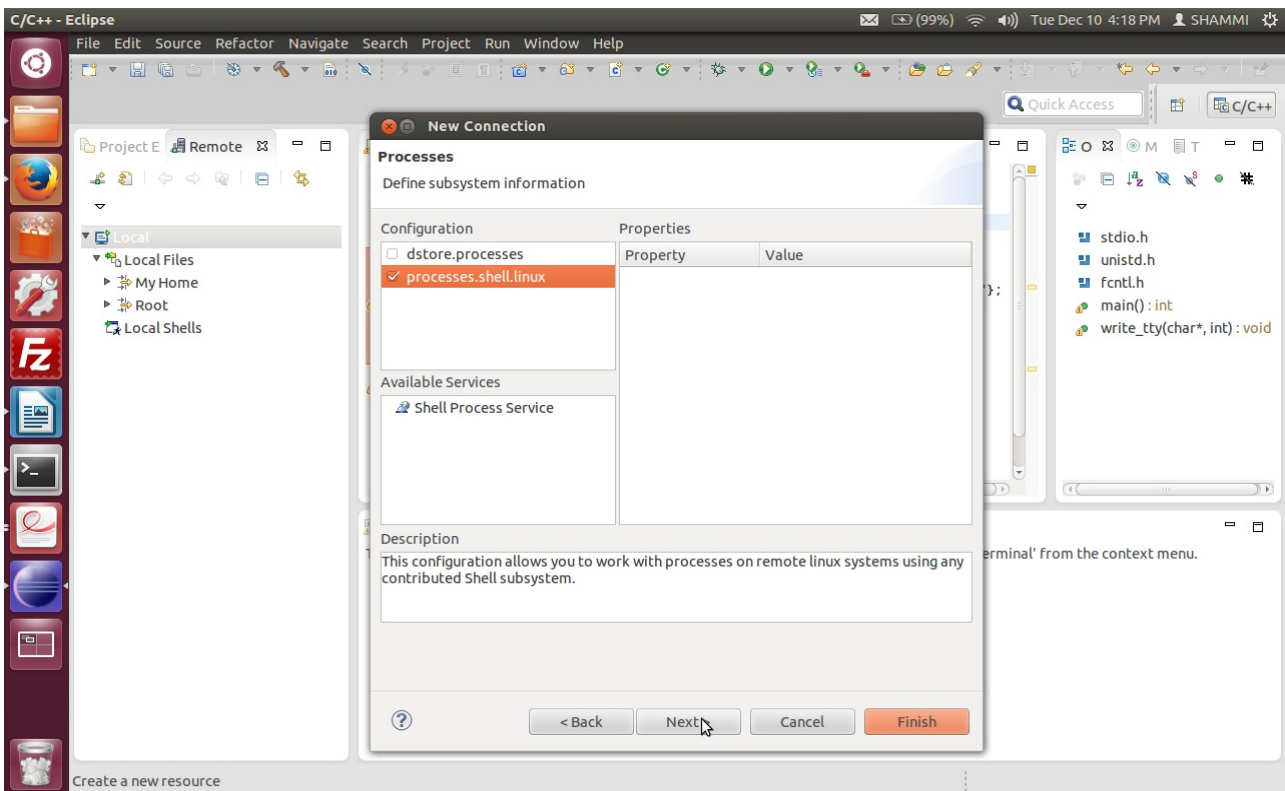
Then write **Host name** as 192.168.1.11 and **connection name** as WEGA-Board.



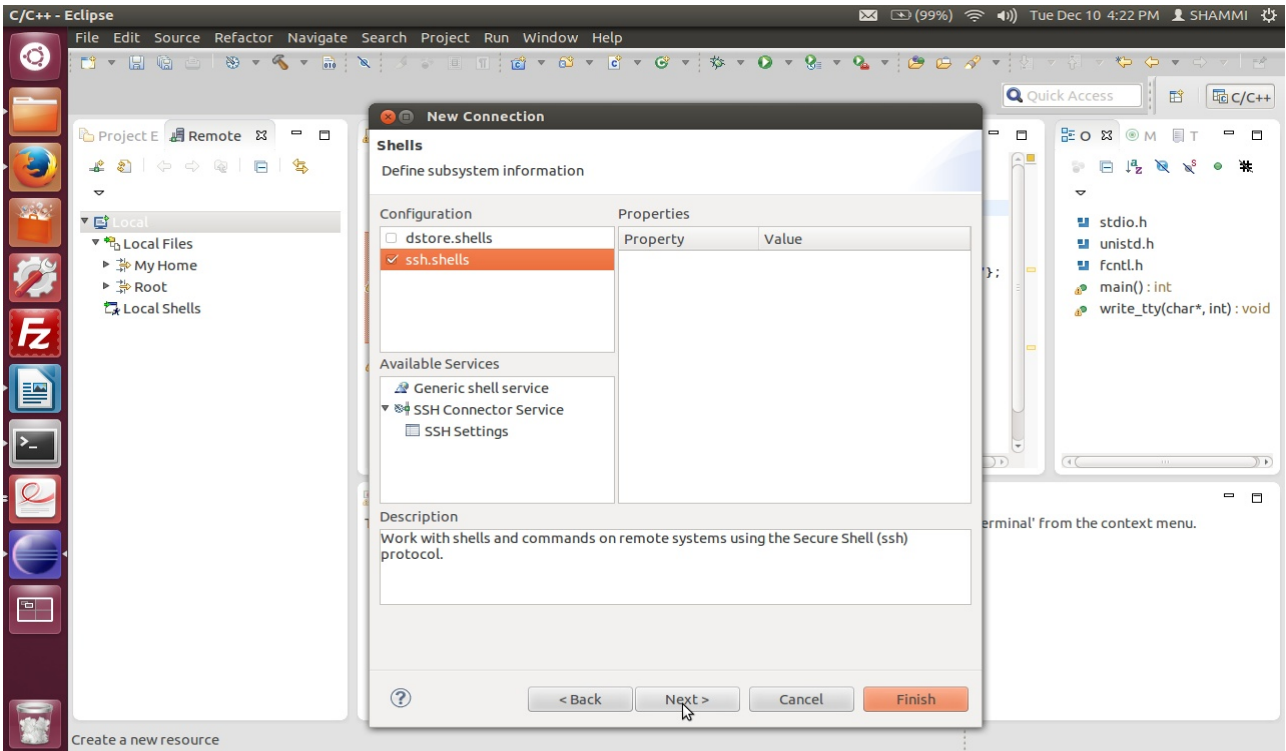
■ Select ssh.files



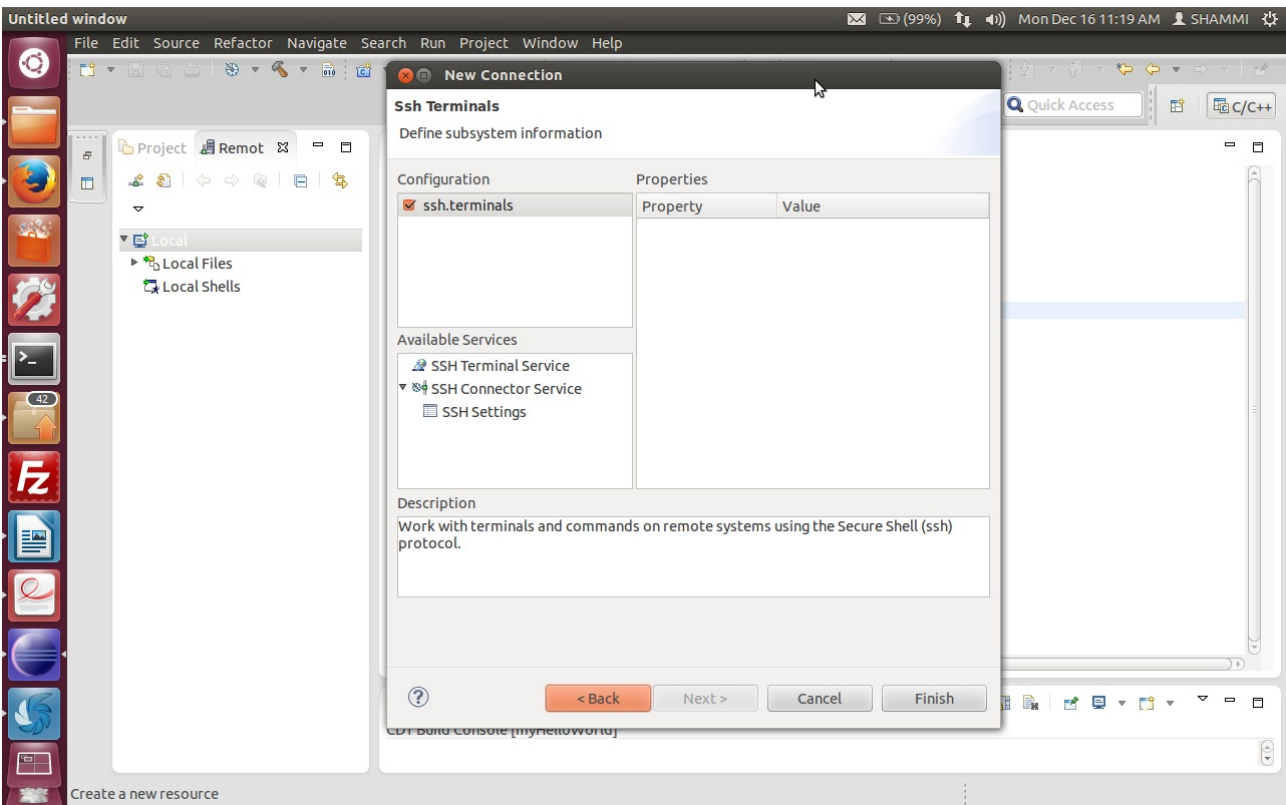
■ select processes.shell.linux and next



- select `ssh.shells` and next

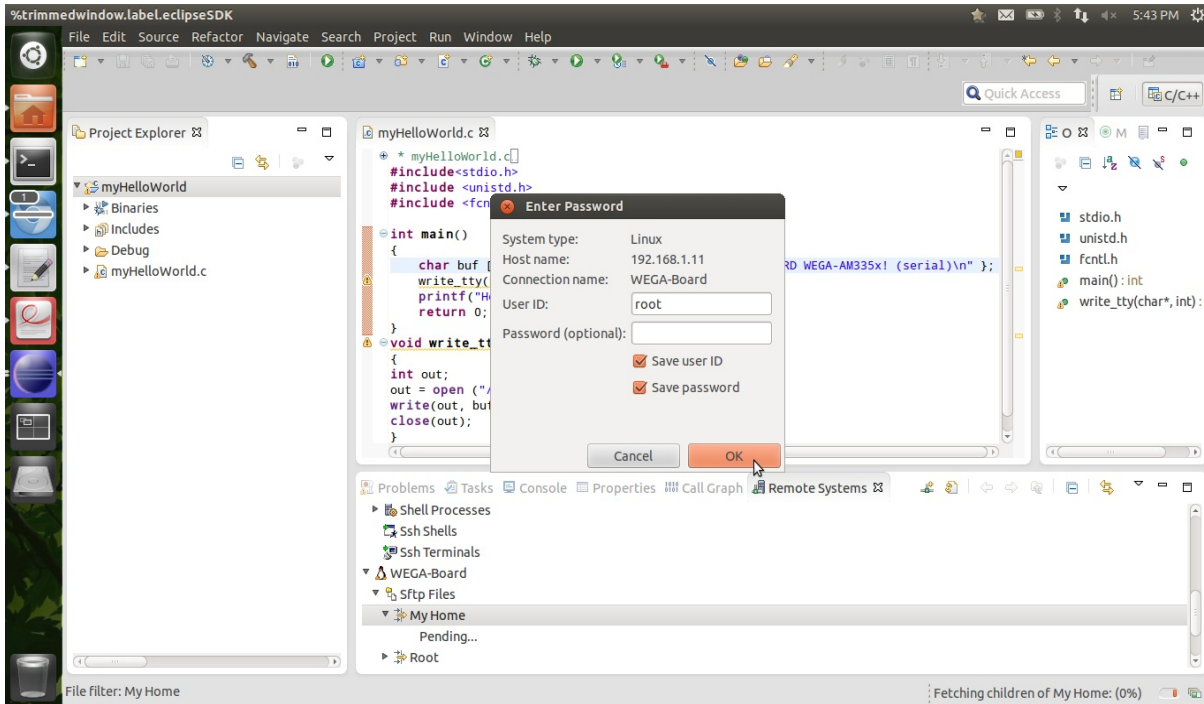


- select `ssh.terminals` and finish

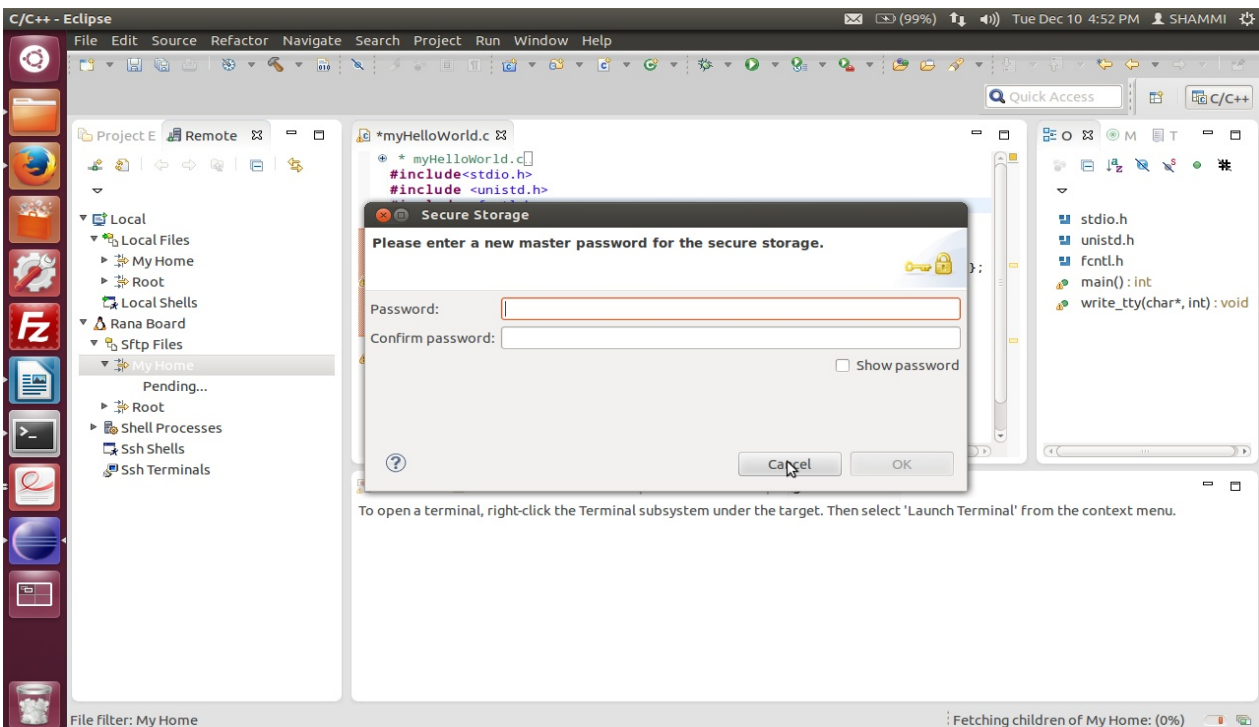


Now we successfully create the connection.

- Click on the **Wega-Board** ► **Sftp Files** ► **My Home**
- Type **User ID** as **root** leave **password blank**. Then press **OK**.



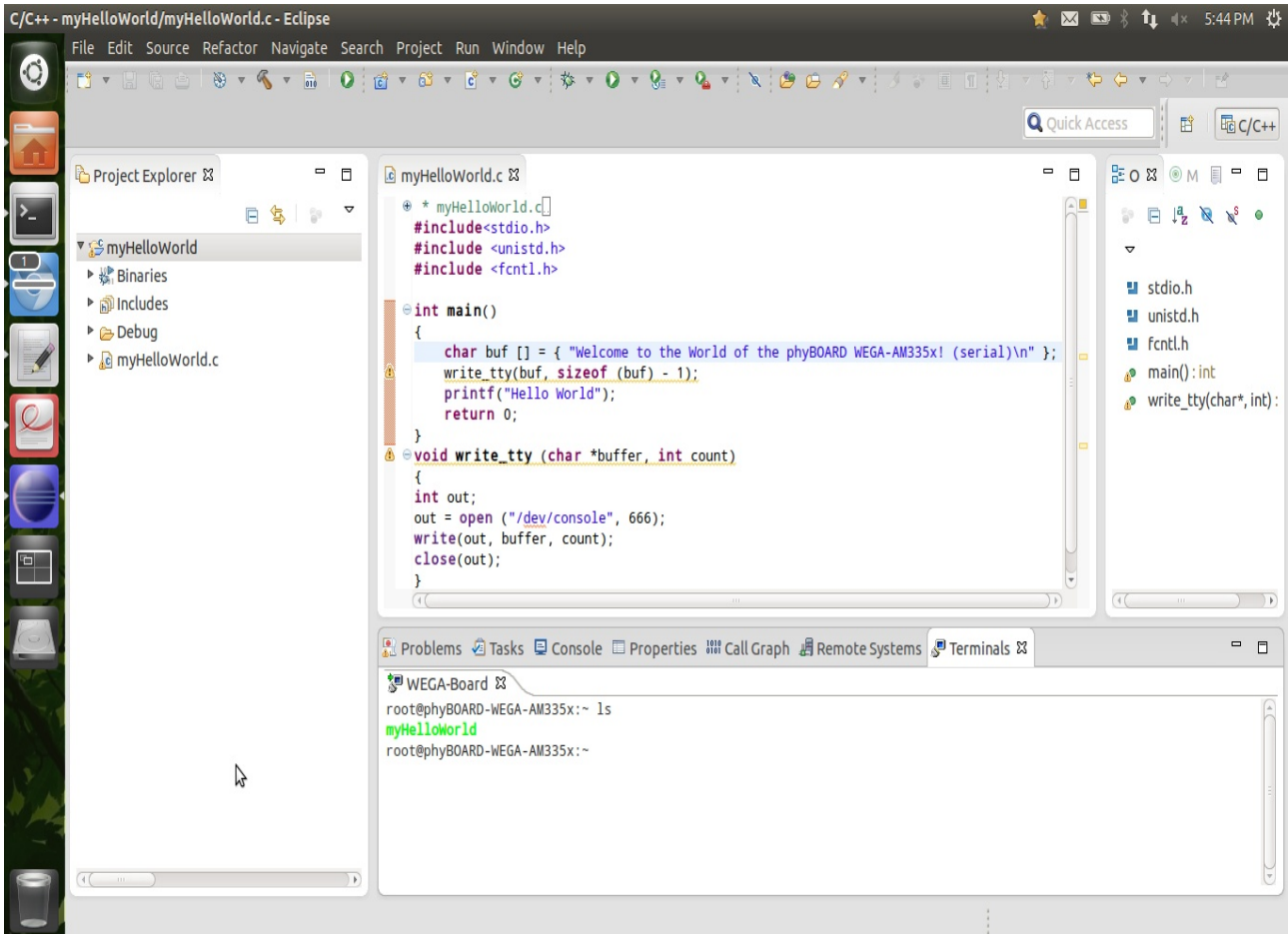
- Then a **secure Storage** tab is opened just cancel it.



1.5.3. Launch the Remote Terminal

- Right click ssh Terminal ► Launch Terminal

Now we can see all the contents of phyBOARD-WEGA-AM335x.



1.6. Debugging an example project

- In this section, we will learn how to use the GNU debugger i.e., GDB on the host for remote debugging in conjunction with the GDB server on the target.
- First, you will start the GDB server on the target. Then you will configure the Eclipse platform and start the GNU debugger out of Eclipse using the Debug view.
- The Debug view allows you to manage the debugging and running of a program in the workbench. Using the Debug view you will be able to set breakpoints/watchpoints in the code and trace variables and registers.
- The GDB client is running on the host and is used to control the GDB server on the target, which in turn controls the application running on the target.
- GDB client and GDB server can communicate over a TCP/IP network connection as well as via a serial interface.
In this Quickstart we will only describe debugging via TCP/IP.

1.6.1. Starting the GDB server on the target

In this passage you will learn how to start the GDB server on the target. The GDB server will be used to start and control the [myHelloWorld](#) program.

To debug a program with GDB, the program needs extended debugging symbols.

This has already been added while building the program.

- Open [Minicom](#)

```
#sudo minicom -D /dev/ttyUSBx.
```

Type User name `"root"` and press Enter

Start the GDB server:

```
# gdbserver 192.168.1.4:10000 myHelloWorld
```

Note: 192.168.1.4 is Host-system-IP

You have started the GDB server on the target. The GDB server is now waiting for connections on TCP port - 10000.

1.6.2. Configuring and starting the debugger in Eclipse

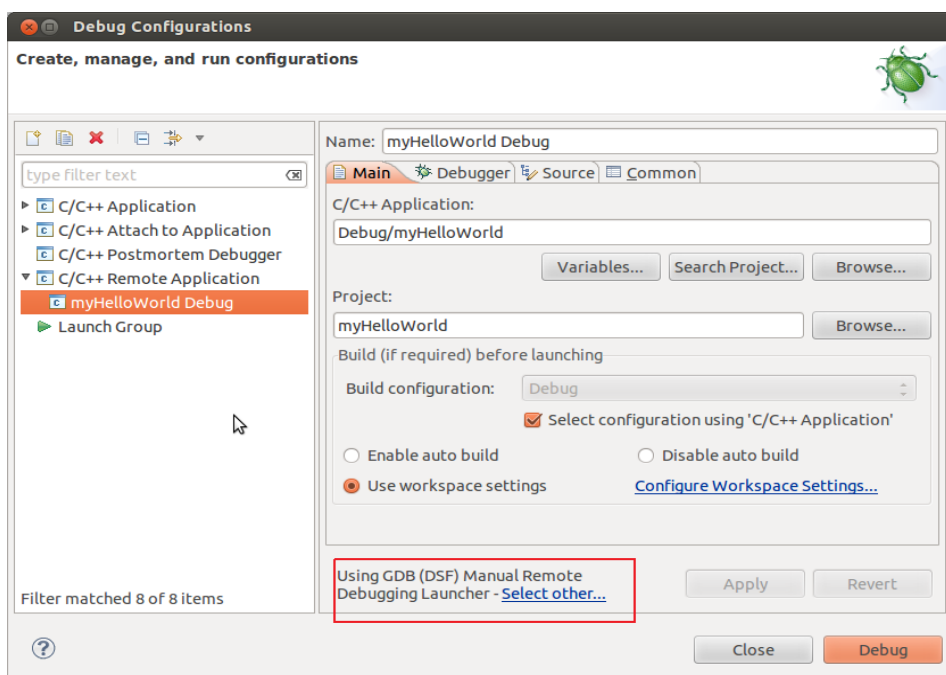
In this passage we will learn how to configure our project settings to use Eclipse with the GNU debugger.

After the configuration of our project settings, the GNU debugger will start and connect to the GDB server on the target.

- Start Eclipse if the application is not started yet
- Right-click on the myHelloWorld project in the Navigator window
- Select **Debug As** ► **Debug Configurations**

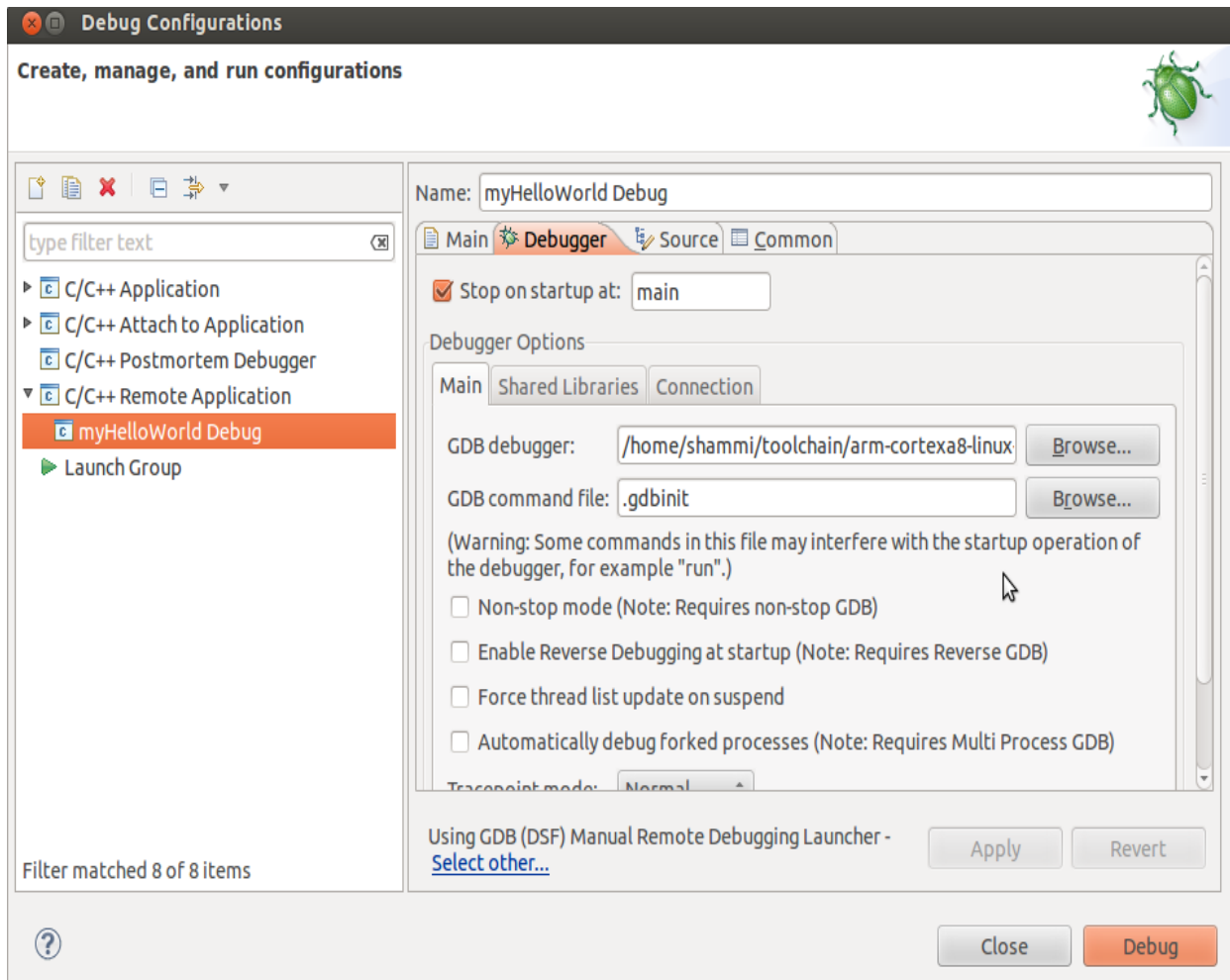
A dialog to create, manage and run applications appears.

- Double click on **C/C++ Remote Application** ► select **myHelloWorld Debug**

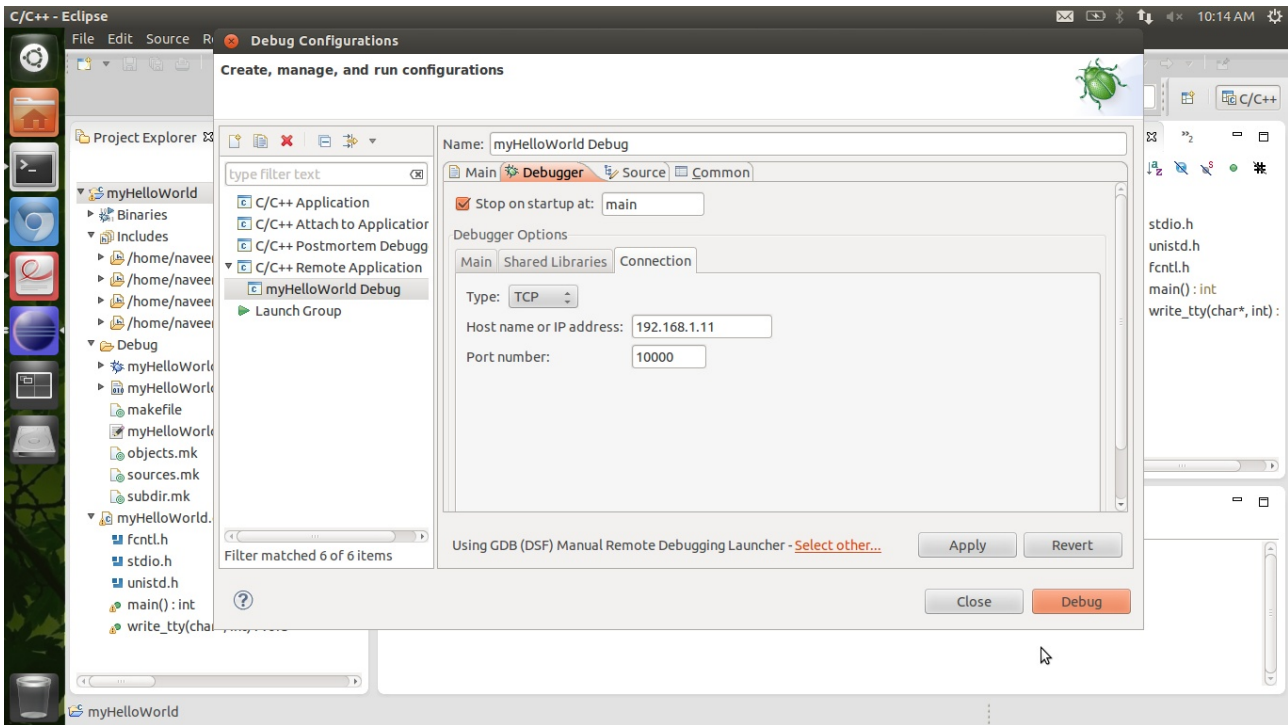


Note: Make sure that on the bottom of the Debug configuration Window it says “Using GDB (DSF) **Manual Remote Debugging Launcher**”. If it does not, then click on the “**Select Other**” and select this option.

- Select the Debugger tab
- Click the Browse button right beside the GDB debugger input field.
- Navigate to the directory <Path of the Toolchain>/bin/arm-cortexa8-linux-gnueabi-hf-gdb
- Click OK



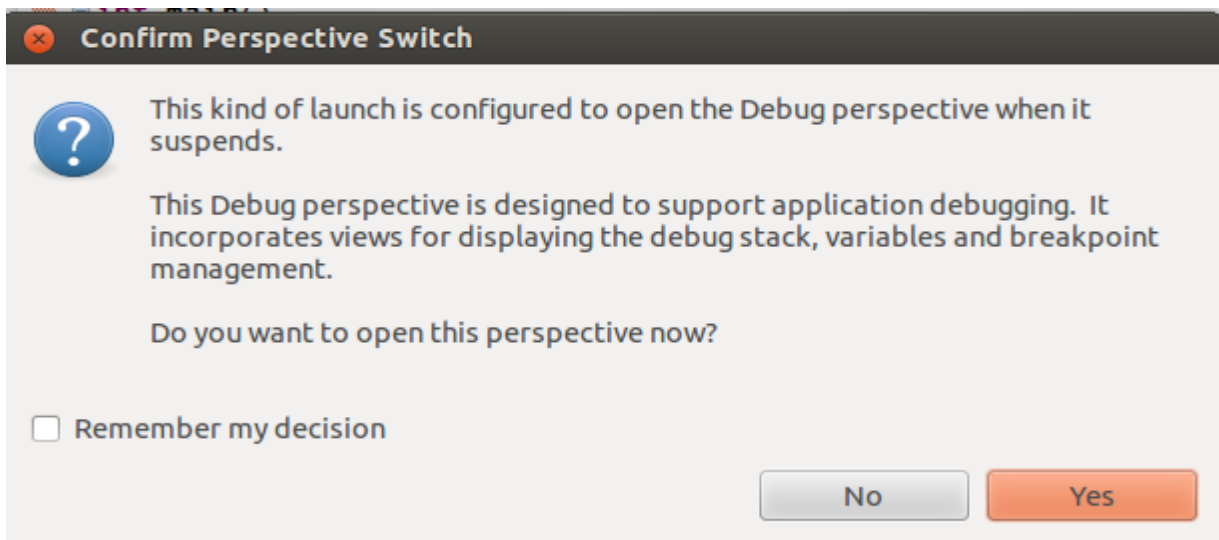
- Select the **Connection** tab and select TCP in the drop-down box. Enter 192.168.1.11 (the target's IP address) in the Host name input field. And port number as 10000.



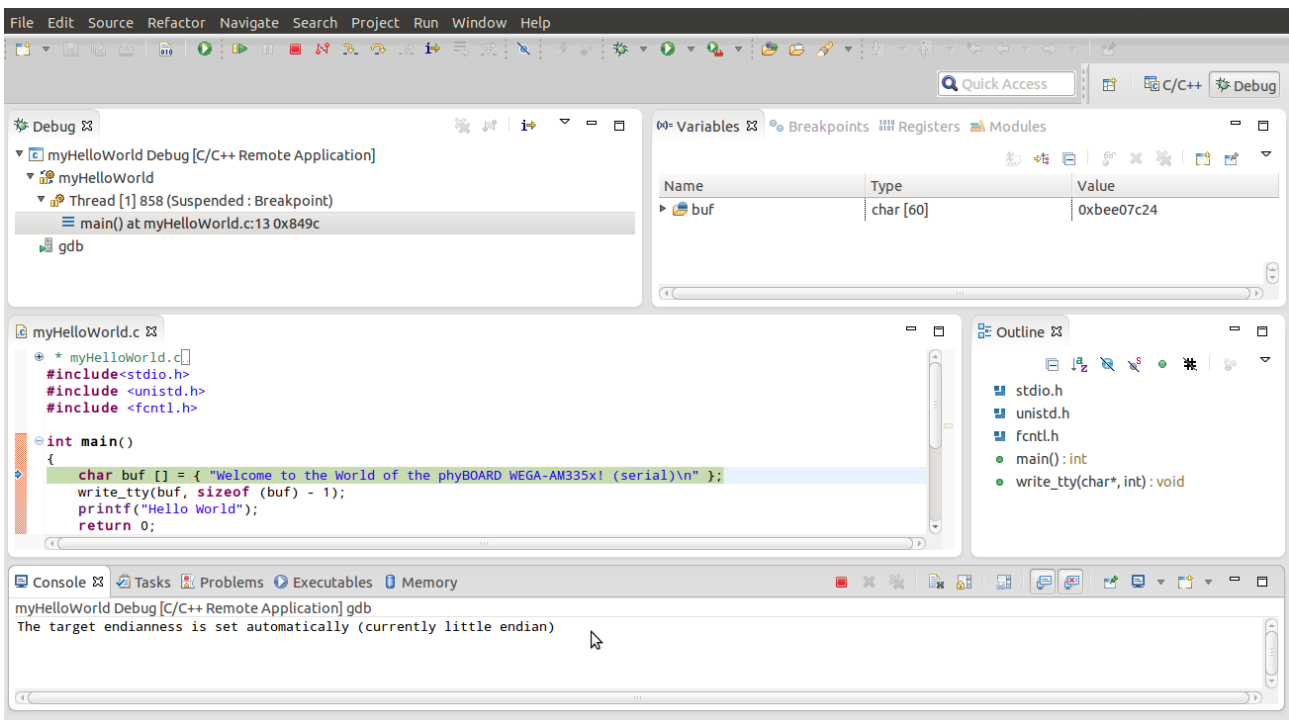
- Click Apply
- Click Debug

A new dialog appears.

- Select Yes to switch to the Debug perspective



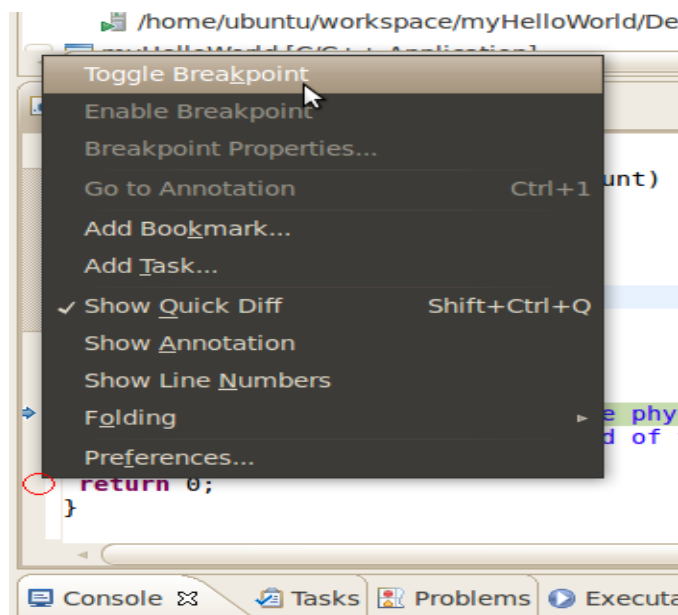
The host's GDB is now connected to the GDB server on the target.



We have configured our project for Remote-Debugging, GNU debugger in Eclipse and connected the host's GDB with the target's GDB server. Now we can debug the project.

1.6.3. Setting a Breakpoint

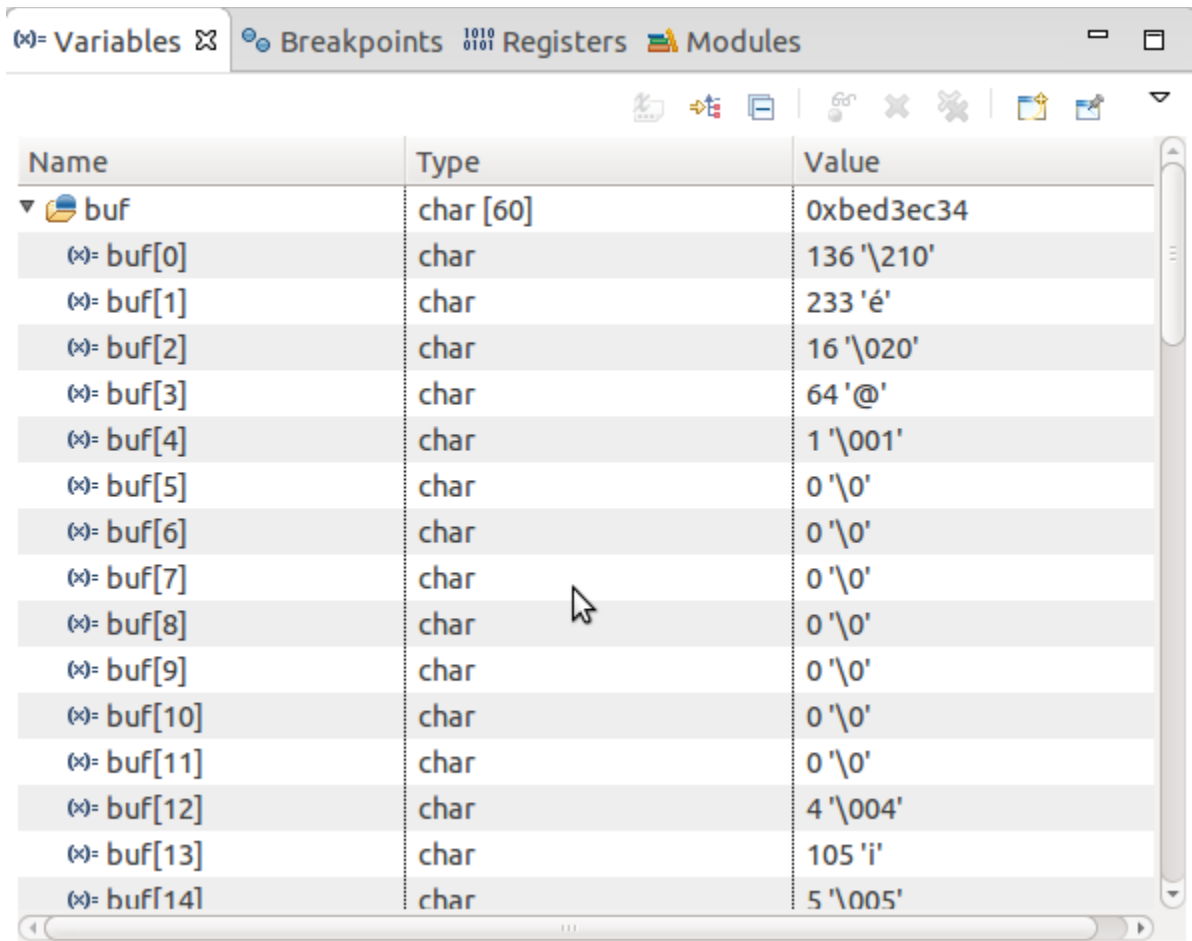
Now we will set a breakpoint in our program. This breakpoint will be set on the last line of the function main(). If you resume the application, the debugger will stop on this line.



Select the last line in `main()`. Right-click into the small grey border on the left-hand side and select Toggle Breakpoint to set a new breakpoint

1.6.4. Stepping and Watching Variable Contents

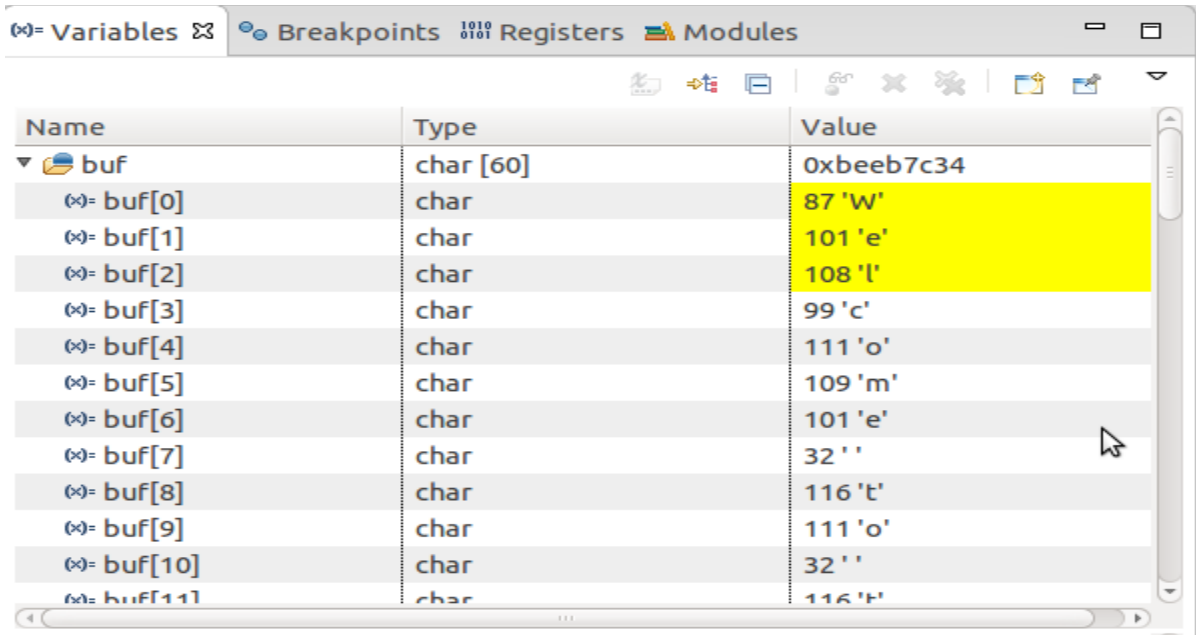
In this part we will step through the example project with the debugger. we will also learn how to check the content of a variable. Expand `buf` in the Variables window



- Click the Step Over button in the Debug window to step to the next line



we will see the content of the buf variable in the Variables window.

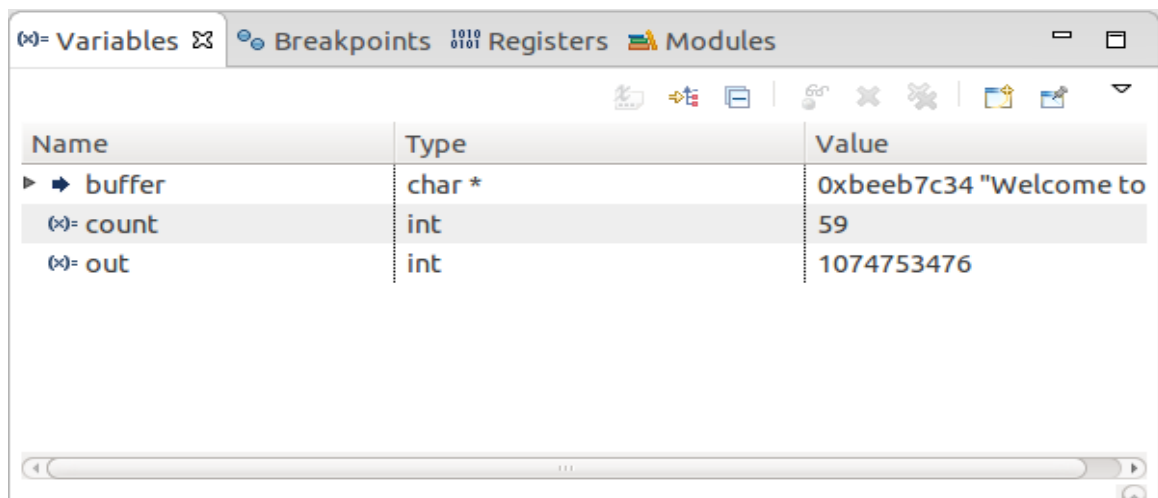


- Click on the variable buf.
- Then click the button Step into to enter the function write_tty()



The debugger stops in `write_tty()`.

we will see the following variable window:



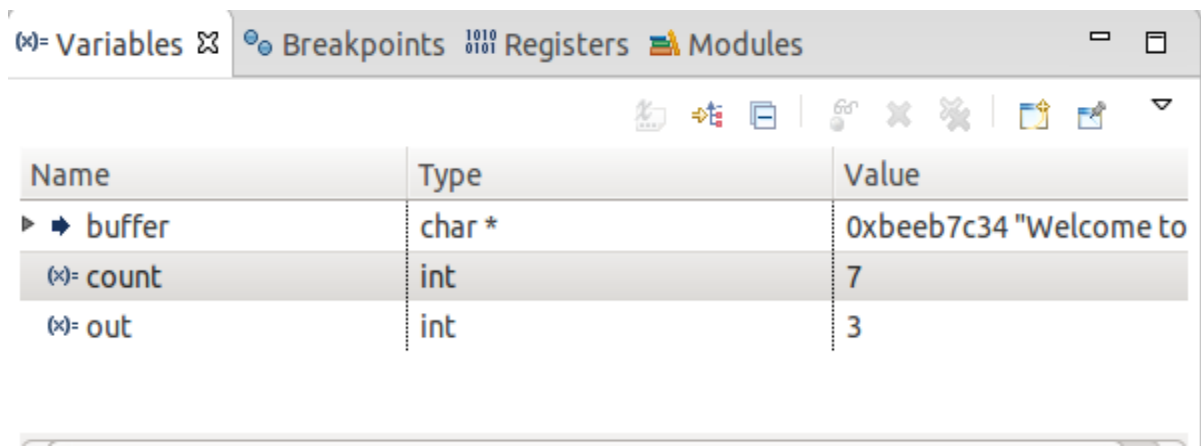
- Click on the variable buffer, as pointed in above Figure.

We will probably see a different address on the buffer pointer. Remember what address is shown in our case; we will need this address later.

1.6.5. Stepping and Watching Variable Contents

In this section we will change the value of a variable. At the end of this part we will see the effect of this change.

- Select the count variable in the Variables window
- Double click on value and Change Value to 7



- Click the Step Over button two times



Change to [Minicom](#)

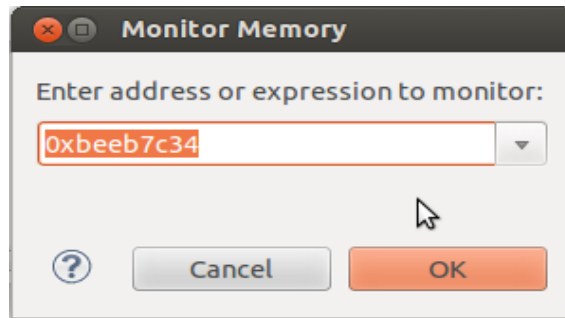
```
root@phyBOARD-WEGA-AM335x:~ gdbserver 192.168.1.10:10000 myHelloWorld
Process myHelloWorld created; pid = 858
Listening on port 10000
Remote debugging from host 192.168.1.10
Welcome
```

We will see the output [Welcome](#) in the Minicom window. This shows when changing the counter variable's value to 7 only the first seven characters of the buffer are output, instead of the whole sentence.

1.6.6. Using the Memory Monitor

In the last section of this chapter you will use the memory monitor to control the content at a memory address.

- Select the Memory tab
- Click Add Memory Monitor

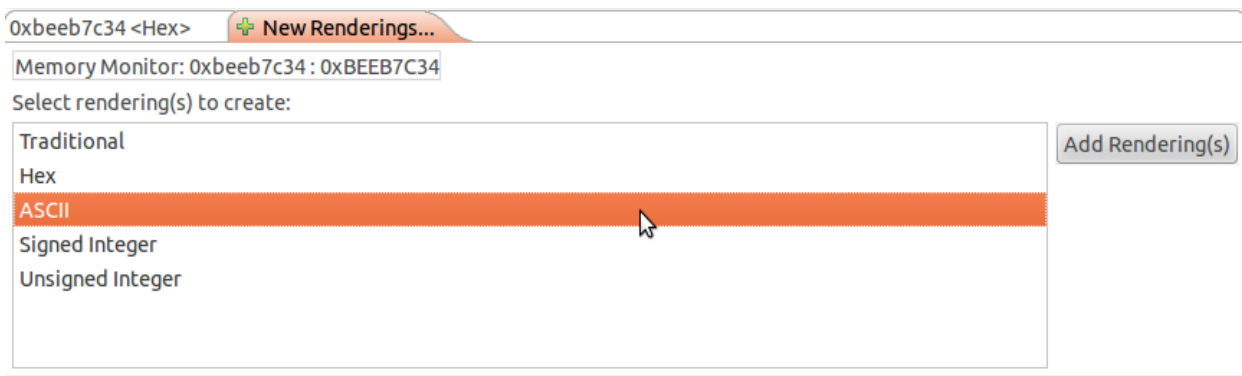


- Enter the address of buffer and click *OK*. Remember that the variable's address might differ with your system.

You will see following output:

Address	0 - 3	4 - 7	8 - B	C - F
BEEB7C30	50B60340	57656C63	6F6D6520	746F2074
BEEB7C40	68652057	6F726C64	206F6620	74686520
BEEB7C50	70687942	4F415244	2D52414E	412D414D
BEEB7C60	33333578	21202873	65726961	6C290A00
BEEB7C70	00000000	0CFE1040	00002240	C47DEBBE
BEEB7C80	01000000	90840000	00000000	00000000
BEEB7C90	5C830000	00000000	00000000	00000000
BEEB7CA0	00700F40	00000000	787CEBBE	C8FD1040
BEEB7CB0	00000000	00000000	00000000	00000000

- Click New Rendering
- Select ASCII and click Add Rendering(s).



You will see following output on your screen.

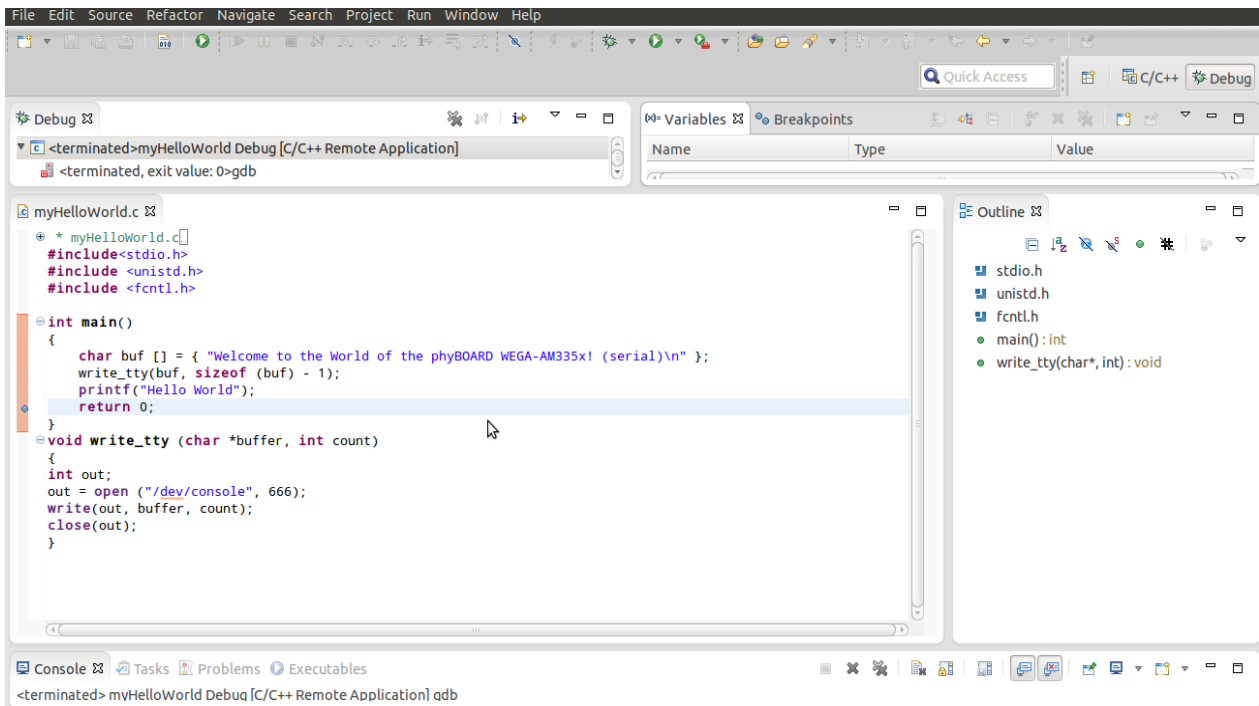
Address	0 - 3	4 - 7	8 - B	C - F
BEEB7C30	P@	Welc	ome	to t
BEEB7C40	he W	orld	of	the
BEEB7C50	phyB	OARD	-RAN	A-AM
BEEB7C60	335x	! (s	eria	l)
BEEB7C70		p@	"@	Ä}e%
BEEB7C80		„		
BEEB7C90	\f			
BEEB7CA0	p@		x e%	Ëý@
BEEB7CB0				
BEEB7CC0				
BEEB7CD0				

You can see the contents of the variable buffer at the address **0xbeeb7c34** (or whatever address is used on your system).

- Now click the Resume button from the menu bar



The debugger stops at the breakpoint in the last line of main() i.e return 0.



- Click the *Resume* button to end the application.

PHYTEC

**Get the dialog going ...
... and stay in touch**

India

**PHYTEC Embedded Pvt. Ltd.
16/9C, 3rd Floor,
3rd Main, 8th Block,
Opp: Police Station,
Koramangala,
Bangalore -560095
www.phytec.in**

Germany

**PHYTEC Messtechnik GmbH
Robert-Koch-Straße 39
D-55129 Mainz
Tel.: +49 6131 9221-32
Fax: +49 6131 9221-33
www.phytec.de
www.phytec.eu**

America

**PHYTEC America LLC
203 Parfitt Way SW, Suite G100
Bainbridge Island, WA 98110
Tel.: +1 206 780-9047
Fax: +1 206 780-9135
www.phytec.com**

France

**PHYTEC France SARL
17, place St. Etienne
F-72140 Sillé le Guillaume
Tel.: +33 2 43 29 22 33
Fax: +33 2 43 29 22 34
www.phytec.fr**

.....**We are looking forward to hearing from you!**.....