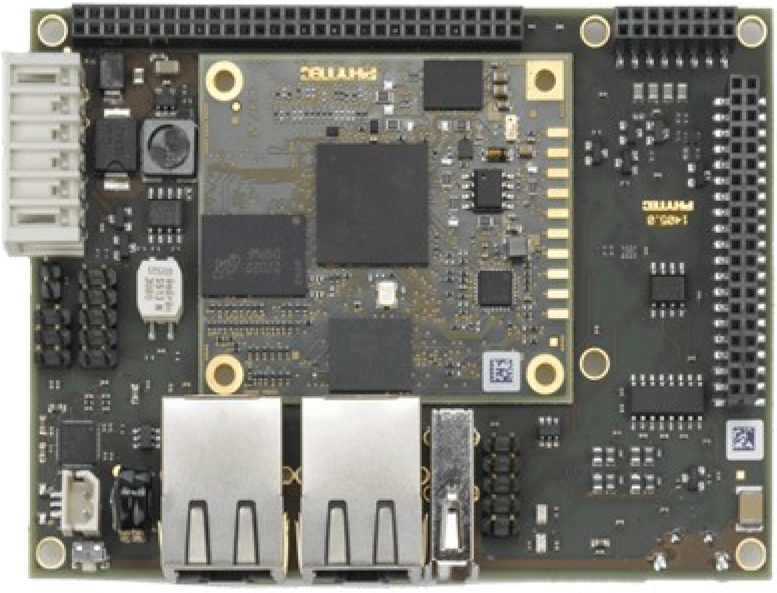
phyBOARD WEGA-AM335x
Single Board Computer

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**

A product of a PHYTEC Technology Holding Company

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.

	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

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 eclipse from below link:

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

Note: Skip the above step if you have install the WEGA_SDK for windows Host.

1.2. Eclipse IDE Configuration for phyBOARD-WEGA-AM335

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-gnueabihf.tar.bz2

For Windows:

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

Note: Skip the above step if you have install the WEGA_SDK for windows Host.

Ip address settings in windows host:

- Click Start ▸ Control Panel ▸ open Network and Sharing Center
- From the Tasks menu on the left, choose Change Adaptor Settings
- 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 192.168.1.196
- Press Tab and the Subnet Mask section will populate with default numbers
- Enter gateway 192.168.1.1
- Hit Ok .

1.2.2. Target IP address configuring using serial console

WEGA Board is configured with the default ip-address for eth0 - 192.168.1.196 and for the usb0 - 192.168.1.156. These addresses can be change using below procedure.

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

```
root@phyBOARD-WEGA-AM335x:~ ifconfig -a
```

All the network interfaces details will be listed.

- To configure the ip address manually

```
root@phyBOARD-WEGA-AM335x:~ ifconfig usb0 192.168.1.156 up

root@phyBOARD-WEGA-AM335x:~ ifconfig eth0 192.168.1.196 up
```

```
// To configure the gateway ip address

    root@phyBOARD-WEGA-AM335x:~ route add default gw 192.168.1.1
    root@phyBOARD-WEGA-AM335x:~ route
```

where usb0 & eth0 are the LAN interface.

Note:
 - To make the ip address setting permanent make changes in /etc/network/interfaces & /etc/init.d/networking

 - 192.168.1.156 & 192.168.1.196 is not mandatory you can use any IP but it should be different from the server IP.

1.2.3. Eclipse Configuration for remote connection

Launch the Eclipse IDE

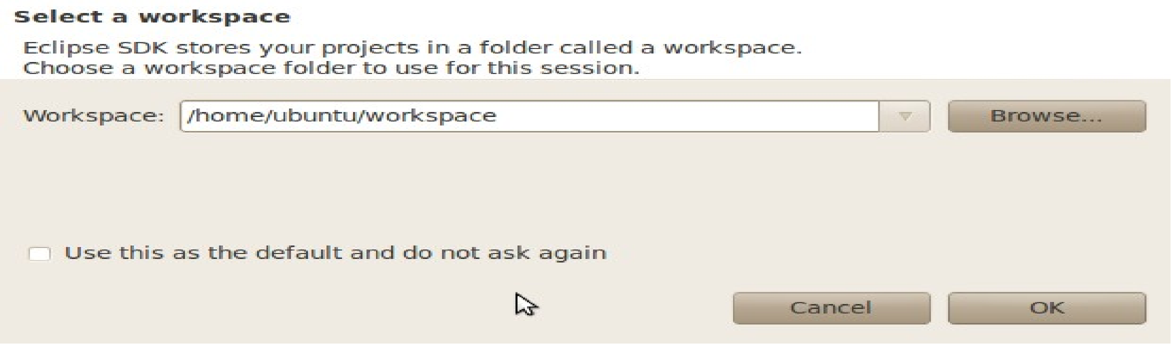
For Linux:

- 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



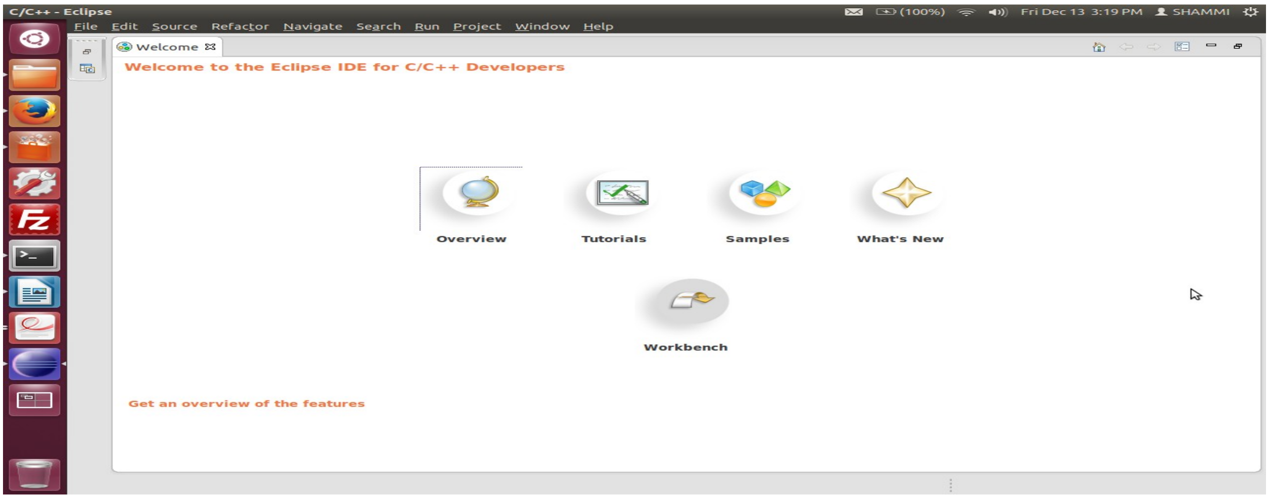
Select a workspace

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

Workspace: /home/ubuntu/workspace

□ Use this as the default and do not ask again

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

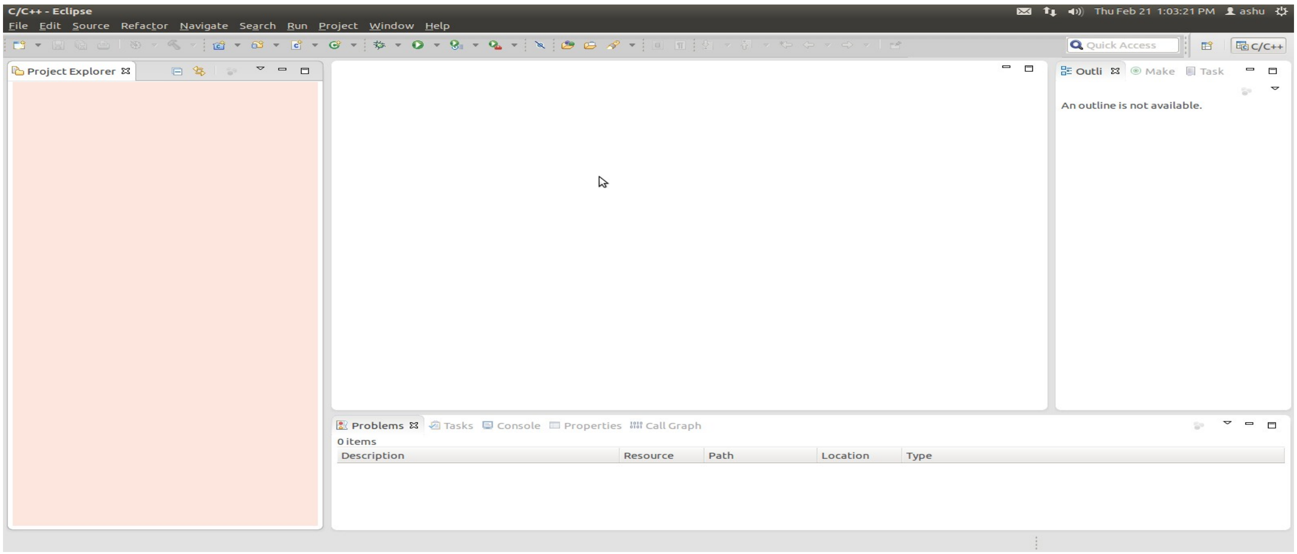


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
(100%)
Fri Dec 13 3:19 PM
SHAMMI
Welcome

Welcome to the Eclipse IDE for C/C++ Developers

Get an overview of the features

Now you can see the Eclipse Workbench as below:



An outline is not available.

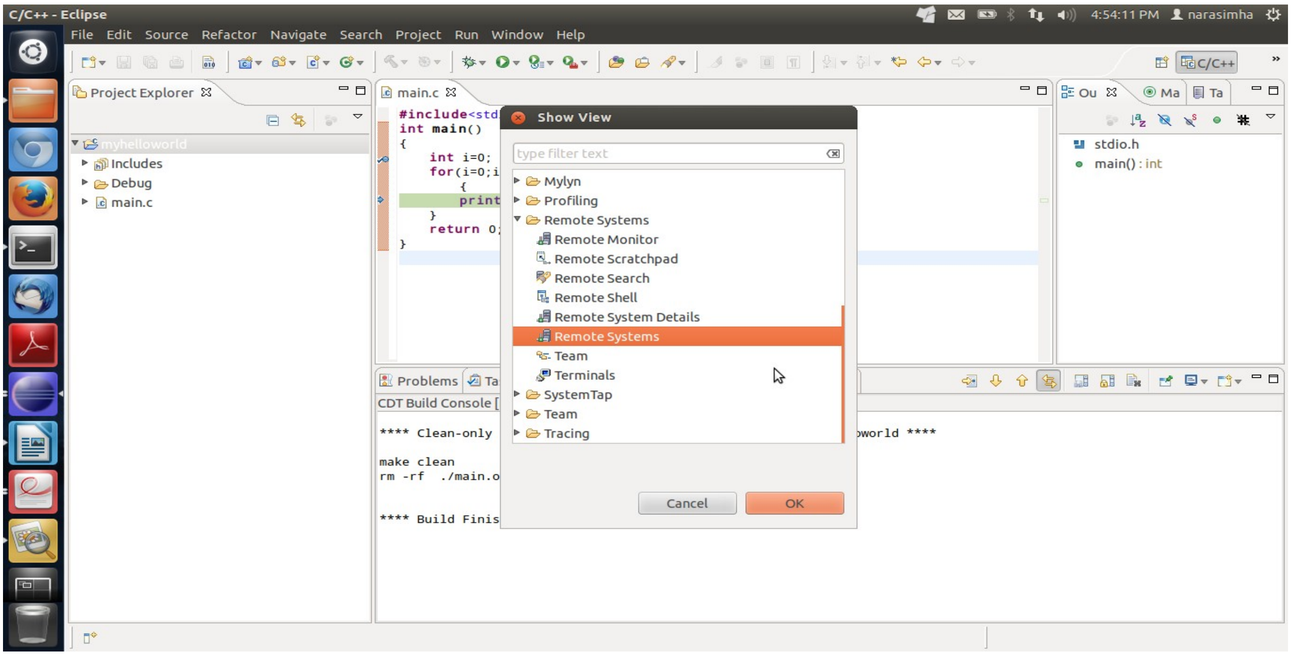
0 items

Description	Resource	Path	Location	Type

2 . Remote systems Settings For Windows (or) Linux:

You have to set the address manually, Left-click the Window tab

Show view ► other ► Remote Systems and ok



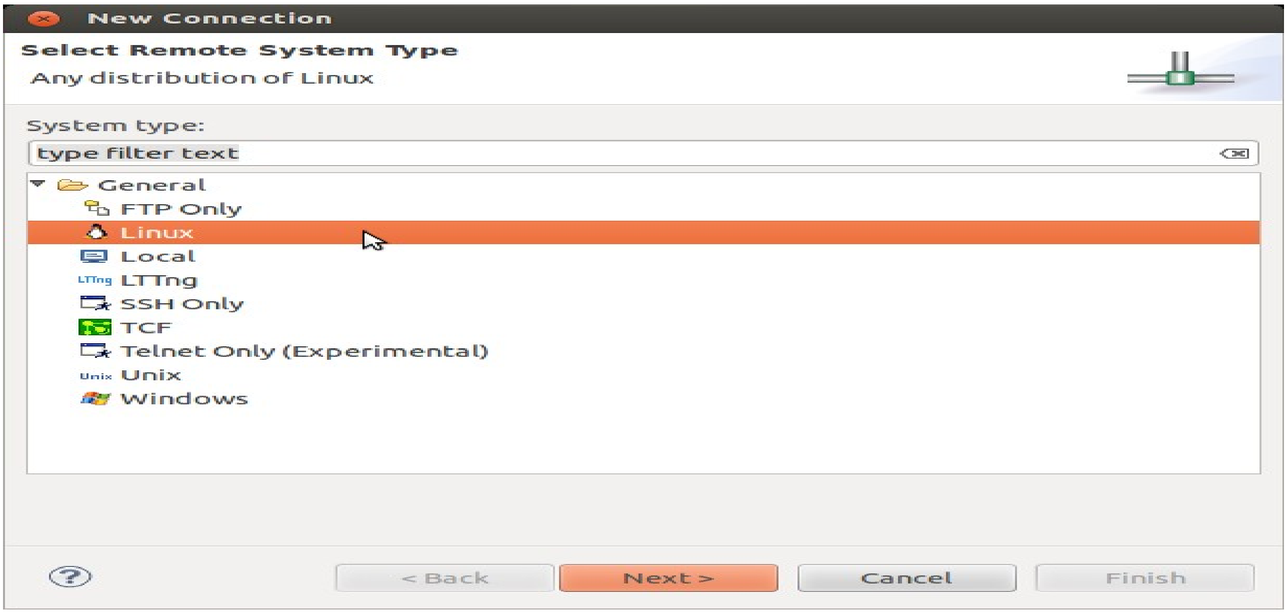
```
#include<std
int main()
{
    int i=0;
    for(i=0;i
        {
        print
    }
    return 0;
}
```

```
**** Clean-only                                    world ****

make clean
rm -rf  ./main.o

**** Build Finis
```

2.1. Create New Connection for Remote System login

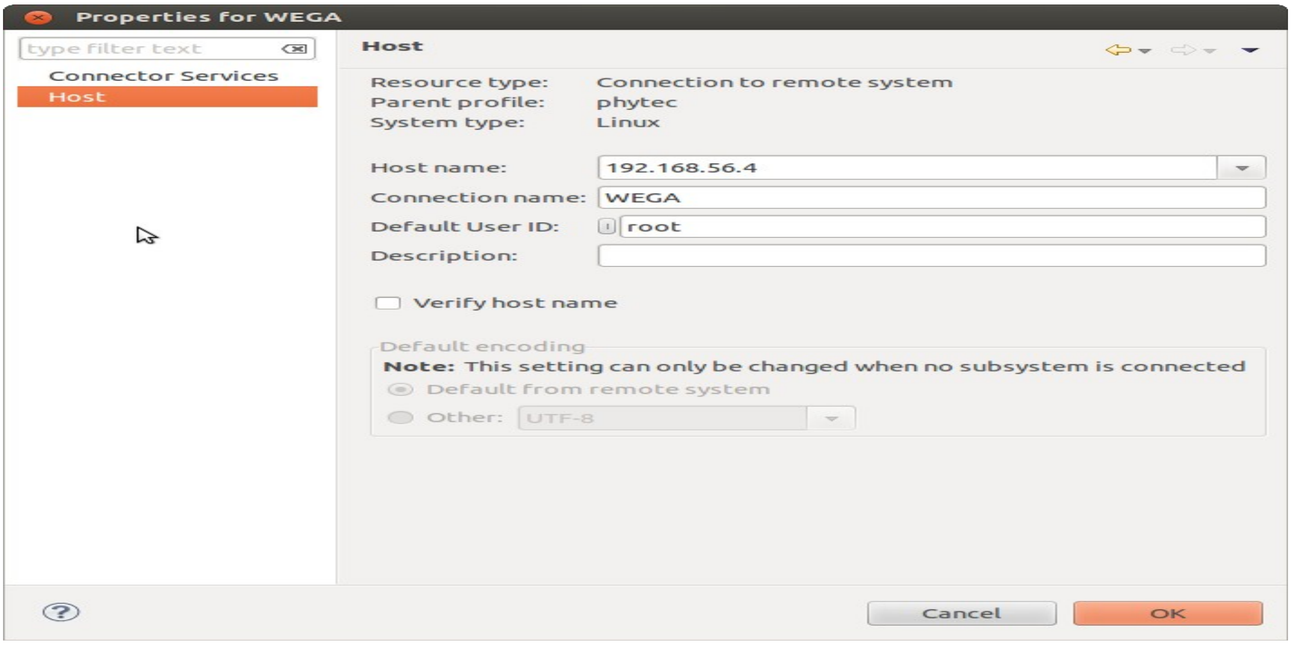
- Right Click on Local select new connection

  select linux



2.2. Set the Host Name and IP

1. Then write Host name as 192.168.56.4 and connection name as WEGA.



Properties for WEGA

Host

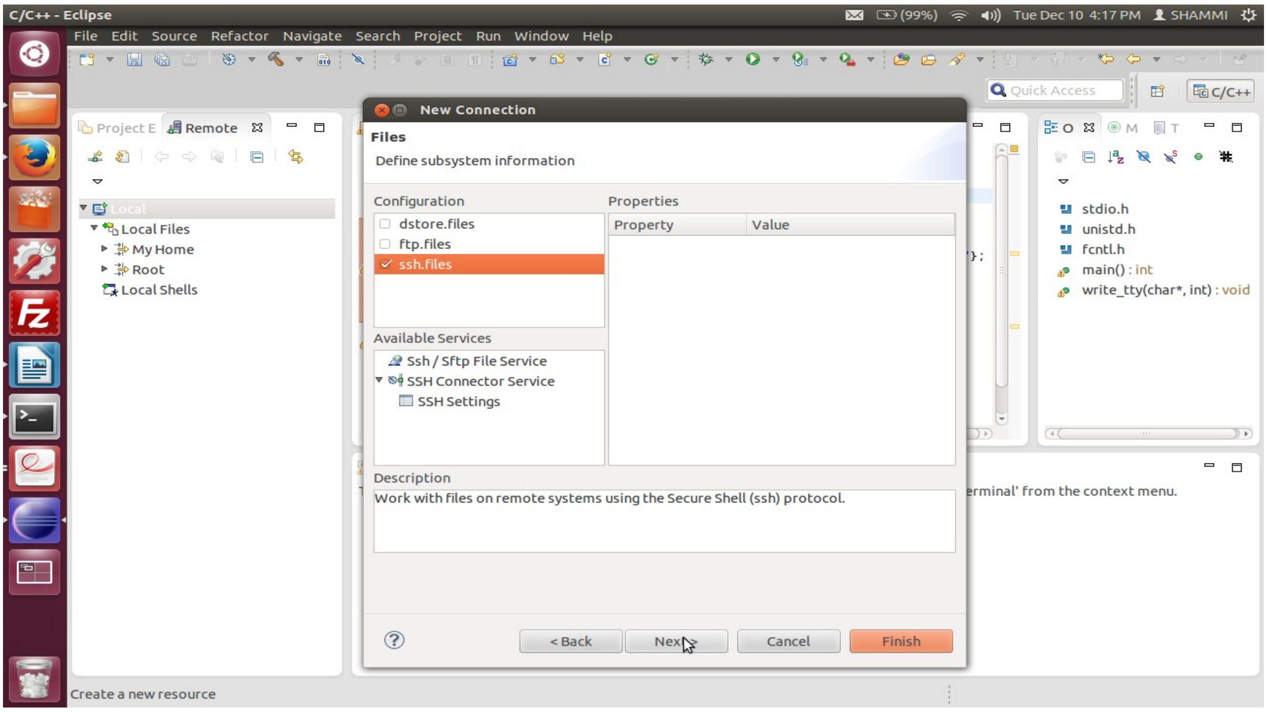
Resource type: Connection to remote system
Parent profile: phytec
System type: Linux

Host name: 192.168.56.4
Connection name: WEGA
Default User ID: root
Description:

□ Verify host name

Default encoding
Note: This setting can only be changed when no subsystem is connected
○ Default from remote system
○ Other: UTF-8

2. Select ssh.files



New Connection

Files
Define subsystem information

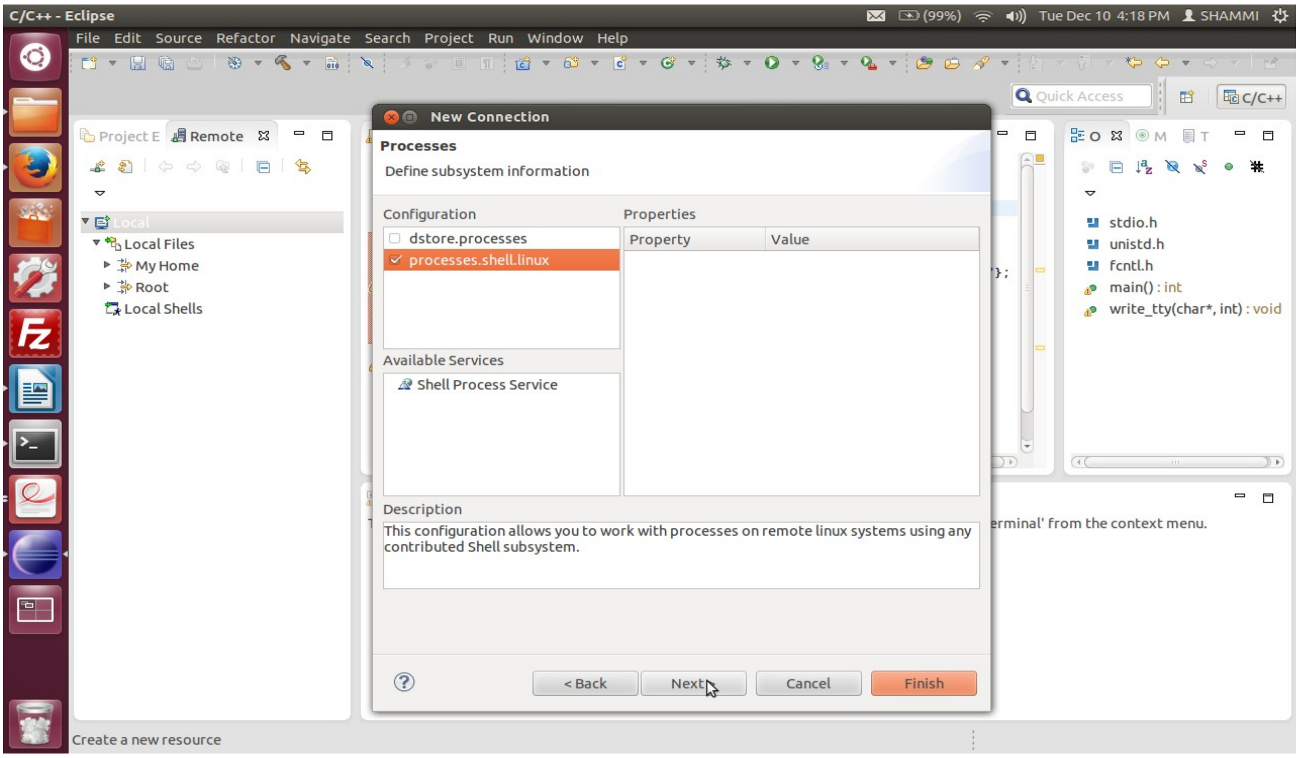
Configuration
□ dstore.files
□ ftp.files
☑ ssh.files

Properties
Property	Value

Available Services
Ssh / Sftp File Service
SSH Connector Service
SSH Settings

Description
Work with files on remote systems using the Secure Shell (ssh) protocol.

3. select processes.shell.linux and next



New Connection

Processes

Define subsystem information

Configuration

□ dstore.processes
☑ processes.shell.linux

Properties

Property	Value

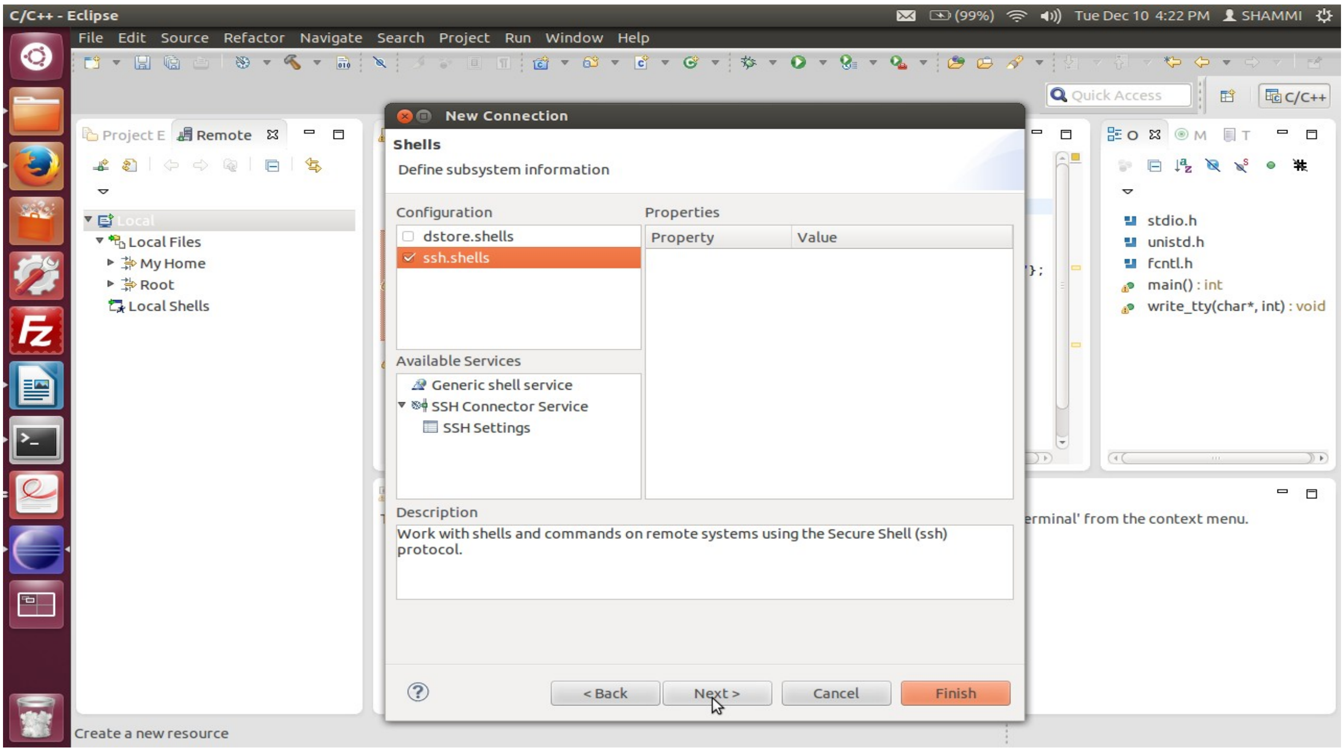
Available Services

Shell Process Service

Description

This configuration allows you to work with processes on remote linux systems using any contributed Shell subsystem.

4. select ssh.shells and next



New Connection

Shells

Define subsystem information

Configuration

□ dstore.shells
☑ ssh.shells

Properties

Property	Value

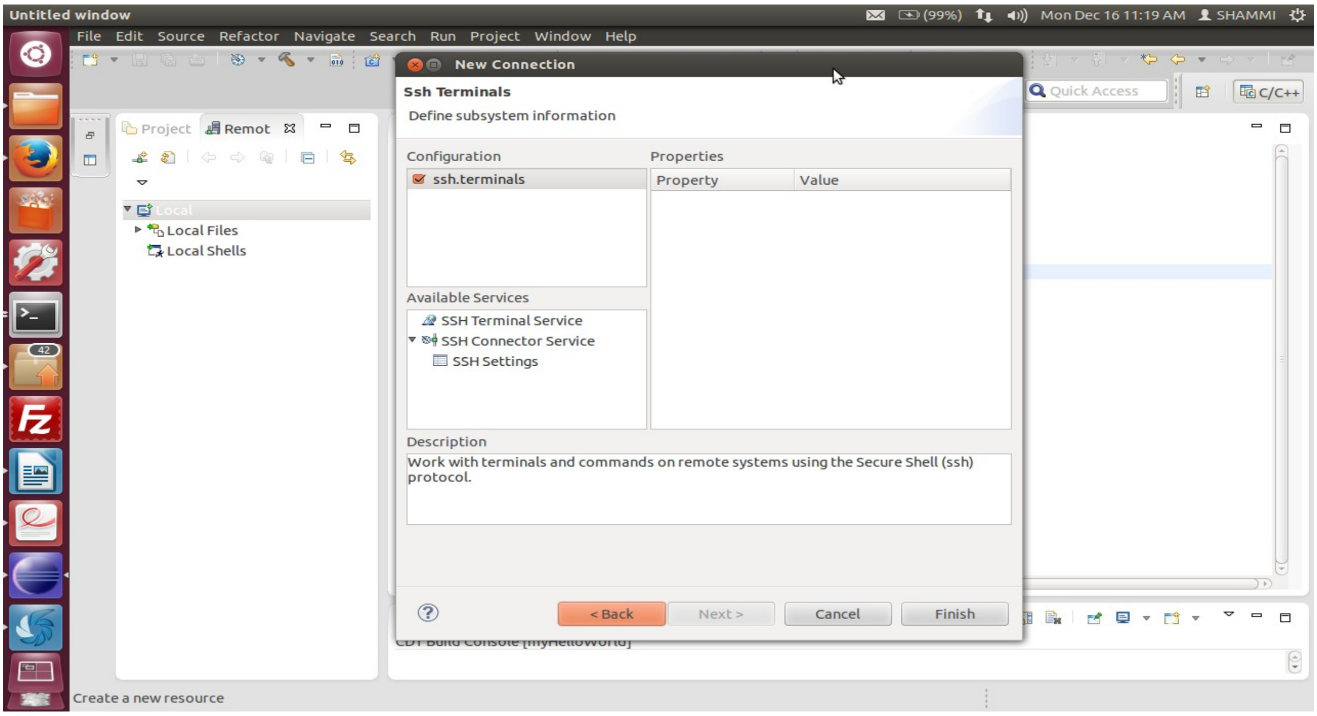
Available Services

Generic shell service
SSH Connector Service
SSH Settings

Description

Work with shells and commands on remote systems using the Secure Shell (ssh) protocol.

5. select ssh.terminals and finish



Ssh Terminals

Define subsystem information

Configuration

☑ ssh.terminals

Properties

Property	Value

Available Services

- SSH Terminal Service
- SSH Connector Service
  - SSH Settings

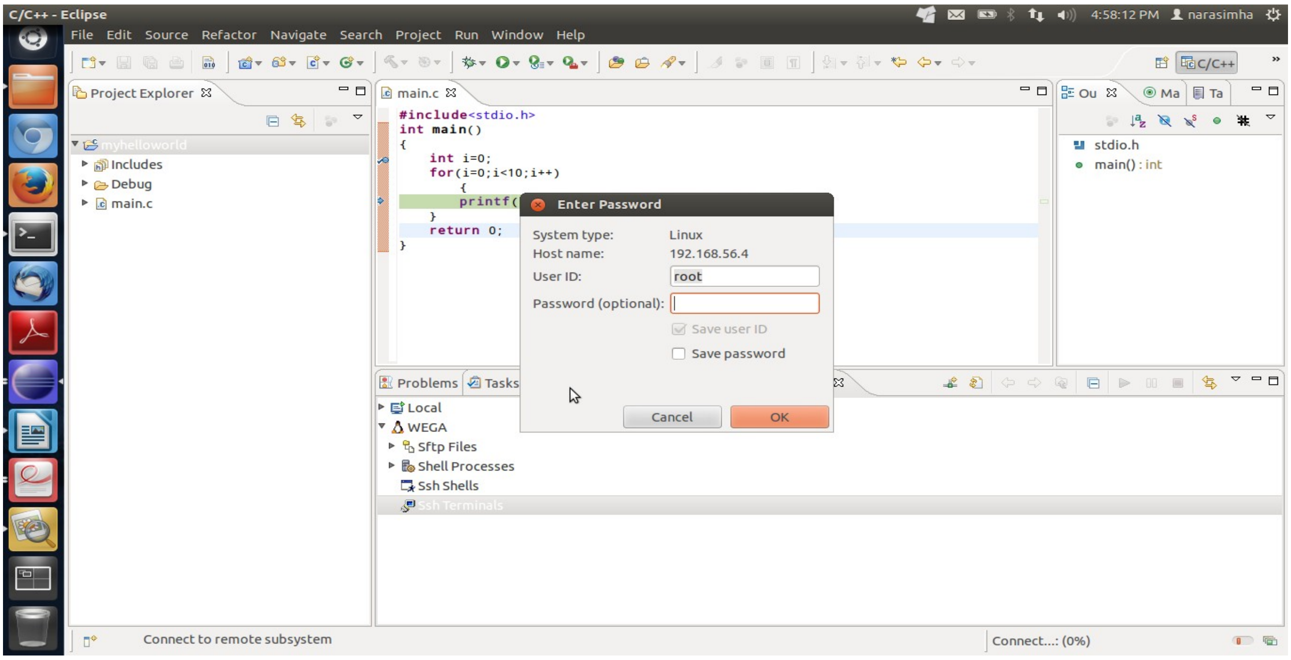
Description

Work with terminals and commands on remote systems using the Secure Shell (ssh) protocol.

Now we successfully create the connection.

Connecting to Board-IP:

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



```
#include<stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
        {
        printf(
    }
    return 0;
}
```

System type: Linux

Host name: 192.168.56.4

User ID: root

Password (optional):

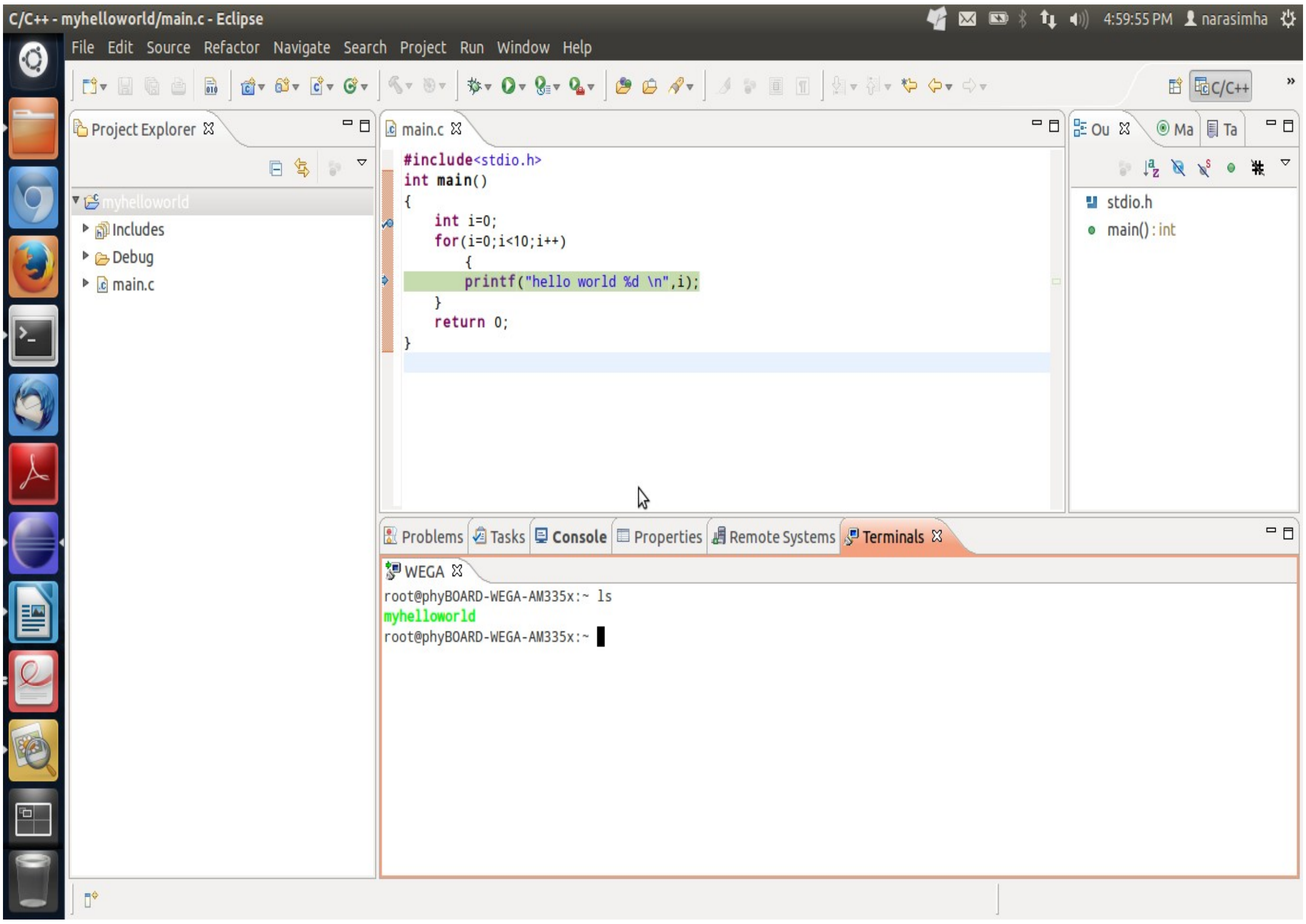
☑ Save user ID

□ Save password

2.3. Launch the Remote Terminal

- Right click ssh Terminal ► Launch Terminal

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



```
#include<stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
        {
        printf("hello world %d \n",i);
    }
    return 0;
}
```

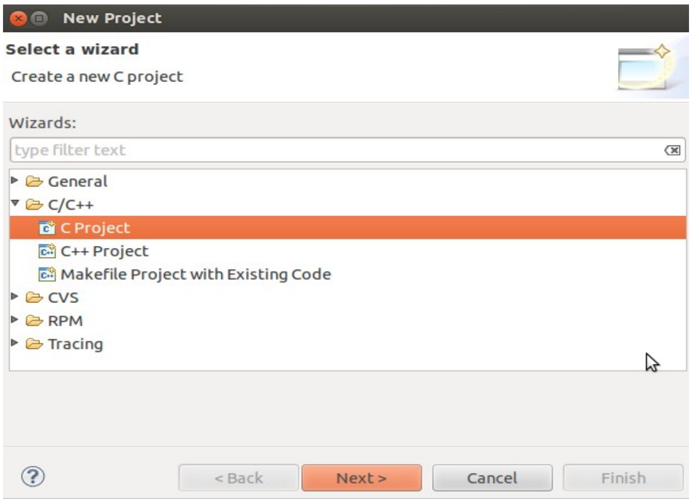
```
root@phyBOARD-WEGA-AM335x:~ ls
myhelloworld
root@phyBOARD-WEGA-AM335x:~
```

3. Creating a New Project

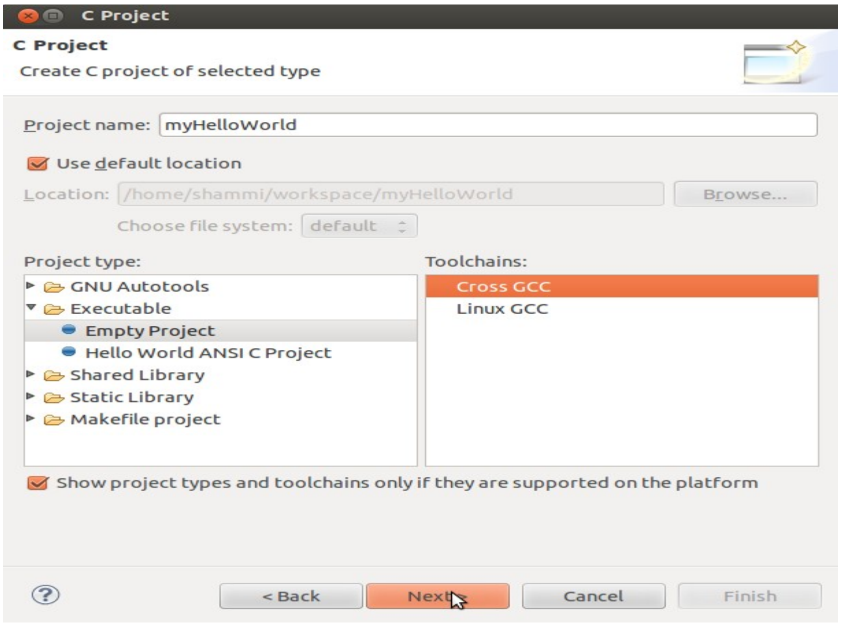
In this section we will learn how to create a new project with Eclipse and how to configure the project for use with the GNU – C/C++ cross development toolchain.

- 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



Project name: myHelloWorld

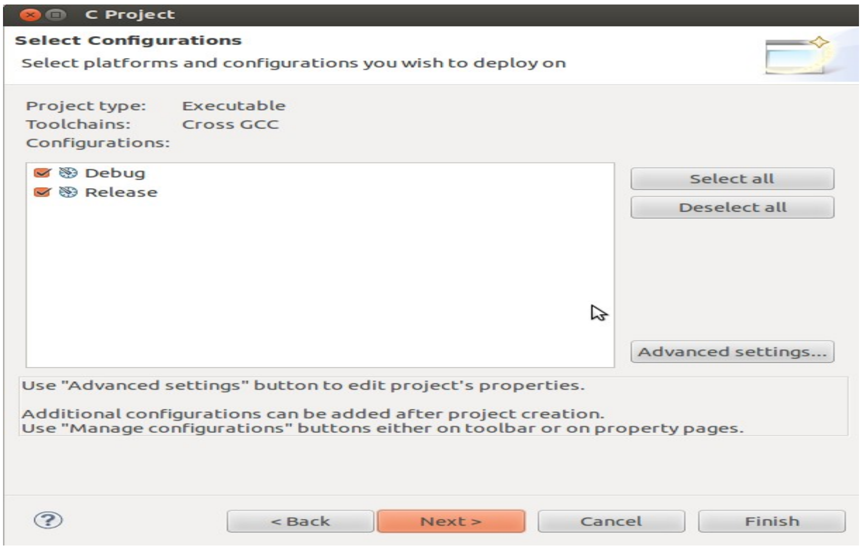
☑ Use default location

Location: /home/shammi/workspace/myHelloWorld Browse...

Choose file system: default

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

- Click Next


C Project

Select Configurations

Select platforms and configurations you wish to deploy on

Project type: Executable

Toolchains: Cross GCC

Configurations:

☑ Debug

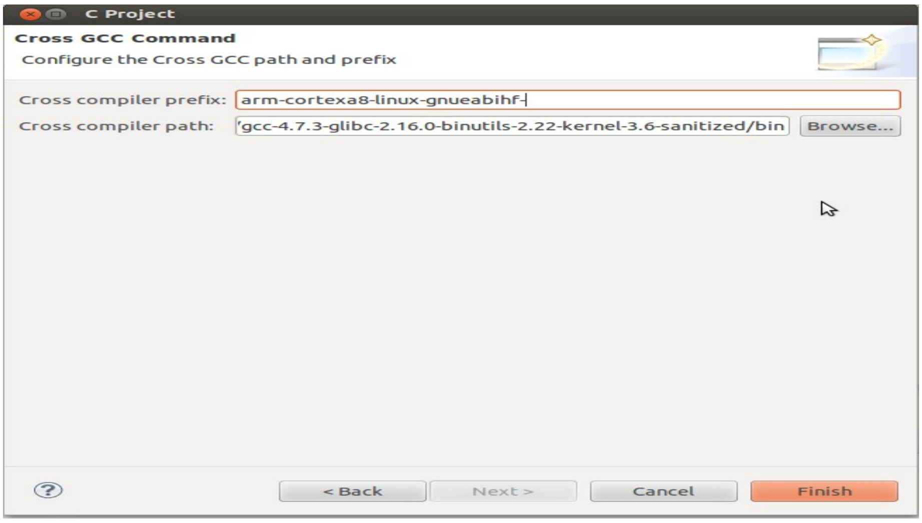
☑ Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

- **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>


C Project

Cross GCC Command

Configure the Cross GCC path and prefix

Cross compiler prefix: arm-cortexa8-linux-gnueabihf-

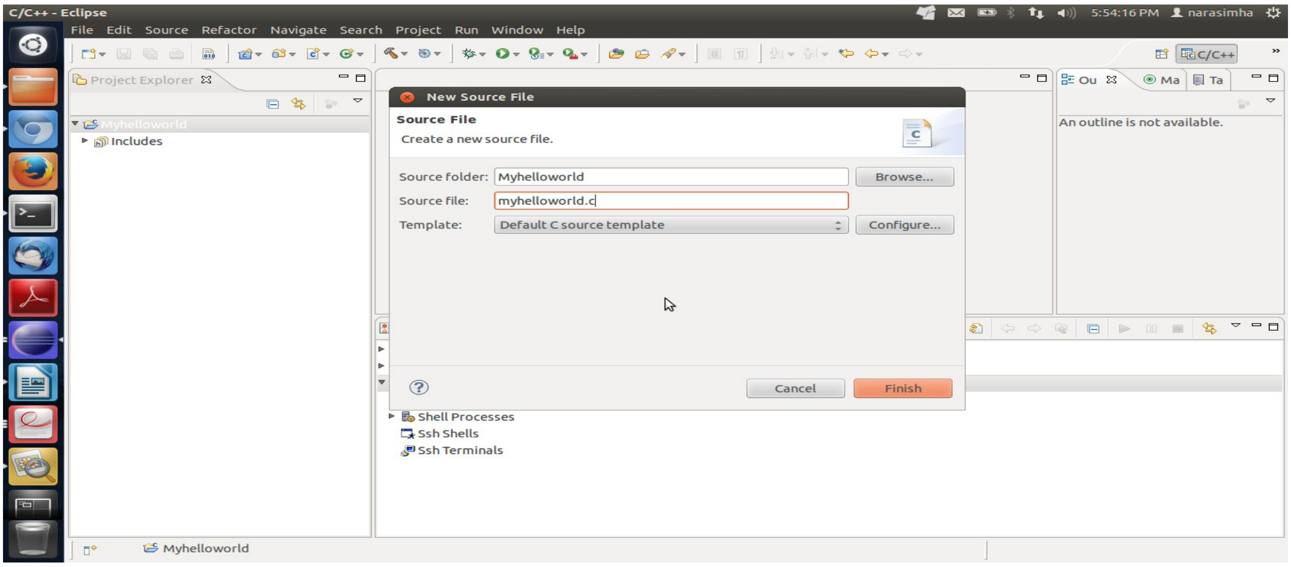
Cross compiler path: 'gcc-4.7.3-glibc-2.16.0-binutils-2.22-kernel-3.6-sanitized/bin

Note

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

3.1. 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.



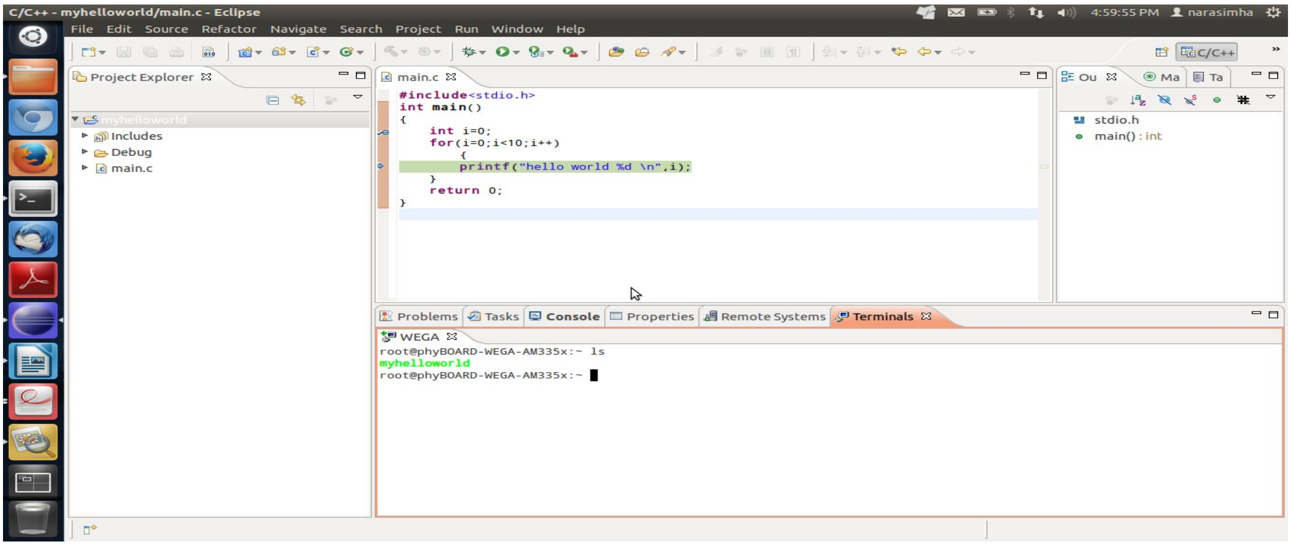
Source File
Create a new source file.

Source folder: Myhelloworld
Source file: myhelloworld.c
Template: Default C source template

3.2. Running and Debugging an example project

In this section, we will run the application on target for remote debugging in conjunction with the transfering the application binary.

Here Right click on project and select build project



```
#include<stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
        {
        printf("hello world %d \n",i);
    }
    return 0;
}
```

```
root@phyBOARD-WEGA-AM335x:~ ls
myhelloworld
root@phyBOARD-WEGA-AM335x:~
```

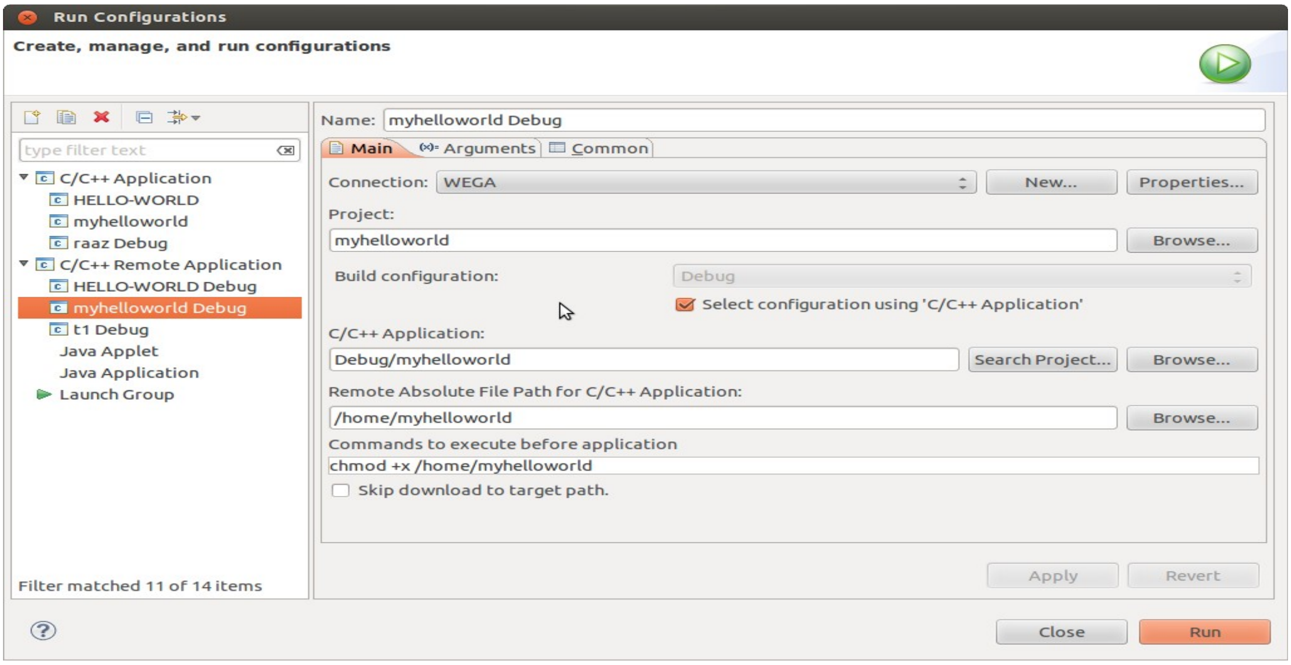
3.3. Configuring to Run project in Eclipse

- Start Eclipse if the application is not started yet.
- Right-click on the myHelloWorld project in the Navigator window
- Select Run As ▸ Run Configurations

A dialog to create, manage and run applications appears.

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

Make sure that check your ip Connection name as WEGA.



Name: myhelloworld Debug

Connection: WEGA

Project: myhelloworld

Build configuration: Debug

☑ Select configuration using 'C/C++ Application'

C/C++ Application: Debug/myhelloworld

Remote Absolute File Path for C/C++ Application: /home/myhelloworld

Commands to execute before application: chmod +x /home/myhelloworld

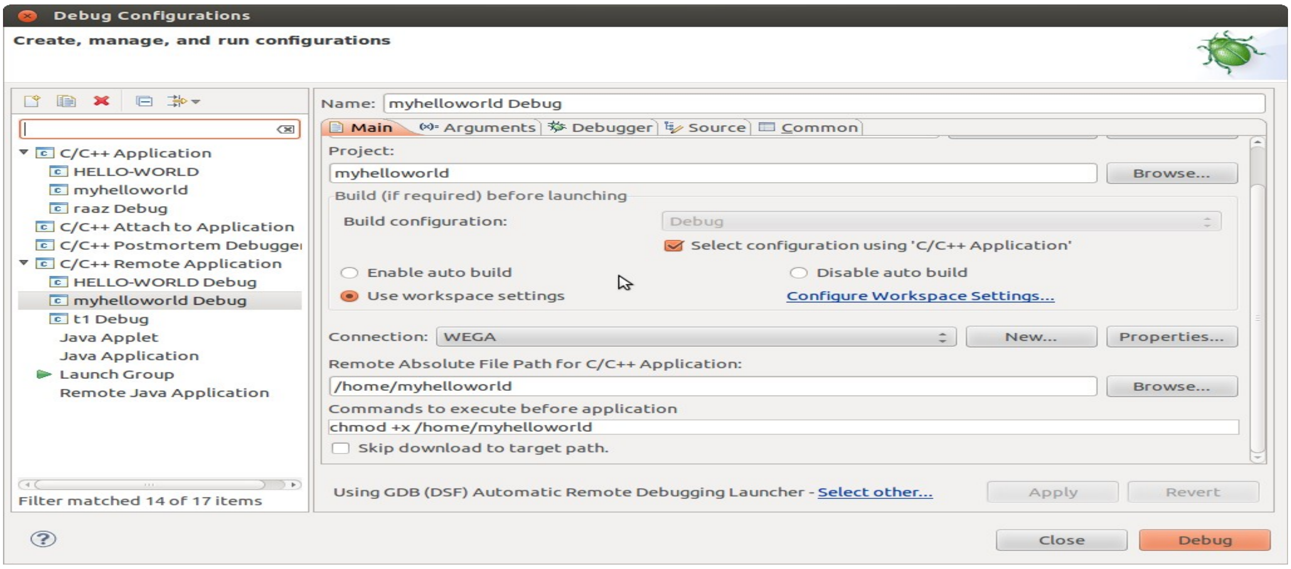
□ Skip download to target path.

3.4. Configuring to Debug project in Eclipse

- Right-click on the myHelloWorld project in the Navigator window

- Select Debug As ► Debug Configurations

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



Debug Configurations

Create, manage, and run configurations

- C/C++ Application
  - HELLO-WORLD
  - myhelloworld
  - raaz Debug
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - HELLO-WORLD Debug
  - myhelloworld Debug
  - t1 Debug
- Java Applet
- Java Application
- Launch Group
- Remote Java Application

Filter matched 14 of 17 items

Name: myhelloworld Debug

Main | Arguments | Debugger | Source | Common

Project: myhelloworld

Build (if required) before launching

Build configuration: Debug

☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Connection: WEGA

Remote Absolute File Path for C/C++ Application: /home/myhelloworld

Commands to execute before application: chmod +x /home/myhelloworld

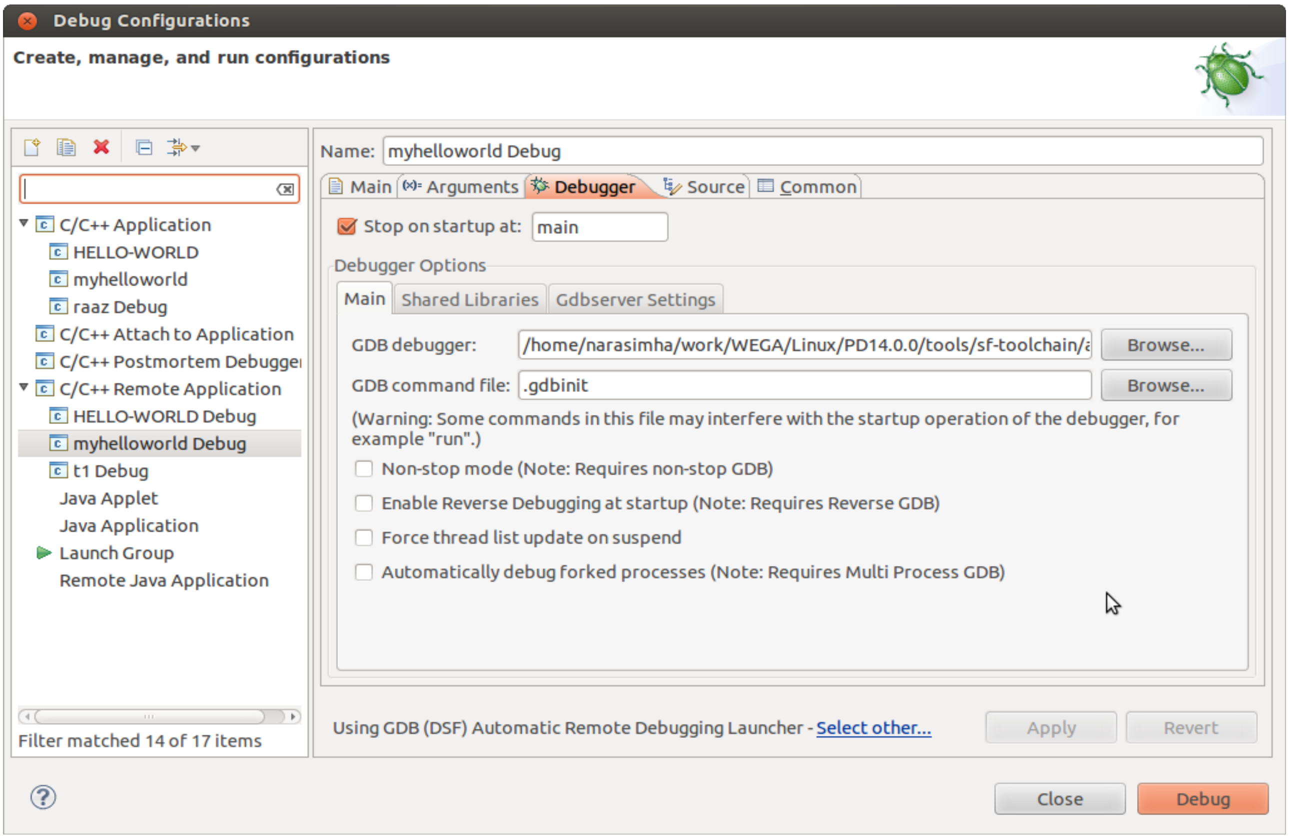
□ Skip download to target path.

Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...

- 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-gnueabihf-gdb

- Click OK



Debug Configurations

Create, manage, and run configurations

Filter matched 14 of 17 items

Name: myhelloworld Debug

☑ Stop on startup at: main

Debugger Options

GDB debugger: /home/narasimha/work/WEGA/Linux/PD14.0.0/tools/sf-toolchain/a

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

□ Non-stop mode (Note: Requires non-stop GDB)

□ Enable Reverse Debugging at startup (Note: Requires Reverse GDB)

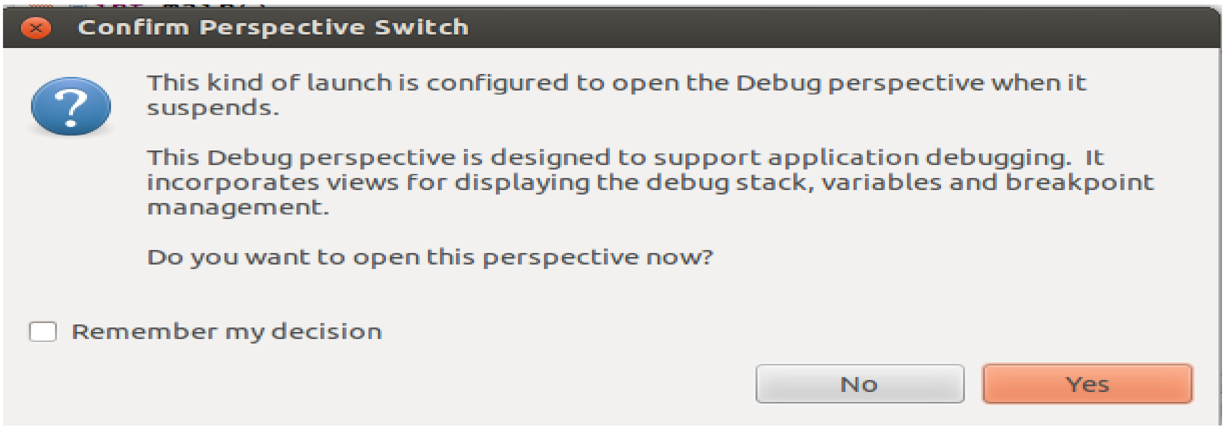
□ Force thread list update on suspend

□ Automatically debug forked processes (Note: Requires Multi Process GDB)

Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...

A new dialog appears.

- Select Yes to switch to the Debug perspective



Confirm Perspective Switch

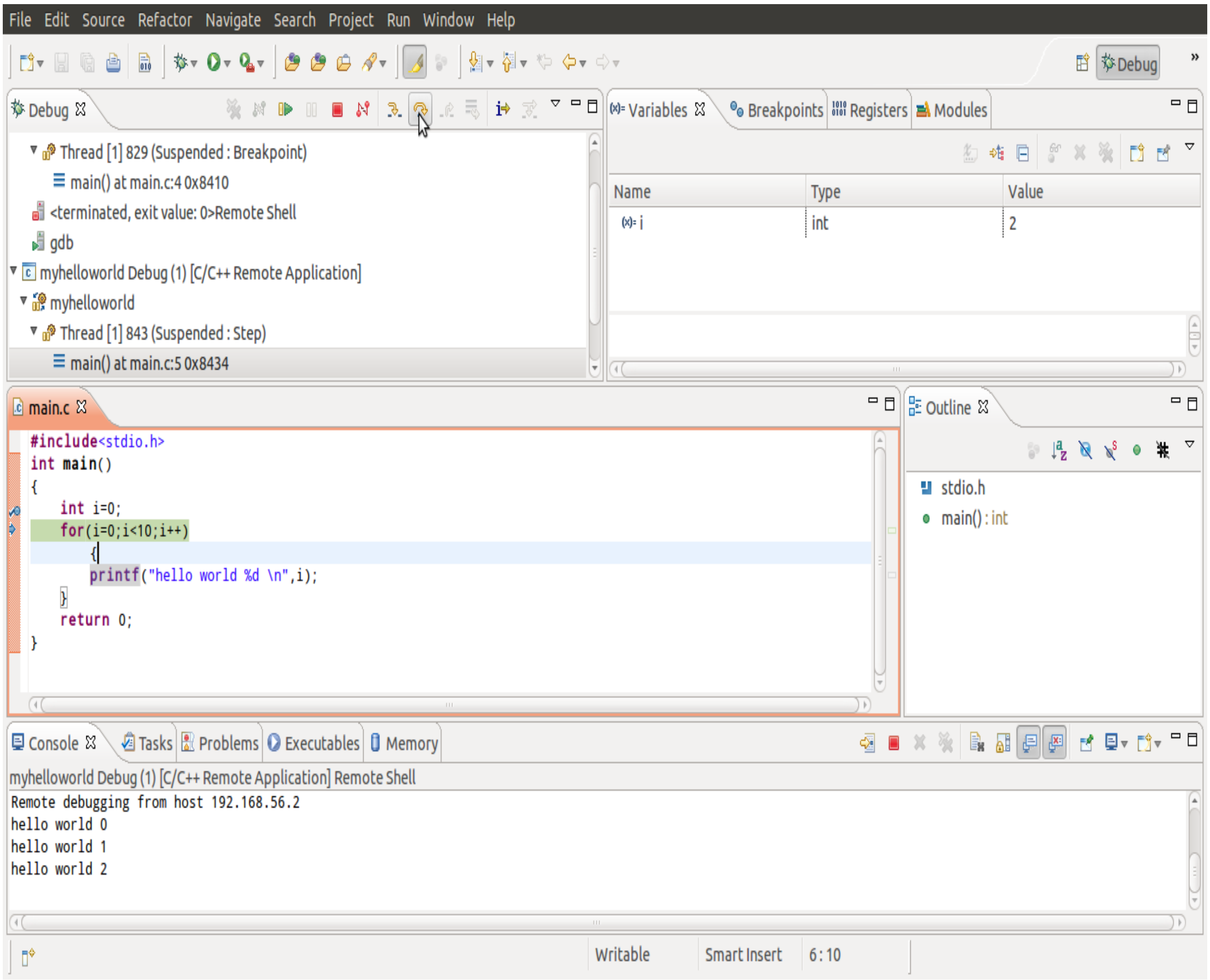
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

□ Remember my decision

Now we can debug the project.



```
Thread [1] 829 (Suspended : Breakpoint)
    main() at main.c:4 0x8410
<terminated, exit value: 0>Remote Shell
gdb
myhelloworld Debug (1) [C/C++ Remote Application]
myhelloworld
Thread [1] 843 (Suspended : Step)
    main() at main.c:5 0x8434
```

Name	Type	Value
i	int	2

```
#include<stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
        {
        printf("hello world %d \n",i);
    }
    return 0;
}
```

stdio.h
main() : int

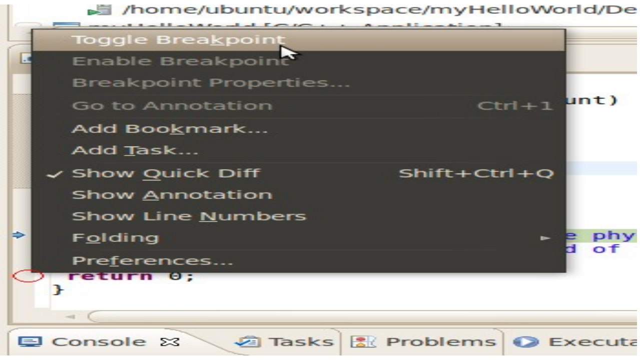
myhelloworld Debug (1) [C/C++ Remote Application] Remote Shell

```
Remote debugging from host 192.168.56.2
hello world 0
hello world 1
hello world 2
```

Writable Smart Insert 6 : 10

3.5. 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.

- Click on the step into(F5) button to observer the step into the program.

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

We will see the content of the debug output in the Variables window.

3.6. Tranfer the binary file to target manually using command line:

In Host:

here below is the file to transfer from your existance project workspace to target board by using command line in linux host.

Ex:

```
narasimha@phytec:~/work/WEGA/eclipse-work/myhelloworld/Debug$ scp
myhelloworld root@192.168.56.4:/home
```

In Target:

Open the terminal using minicom.

- Enter user name as root and press Enter then type ls to see all the file.

```
root@phyBOARD-WEGA-AM335x:~ls
```

Type ./myHelloWorld to start the application

```
root@phyBOARD-WEGA-AM335x:~./myHelloWorld
```

4. APPLICATION PROGRAM GUIDE

Contents

1. GPIO APPLICATION

Contents:

1. GPIO Introduction:

A "General Purpose Input/Output" (GPIO) is a flexible software-controlled digital signal.

<Board_Name> Linux-Kernel comes with default GPIO Driver selected and pins available are **GPIO0_7, GPIO3_7 & GPIO3_8** . For more details of GPIO pins on Expansion see the <Board_Name>_Hardware_Manual.pdf

2. GPIO – Driver Configuration:

To add additional GPIO pins, the pin-muxing in kernel board file need to be done. Follow the GPIO Section of <Board_Name>_System_Development_Guide.pdf

3. GPIO access from shell:

The GPIOs can be accessed using sysfs from below instructions.

- **a. Export: /sys/class/gpio/export**
- **b. Unexport: /sys/class/gpio/unexport**
- **c. Configure direction: /sys/class/gpio/gpio<num>/direction**
- **d. Read / Write: /sys/class/gpio/gpio<num>/value**

```
Ex: For GPIO0_7 the pin# will  be ( 0 x 32 ) + 7 = 7
    $ export GPIO_NUM=7
    $ ls /sys/class/gpio

    $ echo $GPIO_NUM > /sys/class/gpio/export
    $ ls /sys/class/gpio

    # To make the pin "high"
    $ echo 1 > /sys/class/gpio/gpio$GPIO_NUM/value
    # To make the pin "low"
    $ echo 0 > /sys/class/gpio/gpio$GPIO_NUM/value

    # To read the pin status
    $ cat /sys/class/gpio/gpio$GPIO_NUM/value
```

Note: Above commands can be used to access any gpio by modifying the GPIO_NUM variable.

4. GPIO access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

GPIO library and test program-files:

File-Name	**Description**
gpio.c	Library file
gpio.h	Library header
gpio_test.c	Test application for gpio library
Makefile	To build the gpio test program.

GPIO API's for user programming :

Function Name	**Description**
gpio_export	To Export the gpio
gpio_set_dir	To set the GPIO Pin [Direction - OUT/IN]
gpio_set_value	To set the value for the GPIO Pin.
gpio_fd_close	To close the GPIO at the end of GPIO Operations.

Code-Snippet:

```
gpio_num = argv[1];           /* for gpio number */
gpio_dir = argv[2];           /* for output direction */
gpio_val = argv[3];           /* for value (1 or 0) */

#Functions:
gpio_export(&gpio_desc);

gpio_set_dir(&gpio_desc, dir);
gpio_set_value(&gpio_desc, val);

gpio_fd_close(&gpio_desc);
```

5. Test Procedure of GPIO on <BOARD_NAME> using command line:

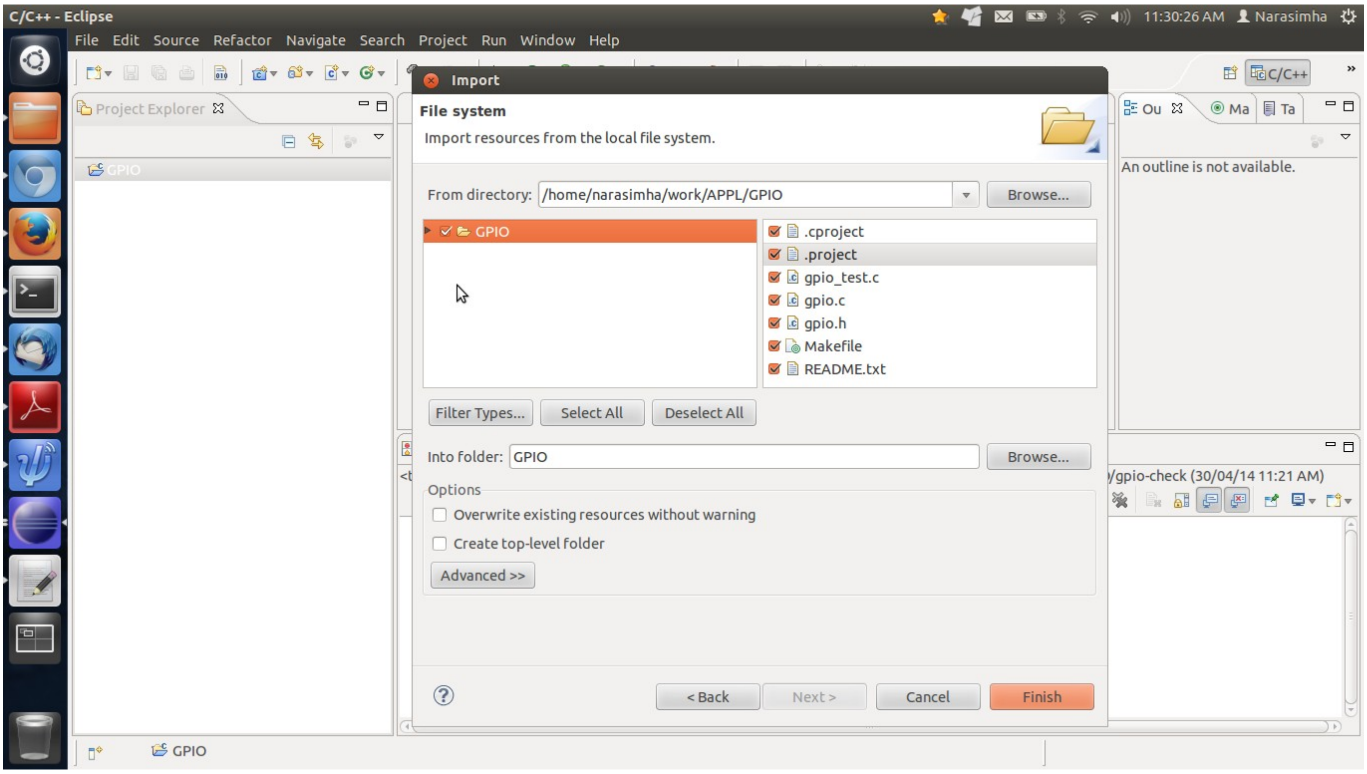
Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the gpio dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

```
     $ export PATH=$PATH:<the path of toolchain bin>
     $ cd <code_base>/app/gpio
     $ make clean
     $ make
     $ scp gpio-check root@<BOARD_NAME>:/home/
     $ ssh root@<BOARD IP-address>
     $ ./gpio-check <GPIO_NUM> <DIR> <VALUE>

To Set/Enable:
     $ ./gpio-check 7 out 1
            [GPIO_NUM : GPIO0_7, Dir : out, Value : 1]

To Clear/Disable:
    $ ./gpio-check 7 out __
            [GPIO_NUM : GPIO0_7, Dir : out,  Value : <empty>]
      [Note: At argv[3], we donot pass anything for making value as "0"]

    $ exit
```

6. Test Procedure of GPIO on <BOARD_NAME> using Eclipse IDE:

1. Select all files as below and click Finish.



C/C++ - Eclipse

Import

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/GPIO Browse...

☑ GPIO

☑ .cproject
☑ .project
☑ gpio_test.c
☑ gpio.c
☑ gpio.h
☑ Makefile
☑ README.txt

Filter Types... | Select All | Deselect All

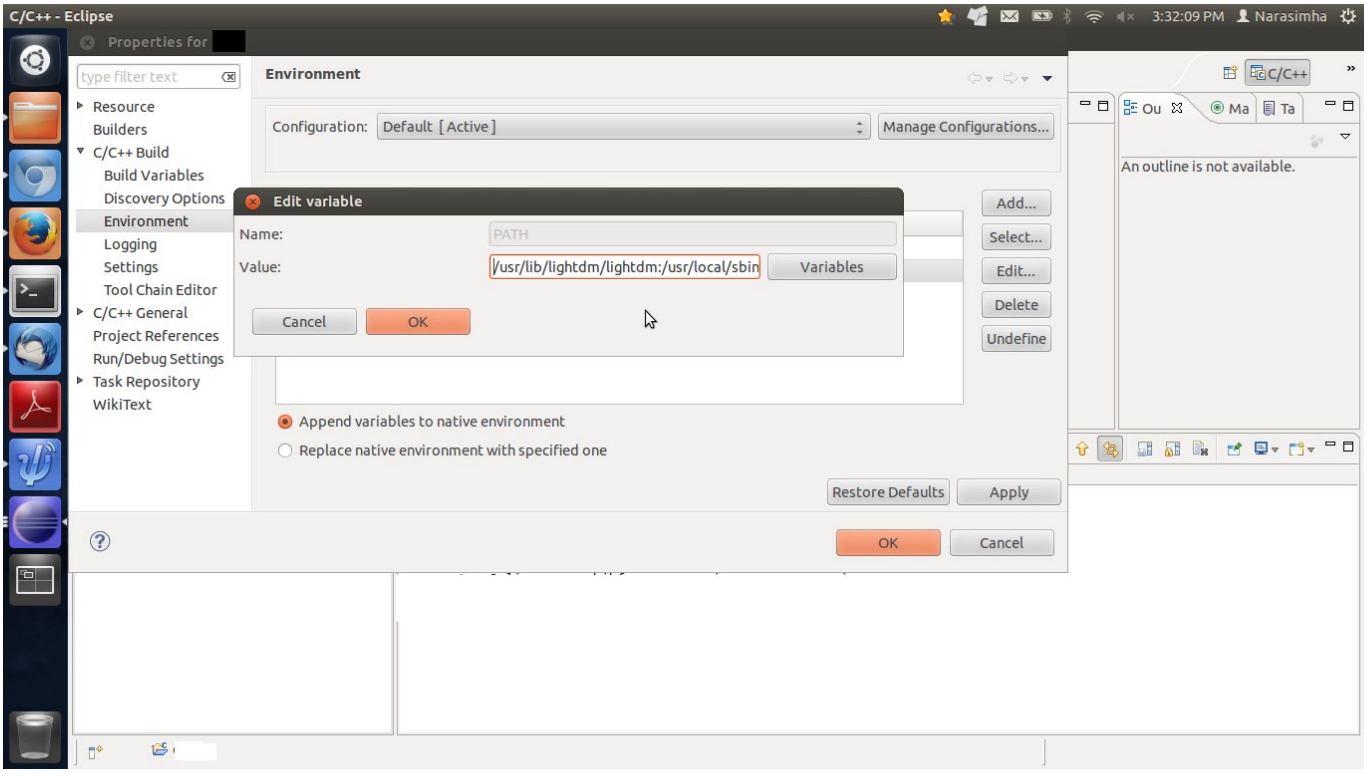
Into folder: GPIO Browse...

Options
□ Overwrite existing resources without warning
□ Create top-level folder
Advanced >>

< Back | Next > | Cancel | Finish

File Edit Source Refactor Navigate Search Project Run Window Help
11:30:26 AM
Narasimha
Project Explorer
GPIO
C/C++
Ou
Ma
Ta
An outline is not available.
/gpio-check (30/04/14 11:21 AM)

2. Select/Set Tool-chain PATH as shown below:



C/C++ - Eclipse

Properties for [redacted]

Environment

Configuration: Default [Active] Manage Configurations...

Edit variable

Name: PATH

Value: /usr/lib/lightdm/lightdm:/usr/local/sbin Variables

Cancel | OK

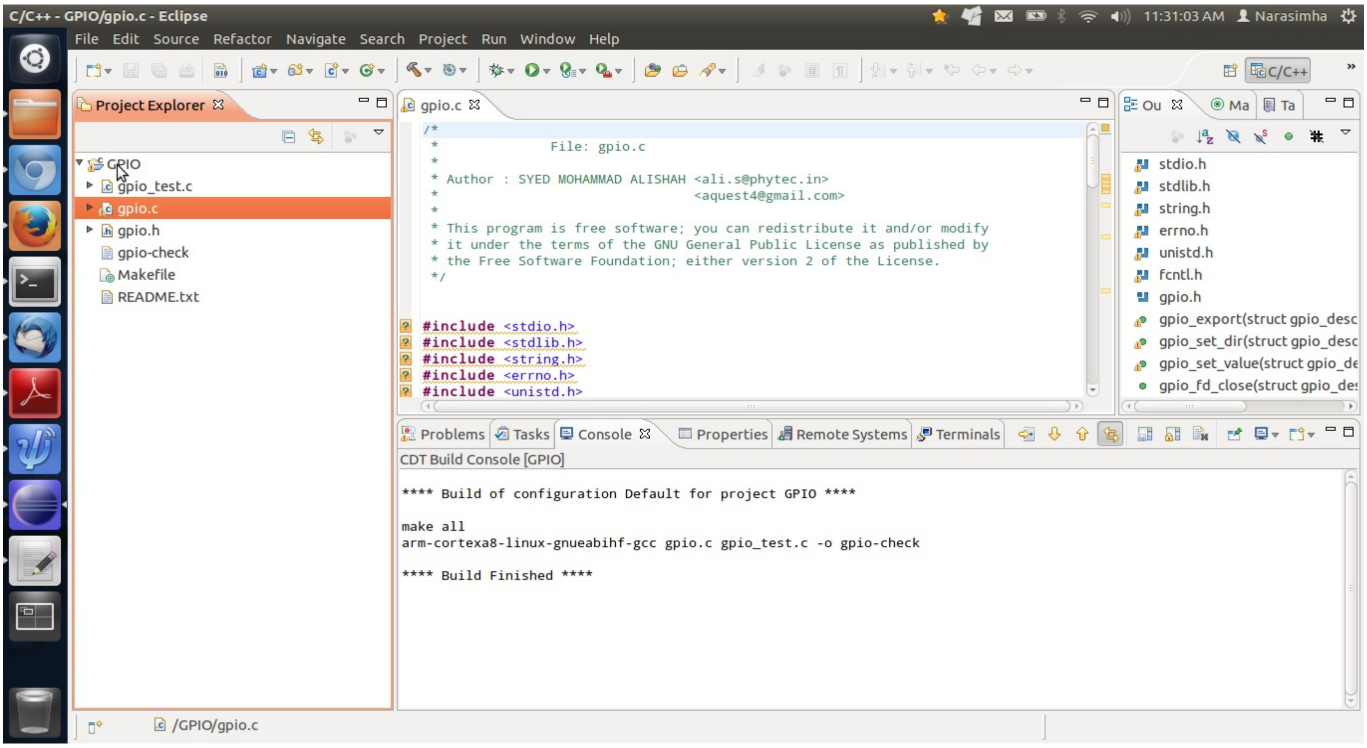
Add... | Select... | Edit... | Delete | Undefine

● Append variables to native environment
○ Replace native environment with specified one

Restore Defaults | Apply

OK | Cancel

3. Here, select the project and Build the project.



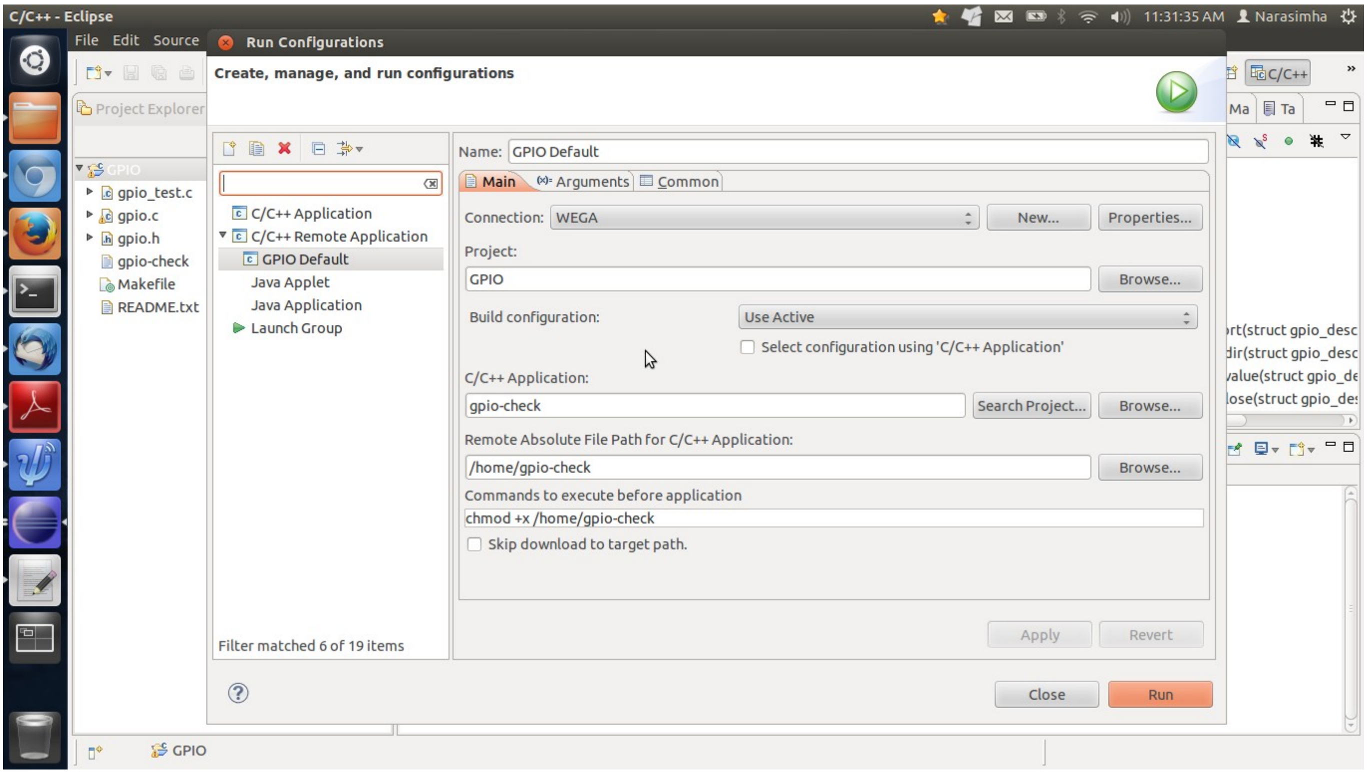
C/C++ - GPIO/gpio.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
11:31:03 AM Narasimha
Project Explorer
GPIO
gpio_test.c
gpio.c
gpio.h
gpio-check
Makefile
README.txt
gpio.c
Ou Ma Ta
stdio.h
stdlib.h
string.h
errno.h
unistd.h
fcntl.h
gpio.h
gpio_export(struct gpio_desc
gpio_set_dir(struct gpio_desc
gpio_set_value(struct gpio_de
gpio_fd_close(struct gpio_des
Problems Tasks Console Properties Remote Systems Terminals
/GPIO/gpio.c

```
/*
 *              File: gpio.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                                <aquest4@gmail.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
```

CDT Build Console [GPIO]

```
**** Build of configuration Default for project GPIO ****

make all
arm-cortexa8-linux-gnueabihf-gcc gpio.c gpio_test.c -o gpio-check

**** Build Finished ****
```

4. Right-Click on **GPIO** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:
[Note: Connection:**WEGA**, Project:**GPIO &** C/C++ Application:**gpio-check** etc.,**]**



Create, manage, and run configurations

Name: GPIO Default

Connection: WEGA

Project: GPIO

Build configuration: Use Active

□ Select configuration using 'C/C++ Application'

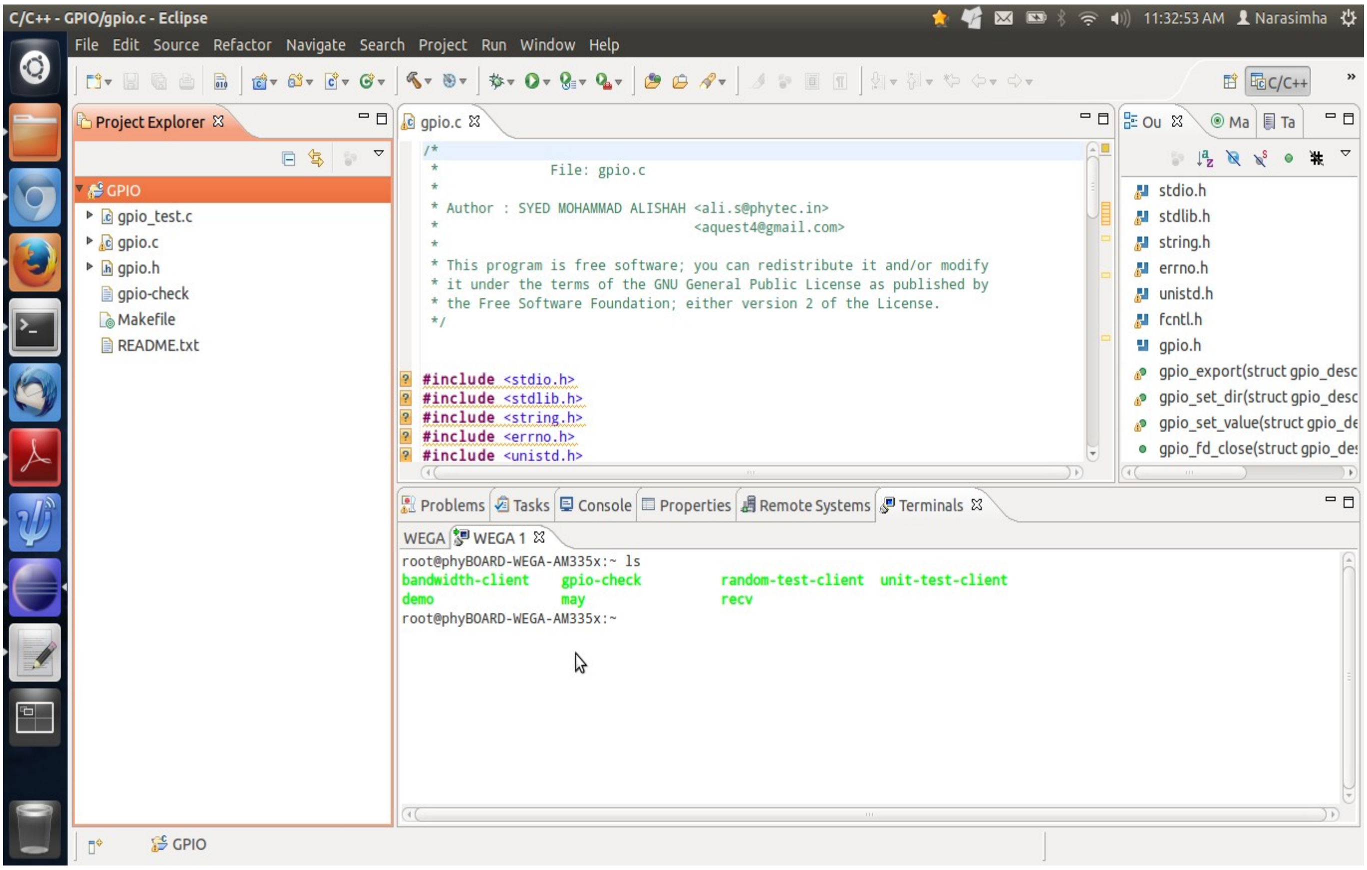
C/C++ Application: gpio-check

Remote Absolute File Path for C/C++ Application: /home/gpio-check

Commands to execute before application: chmod +x /home/gpio-check

□ Skip download to target path.

5. Below figure gives details about gpio-source, gpio-binary and the Remote Console-output.



```
/*
 *              File: gpio.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                                <aquest4@gmail.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    gpio-check          random-test-client  unit-test-client
demo                may                 recv
root@phyBOARD-WEGA-AM335x:~
```

2. UART APPLICATION

Contents:

1. UART Introduction:

Linux names its serial ports in the UNIX tradition. The first serial port has the file name /dev/ttyS0, the second serial port has the file name /dev/ttyS1, and so on.

<Board_Name> Linux-Kernel comes with UART Driver for user-selection. For more details of UART pins on Expansion see the <Board_Name>_Hardware_Manual.pdf

2. UART - Driver Configuration:

To add additional UART's, the pin-muxing in kernel board file need to be done. Follow the UART Section of <Board_Name>_System_Development_Guide.pdf

3. UART access from shell:

The UARTs can be known/viewed using sysfs from below instructions.

Ex: Enquire about tty devices,

```
$ ls /sys/class/tty/
```

Check for **ttyXX** enabled,

```
$ cd /sys/class/tty/ttyXX/
```

Configuring Serial-Port[ttyXX]:

```
 /* Issue below command to configure serial-port */

    $ minicom -s

 /*  minicom - Friendly Serial-Communication Program */
```

Note: Above commands can be used to access any UART.

4. UART access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

UART library and test program-files:

File-Name	Description
UART.c	Library file
UART.h	Library header
UART_test.c	Test application for UART library
Makefile	To build the UART test program.

UART API's for user programming :

Function Name	Description
UART_INIT	To initialize the UART
UART_CONF	To configure the UART
UART_WRITE	To write into UART buffer
UART_READ	To read from UART buffer

Code-Snippet:

```
    struct uart_config_t u1;
    struct uart_descriptor_t u2;

Functions:
    uart_init(&u1,&u2);

    uart_conf(&u1,&u2);

    uart_write(&u1,&u2);

    uart_read(&u1,&u2);
```

5. Test Procedure of UART on <BOARD_NAME> using command line:

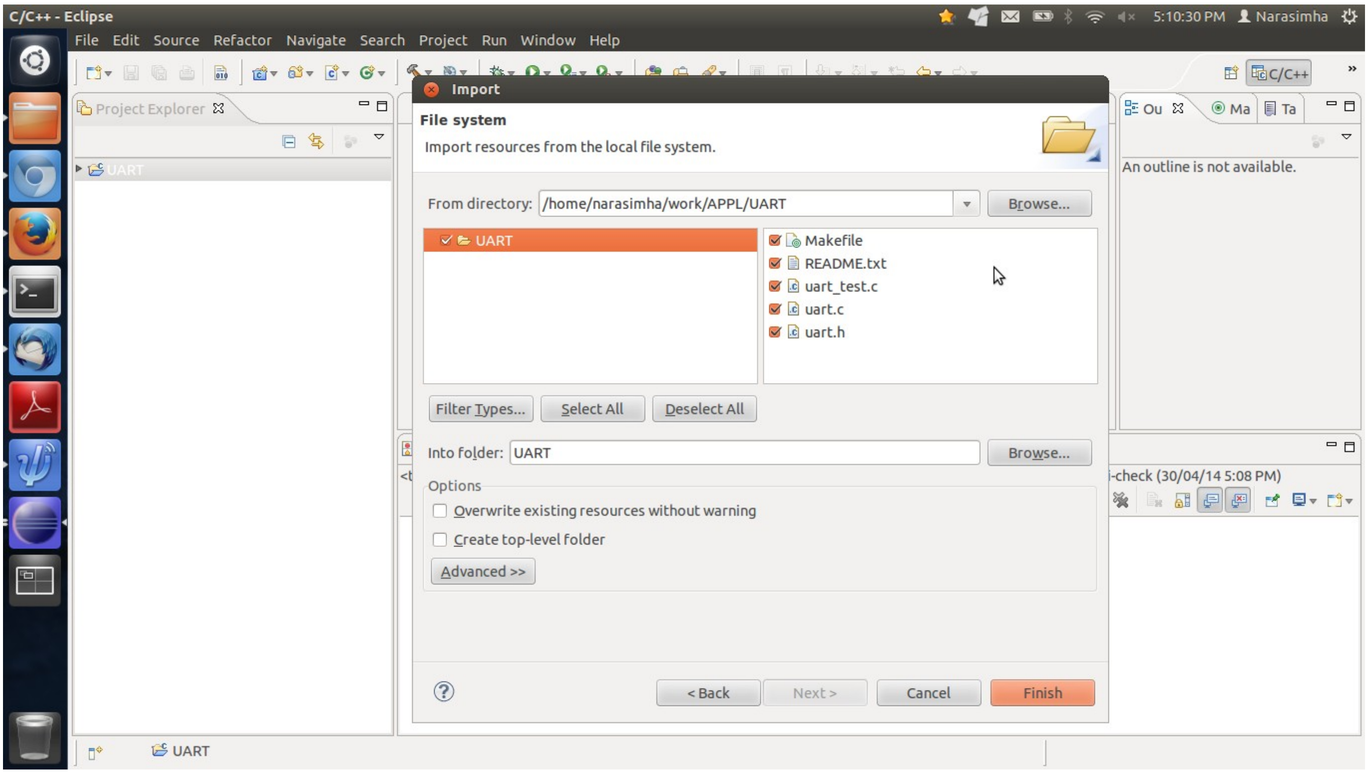
Procedure:

a. **Set the tool-chain path**
b. **Switch to the UART dir and run make command**
c. **Transfer the bin to the target using scp**
d. **Open the target shell and execute it.**
e. **Exit the target shell**

```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/UART
    $ make clean
    $ make
    $ scp uart-check root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
Syntax to Run:
    $ ./uart-check “/dev/ttyXX” “<some-data>”
For Ex:
    $ ./uart-check “/dev/ttyO1” “PHYTEC”
         [Note: Check serial terminal for data]
    $ exit
```

6. Test Procedure of UART on <BOARD_NAME> using Eclipse IDE:

1. Select all files as below and click Finish.



Import

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/UART

☑ UART

☑ Makefile
☑ README.txt
☑ uart_test.c
☑ uart.c
☑ uart.h

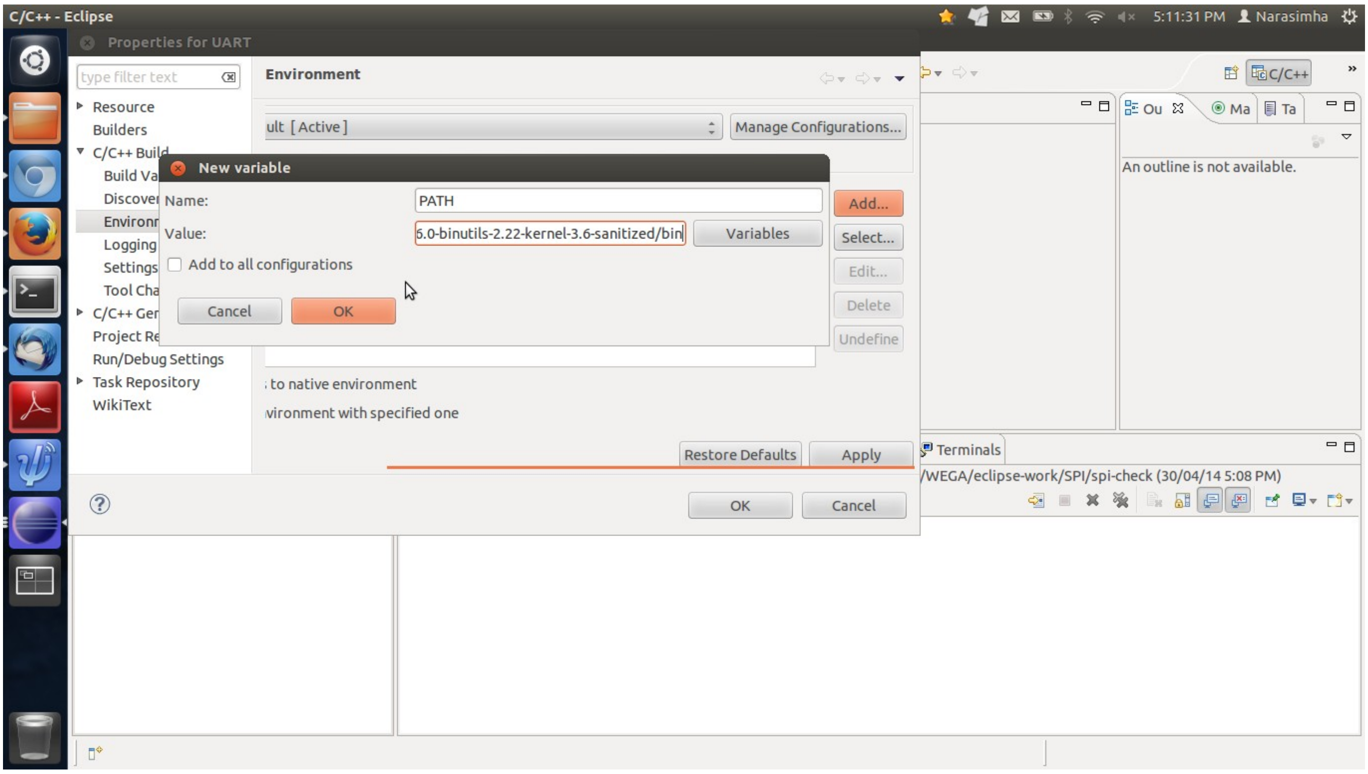
Into folder: UART

Options

□ Overwrite existing resources without warning
□ Create top-level folder

C/C++ - Eclipse
5:10:30 PM Narasimha
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
UART
Filter Types... Select All Deselect All
Browse...
Advanced >>
< Back Next > Cancel Finish
An outline is not available.
-check (30/04/14 5:08 PM)

2. Select/Set Tool-chain PATH as shown below:



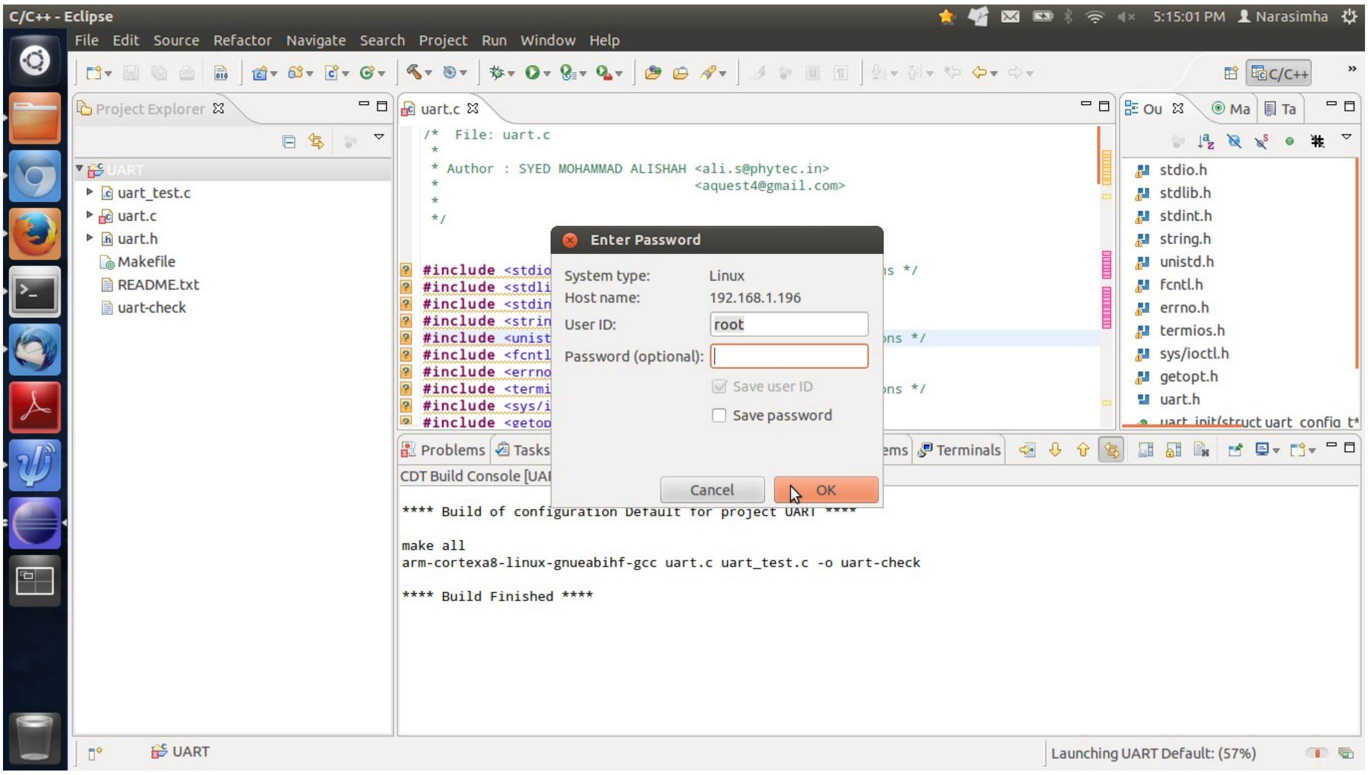
New variable

Name: PATH

Value: 6.0-binutils-2.22-kernel-3.6-sanitized/bin

□ Add to all configurations

3. Here, select the project and Build it as shown below.



```
/*  File: uart.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                                <aquest4@gmail.com>
 *
 *
 */
```

Enter Password

System type: Linux

Host name: 192.168.1.196

User ID: root

Password (optional):

☑ Save user ID

□ Save password

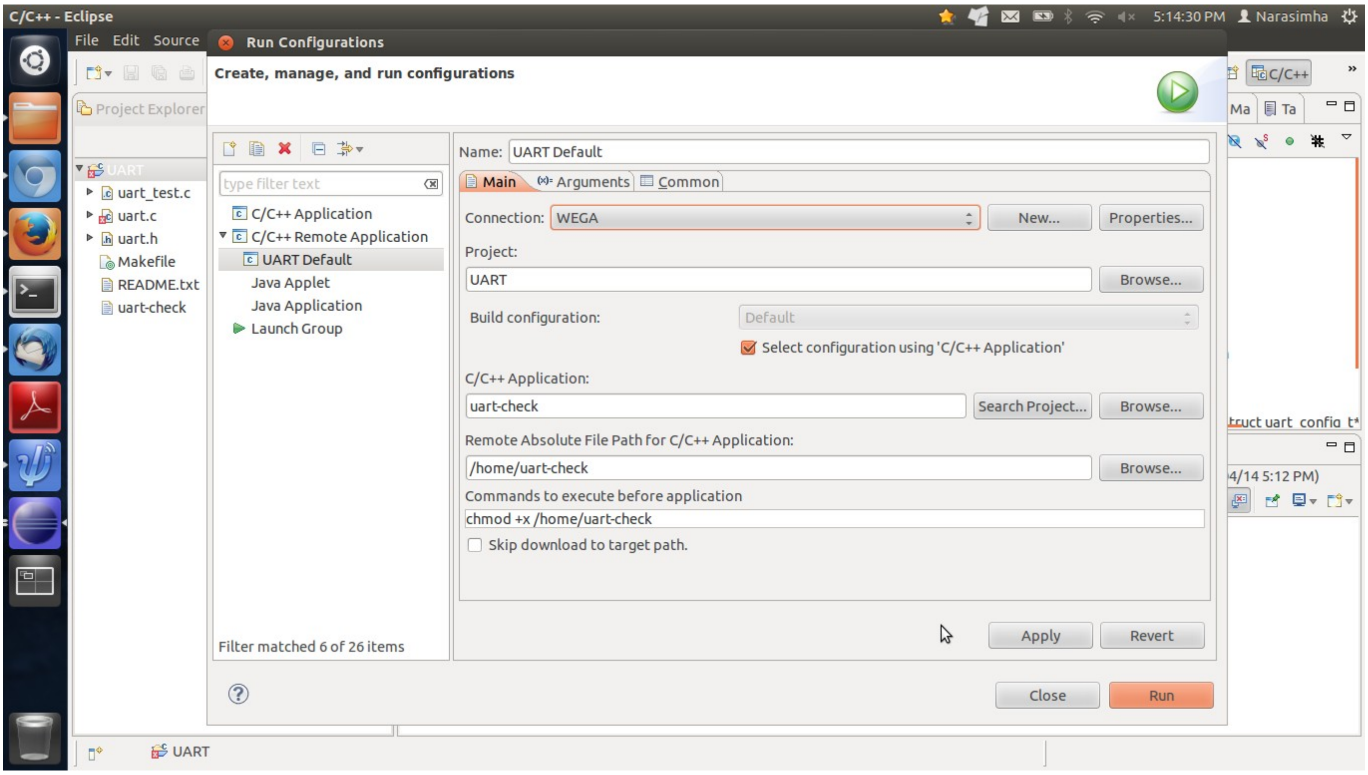
Cancel | OK

CDT Build Console [UART]

```
**** Build of configuration Default for project UART ****

make all
arm-cortexa8-linux-gnueabihf-gcc uart.c uart_test.c -o uart-check

**** Build Finished ****
```

Launching UART Default: (57%)

4. Right-Click on **UART** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:
[Note: Connection:**WEGA**, Project:**UART &** C/C++ Application:**uart-check** etc.,**]**



Run Configurations

Create, manage, and run configurations

- C/C++ Application
- C/C++ Remote Application
  - UART Default
- Java Applet
- Java Application
- Launch Group

Filter matched 6 of 26 items

Name: UART Default

Main | Arguments | Common

Connection: WEGA

Project: UART

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: uart-check

Remote Absolute File Path for C/C++ Application: /home/uart-check

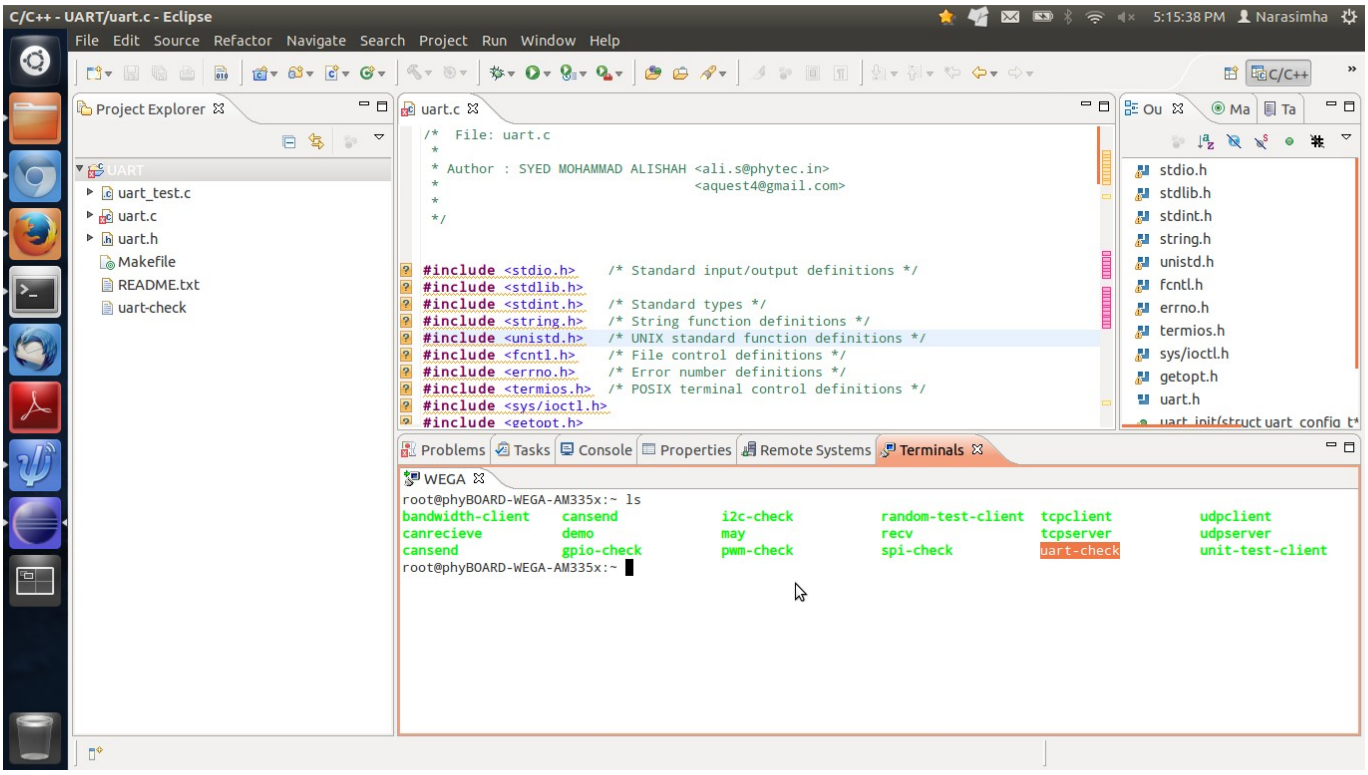
Commands to execute before application: chmod +x /home/uart-check

□ Skip download to target path.

Apply | Revert

Close | Run

5. Below figure gives details about UART-Source, UART-binary and the Remote Console-output.



```
/*  File: uart.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                                <aquest4@gmail.com>
 *
 *
 */


#include <stdio.h>    /* Standard input/output definitions */
#include <stdlib.h>
#include <stdint.h>   /* Standard types */
#include <string.h>   /* String function definitions */
#include <unistd.h>   /* UNIX standard function definitions */
#include <fcntl.h>    /* File control definitions */
#include <errno.h>    /* Error number definitions */
#include <termios.h>  /* POSIX terminal control definitions */
#include <sys/ioctl.h>
#include <getopt.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpclient           udpclient
canrecieve          demo                may                 recv                tcpserver           udpserver
cansend             gpio-check          pwm-check           spi-check           uart-check          unit-test-client
root@phyBOARD-WEGA-AM335x:~
```

3. I2C APPLICATION

Contents:

1. I2C Introduction:

The I²C bus is commonly used to connect relatively low-speed sensors and other peripherals to equipment varying in complexity from a simple to a full-on motherboard.

<Board_Name> Linux-Kernel comes with I2C Driver for user-selection, and the buses available are **I2C < 0|1|2 >**. For more details of I2C pins on Expansion see the <Board_Name>_Hardware_Manual.pdf

2. I2C - Driver Configuration:

To add additional I2C pins, the pin-muxing in kernel board file need to be done. Follow the I2C Section of <Board_Name>_System_Development_Guide.pdf

3. I2C access from shell:

The I2C's(EEPROM) can be accessed using sysfs from below instructions.

Ex:

For EEPROM Dir-access:

```
$ ls /sys/bus/i2c/devices/0-00XX/eeprom


# To write into eeprom
$ echo “<some-text>” > /sys/class/i2c/devices/0-00XX/eeprom



# To read from eeprom
$ cat /sys/class/i2c/devices/0-00XX/eeprom
```

Note: Above commands can be used to access I2C based devices by passing <some-test> into eeprom device.

4. I2C access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

I2C library and test program-files:

File-Name	Description
I2C.c	Library file
I2C.h	Library header
I2C_test.c	Test application for I2C library
Makefile	To build the I2C test program.

I2C API's for user programming :

Function Name	Description
I2C_FD_OPEN	To Initialize the I2C
I2C_WRITE_DATA	To Write Byte data into Register
I2C_READ_DATA	To Read Byte data from Register
I2C_FD_CLOSE	To close the I2C at the end of I2C Operations.

Code-Snippet:

```
i2c_desc.i2c_dev = "/dev/i2c-X”;   /* Where X : I2C Bus-No */



Functions:
i2c_fd_open(&i2c_desc);

i2c_write_data(&i2c_desc);

i2c_read_data(&i2c_desc);

i2c_fd_close(&i2c_desc);
```

5. Test Procedure of I2C on <BOARD_NAME> using command line:

Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the I2C dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/i2c
    $ make clean
    $ make
    $ scp i2c-check root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
```

Syntax for Run:

```
    $ ./i2c-check REG_VAL
```

[Note: 1. I2C Bus-no, Addr & Reg-Addr are already passed from program]

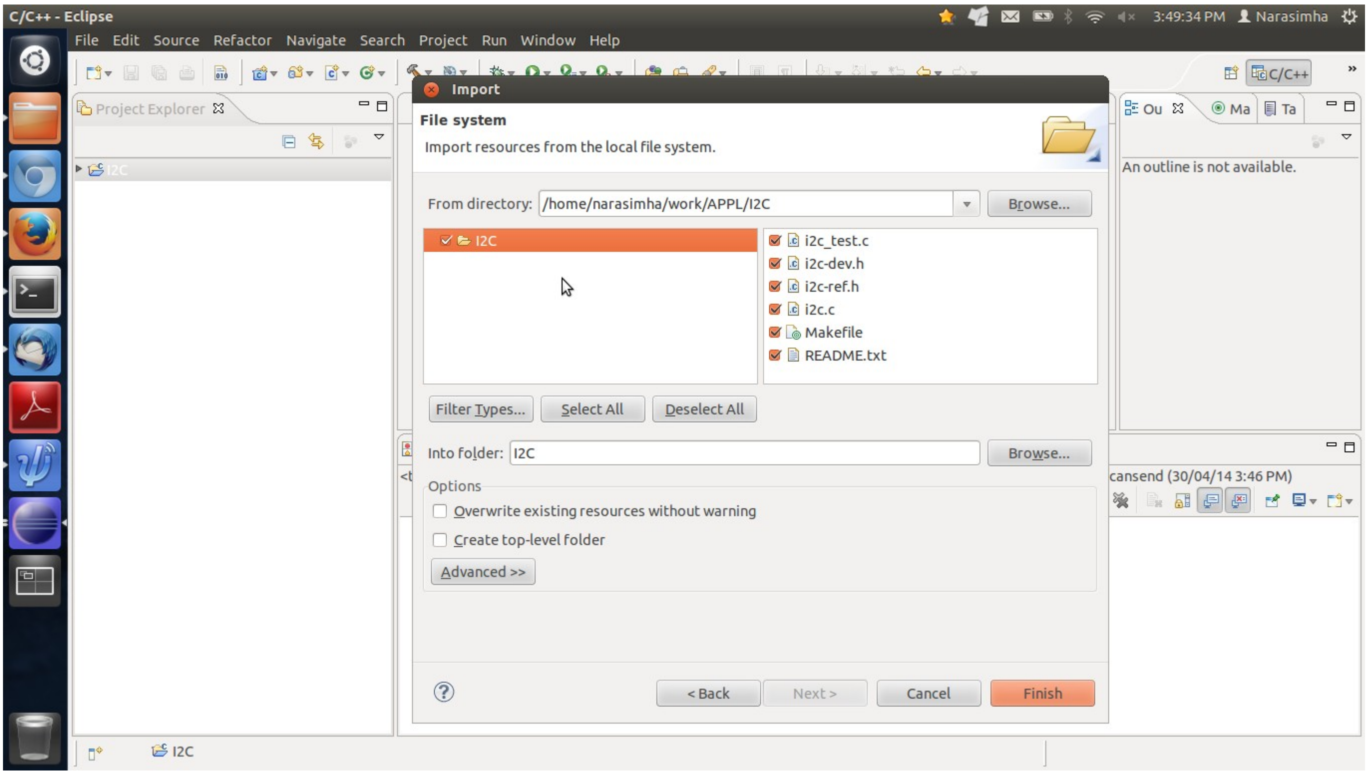
[Note: 2. Only REG_VAL should be passed as argv[1] from command-line]

Ex:

```
    $ ./i2c-check 0x04
    $ exit
```

6. Test Procedure of I2C on <BOARD_NAME> using Eclipse IDE:

1. Select all files as below and click Finish.



Import

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/I2C

☑ I2C

☑ i2c_test.c
☑ i2c-dev.h
☑ i2c-ref.h
☑ i2c.c
☑ Makefile
☑ README.txt

Into folder: I2C

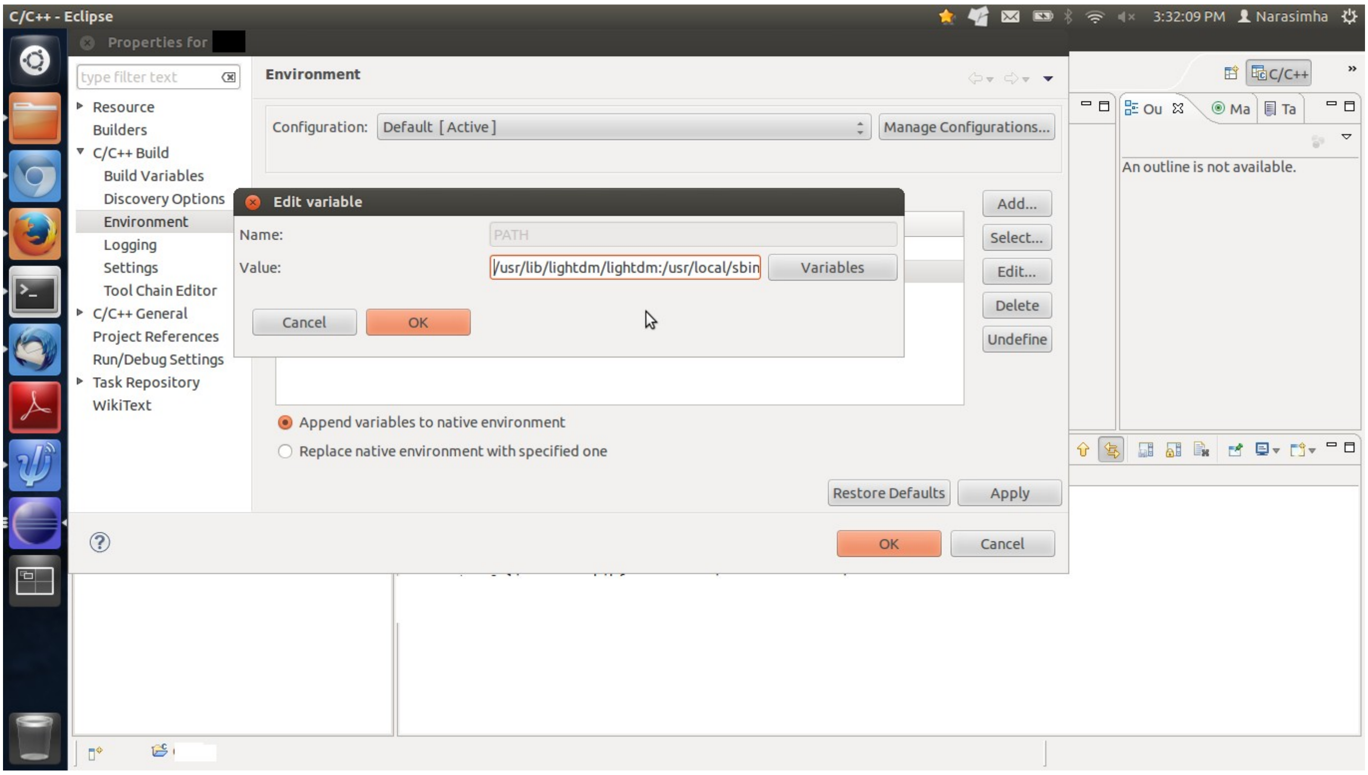
Options

□ Overwrite existing resources without warning

□ Create top-level folder

C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Filter Types... Select All Deselect All
Browse...
Advanced >>
< Back Next > Cancel Finish
An outline is not available.
cansend (30/04/14 3:46 PM)
3:49:34 PM Narasimha

2. Select/Set Tool-chain PATH as shown below:



Properties for [redacted]

Environment

Configuration: Default [Active]

Edit variable

Name: PATH

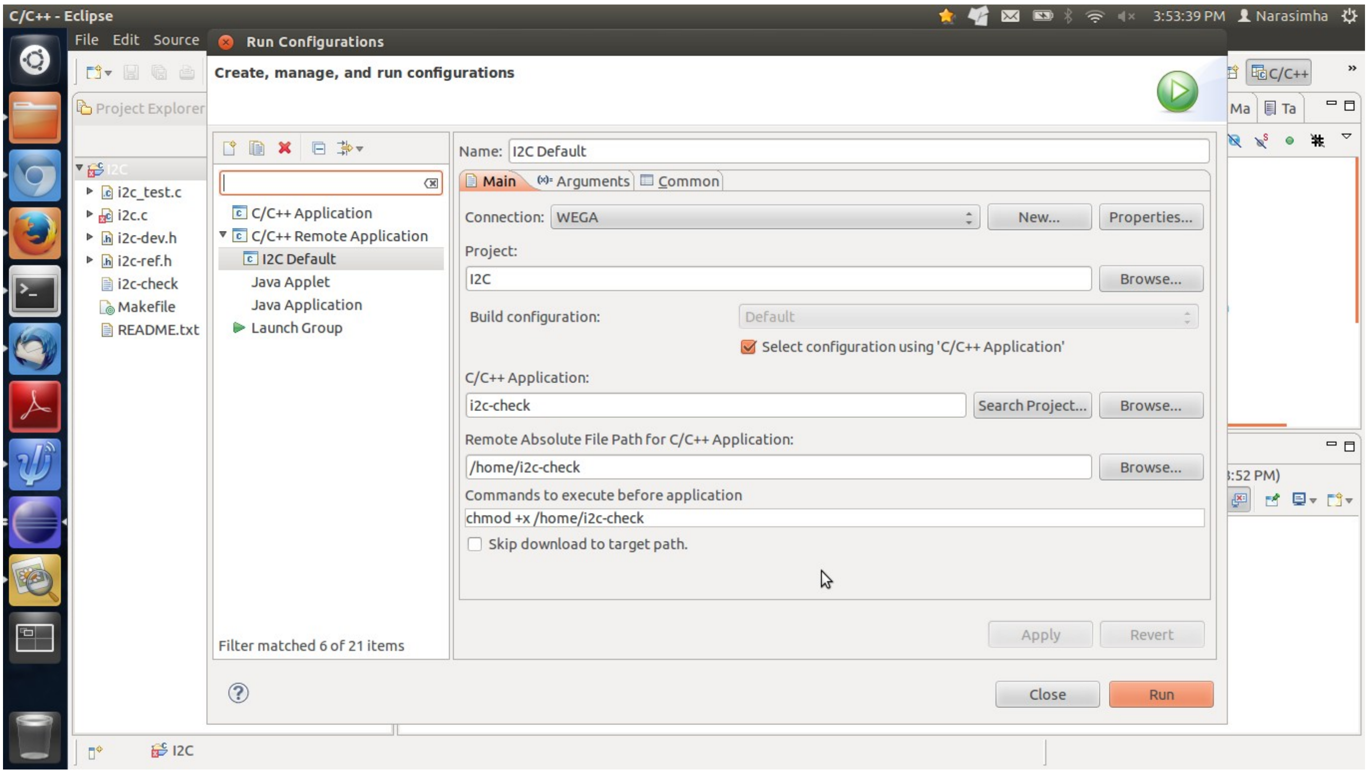
Value: /usr/lib/lightdm/lightdm:/usr/local/sbin

● Append variables to native environment

○ Replace native environment with specified one

3. Right-Click on I2C from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**I2C** **&** C/C++ Application:**i2c-check** etc.,**]**



Name: I2C Default

Connection: WEGA

Project: I2C

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

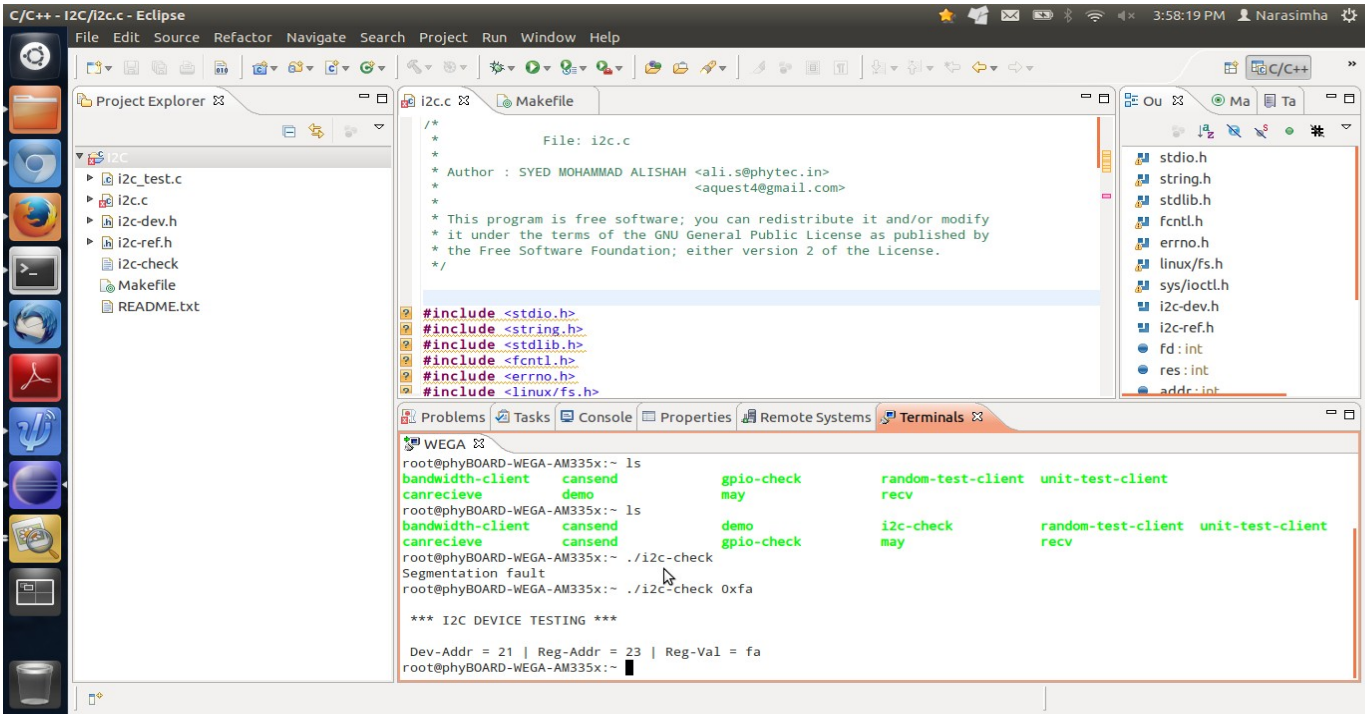
C/C++ Application: i2c-check

Remote Absolute File Path for C/C++ Application: /home/i2c-check

Commands to execute before application: chmod +x /home/i2c-check

☐ Skip download to target path.

4. Below figure gives details about i2c-source, i2c-binary and the Remote Console-output.



```
/*
 *              File: i2c.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                                <aquest4@gmail.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */


#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <linux/fs.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             gpio-check          random-test-client  unit-test-client
canrecieve          demo                may                 recv
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             demo                i2c-check           random-test-client  unit-test-client
canrecieve          cansend             gpio-check          may                 recv
root@phyBOARD-WEGA-AM335x:~ ./i2c-check
Segmentation fault
root@phyBOARD-WEGA-AM335x:~ ./i2c-check 0xfa

 *** I2C DEVICE TESTING ***

 Dev-Addr = 21 | Reg-Addr = 23 | Reg-Val = fa
root@phyBOARD-WEGA-AM335x:~
```

4. SPI APPLICATION

Contents:

1. SPI Introduction:

SPI (Synchronous Peripheral Interface) is a synchronous serial interface to connect peripheral chips like ADCs, EEPROMS, Sensors or other devices.

SPI works in master and slave mode, the master provides the clock signal and each slave has a dedicated chip-select.

<Board_Name> Linux-Kernel comes with SPI Driver for user-selection. For more details of SPI pins on Expansion see the <Board_Name>_Hardware_Manual.pdf

2. SPI - Driver Configuration:

<BOARD_NAME> The pin-muxing in kernel board file needed for SPI<0|1>.

SPI Interface with <BOARD_NAME> on the Expansion Connector can be accessed from userspace using the spidev Interface. Follow the SPI Section of <Board_Name>_System_Development_Guide.pdf

3. SPI access from shell:

The SPIs can be known/viewed using sysfs from below instructions.

```
$ ls /sys/class/spidev/spidevB.C
```

Note: i) B in spidev is Bus-no.
ii) C in spidev is Chip-select.
iii) async read/write is not available in userspace.

4. SPI access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

SPI library and test program-files:

File-Name	Description
spi.c	Library file
spi.h	Library header
spi_test.c	Test application for SPI library
Makefile	To build the SPI test program.

SPI API's for user programming :

Function Name	Description
SPI_OPEN	To open the SPI *Interface*
SPI_CONFIG	To Configure the SPI[mode, bits-per-word and speed]
SPI_WRITE	To write into write-buffer
SPI_READ	To read from read-buffer
SPI_CLOSE	To close the SPI at the end of SPI Operations.

Code-Snippet:

```
     spi_init(&spi_desc);

     spi_config(&spi_desc);

     txbuff[0] = spi_htoi(argv[1]);    /* value to be transmitted */

For Transaction:
     spi_trx(&spi_desc,1);

For Half-Duplex(Write/Read):
     spi_write(&spi_desc,txbuff,tx_len);
     spi_read(&spi_desc,rxBuff,rx_len);


     spi_free(&spi_desc);
```

5. Test Procedure of SPI on <BOARD_NAME> using command line:

Procedure:

a. **Set the tool-chain path**

b. **Switch to the SPI dir and run make command**

c. **Transfer the bin to the target using scp**

d. **Open the target shell and execute it.**

e. **Exit the target shell**

```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/spi
    $ make clean
    $ make
    $ scp spi-check root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
```

Syntax to Run:

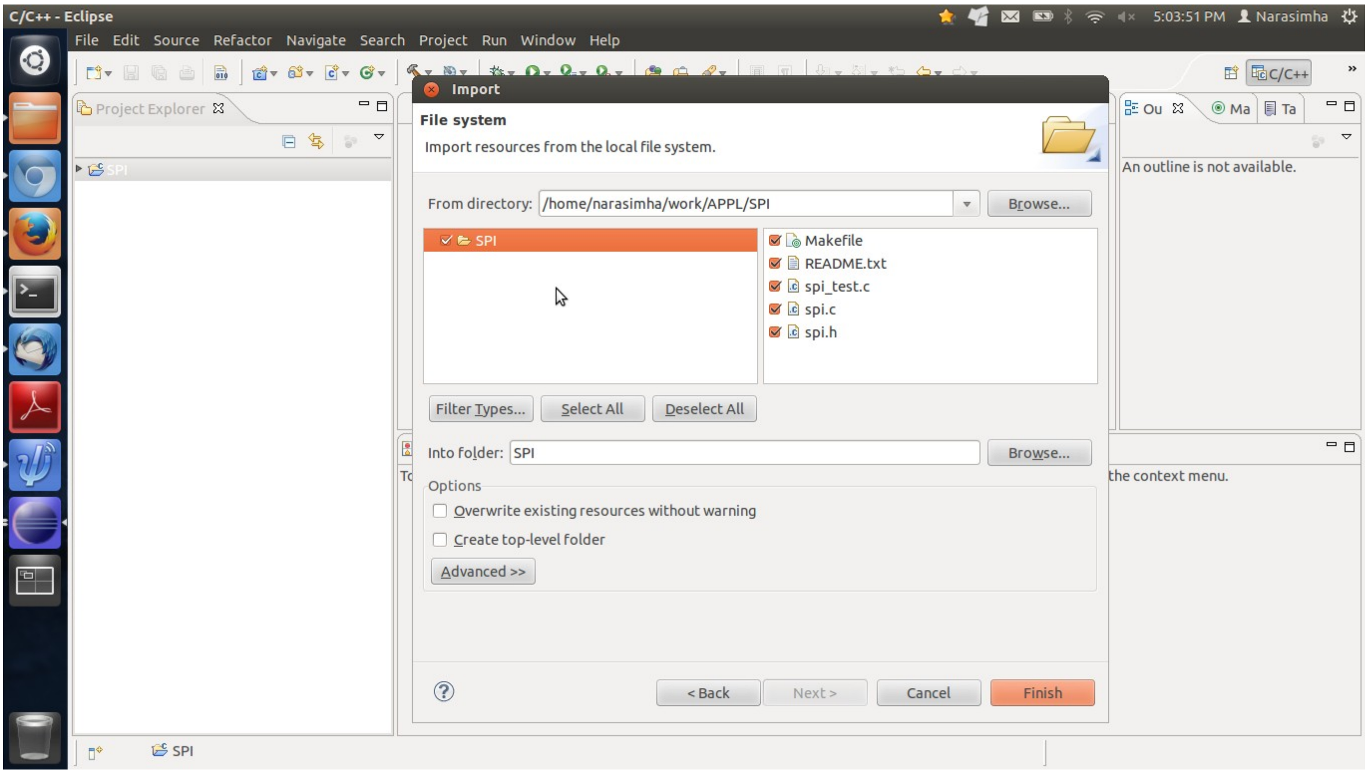
```
    $ ./spi-check <txbuff-value>
```

For Ex:

```
    $ ./spi-check 0xFA

    $ exit
```

6. Test Procedure of SPI on <BOARD_NAME> using Eclipse IDE:

1. After importing Watchdog-source from existing location(local storage), select all files as below and click Finish.



File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/SPI

☑ SPI

☑ Makefile
☑ README.txt
☑ spi_test.c
☑ spi.c
☑ spi.h

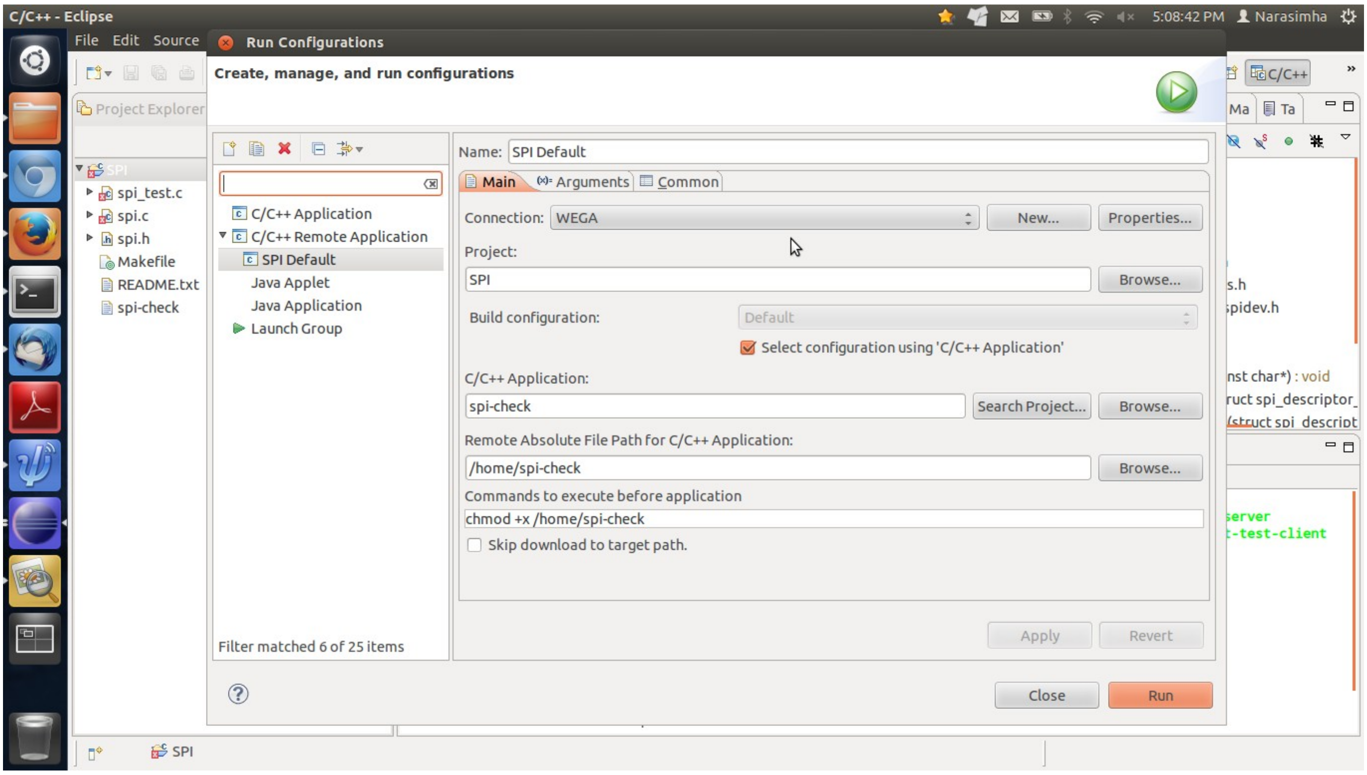
Into folder: SPI

Options

☐ Overwrite existing resources without warning

☐ Create top-level folder

2. Select/Set Tool-chain PATH as shown below:



Create, manage, and run configurations

Filter matched 6 of 25 items

Name: SPI Default

Connection: WEGA

Project: SPI

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: spi-check

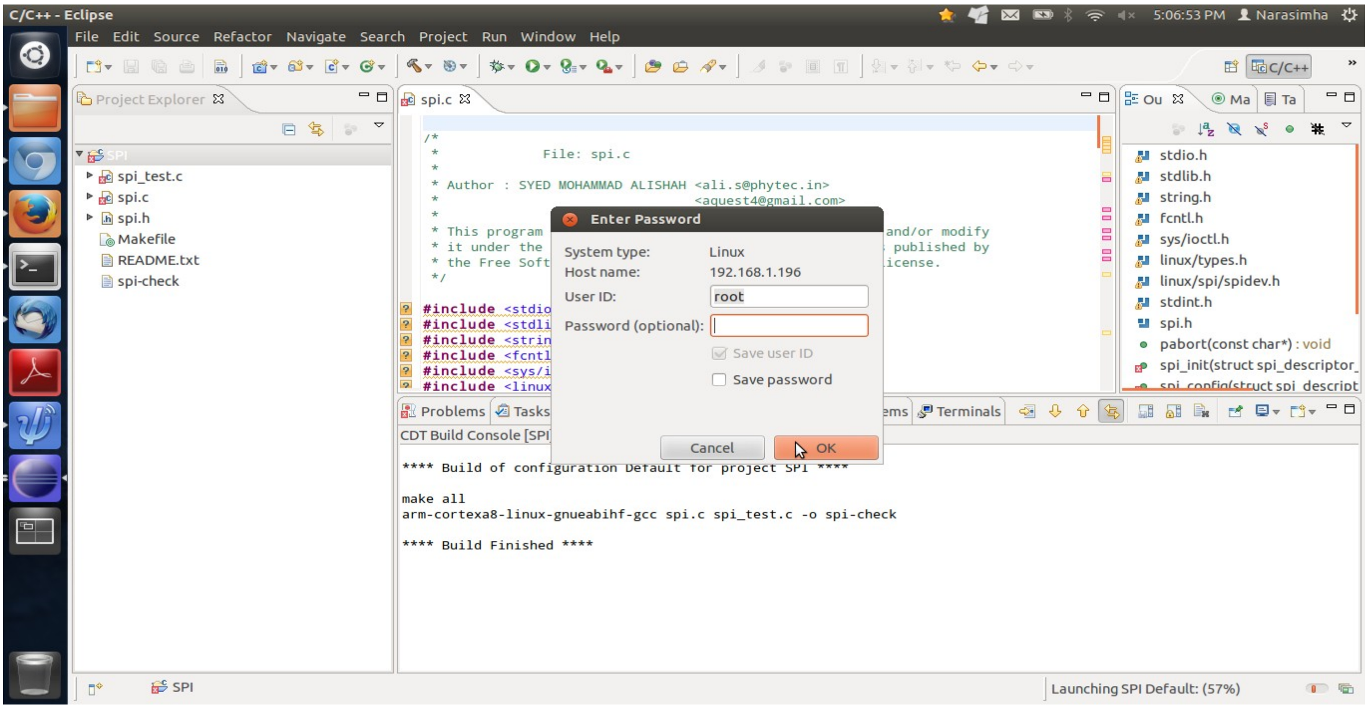
Remote Absolute File Path for C/C++ Application: /home/spi-check

Commands to execute before application: chmod +x /home/spi-check

☐ Skip download to target path.

3. Right-Click on **SPI** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**SPI** & C/C++ Application:**spi-check** etc.,**]**



Enter Password

System type: Linux

Host name: 192.168.1.196

User ID: root

Password (optional):

☑ Save user ID

□ Save password

Cancel OK

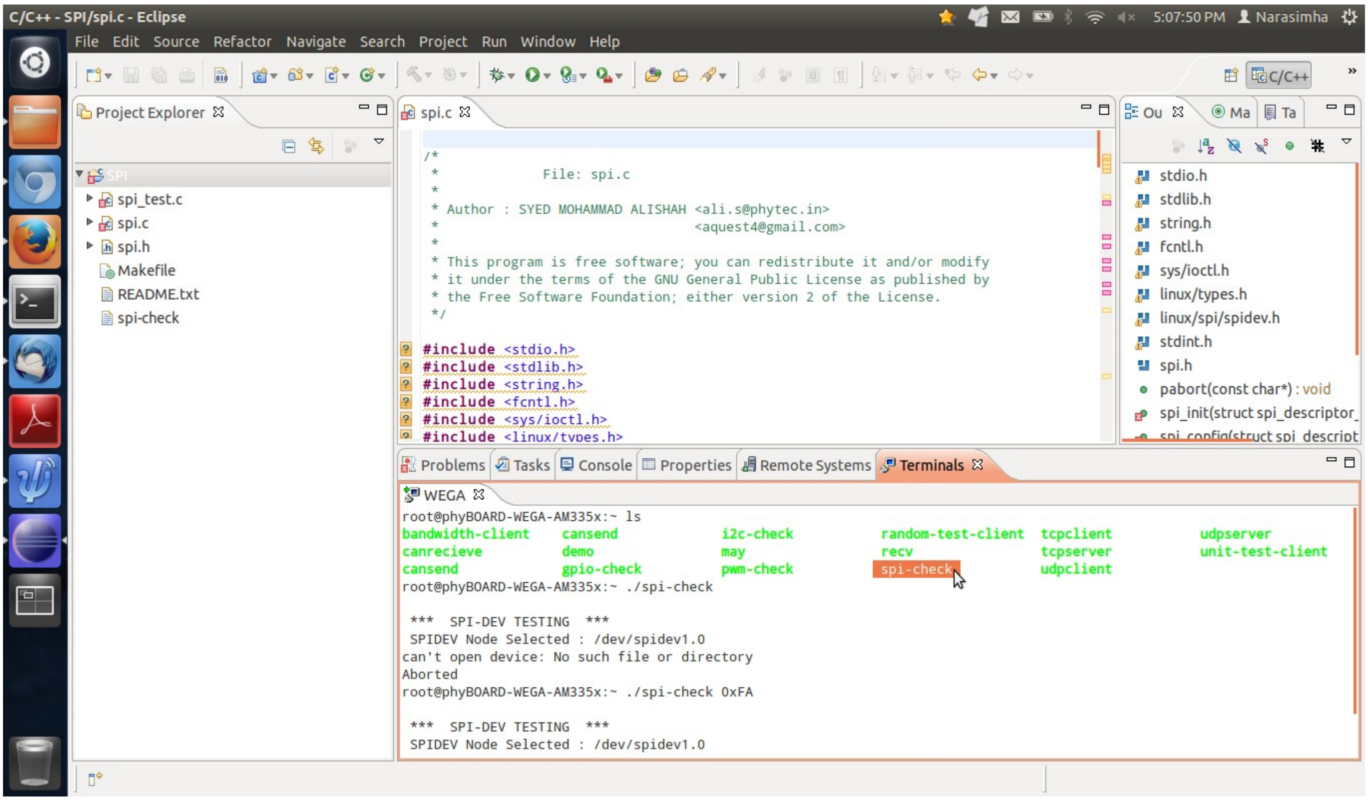
```
/*
 *           File: spi.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                               <aquest4@gmail.com>
```

CDT Build Console [SPI

```
**** Build of configuration Default for project SPI ****

make all
arm-cortexa8-linux-gnueabihf-gcc spi.c spi_test.c -o spi-check

**** Build Finished ****
```

Launching SPI Default: (57%)

4. This step provides info about the spi-source, spi-binary and Remote Console-output.



```
/*
 *           File: spi.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                               <aquest4@gmail.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
```

WEGA

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend     i2c-check   random-test-client  tcpclient   udpserver
canrecieve          demo        may         recv                tcpserver   unit-test-client
cansend             gpio-check  pwm-check   spi-check           udpclient
root@phyBOARD-WEGA-AM335x:~ ./spi-check

 ***  SPI-DEV TESTING  ***
 SPIDEV Node Selected : /dev/spidev1.0
can't open device: No such file or directory
Aborted
root@phyBOARD-WEGA-AM335x:~ ./spi-check 0xFA

 ***  SPI-DEV TESTING  ***
 SPIDEV Node Selected : /dev/spidev1.0
```

5. PWM APPLICATION

Contents:

1. PWM Introduction:

Pulse-width modulation(**PWM**), is a modulation technique that conforms the width of the pulse, based on modulator signal information.

<Board_Name> Linux-Kernel PWM Driver selection. For more details of PWM - see the <Board_Name>_Hardware_Manual.pdf

2. PWM - Driver Configuration:

To set PWM and eCAP.x, pin-muxing in kernel board file need to be done. Follow the PWM Section of <Board_Name>_System_Development_Guide.pdf

3. PWM access from shell:

The PWM can be accessed from sysfs from below instructions.

a. Request:	**/sys/class/pwm/ecap:x/request**
b. Run:	**/sys/class/pwm/ecap:x/run**
c. Period_frequency:	**/sys/class/pwm/ecap:x/period_frequency**
d. Duty Cycle:	**/sys/class/pwm/ecap:x/duty_percent**

Ex:

```
# To request the device,
$ echo 1 > /sys/class/pwm/ecap:x/request
$ ls /sys/class/pwm/ecap:x/request

# To start the PWM
$ echo 1 > /sys/class/pwm/ecap:x/run       /* echo 0 to stop */
$ ls /sys/class/pwm/ecap:x/run

# To set the Period Frequency
$ echo 50 > /sys/class/pwm/ecap:x/period_freq


# To set the Duty-cycle
$ echo 10 > /sys/class/pwm/ecap:x/duty_percent
```

Note: Above commands can be used to access pwm by modifying the various pwm attributes.

4. PWM access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

PWM library and test program-files:

File-Name	Description
pwm.c	Library file
pwm.h	Library header
pwm_test.c	Test application for PWM library
Makefile	To build the pwm test program.

PWM API's for user programming :

Function Name	Description
PWM_ON	To request the pwm
PWM_START	To start with pwm Operations
PWM_PERIOD_FREQ	To set the period freq
PWM_DUTY_CYCLE	To set the duty-cycle
PWM_OFF	To free the device request and stop the pwm.

Code-Snippet:

```
    pwm_on(&pwm_desc);
    pwm_start(&pwm_desc);
    pwm_period_freq(&pwm_desc);
    pwm_duty_cycle(&pwm_desc);
    pwm_off(&pwm_desc);
```

5. Test Procedure of PWM on <BOARD_NAME> using command line:

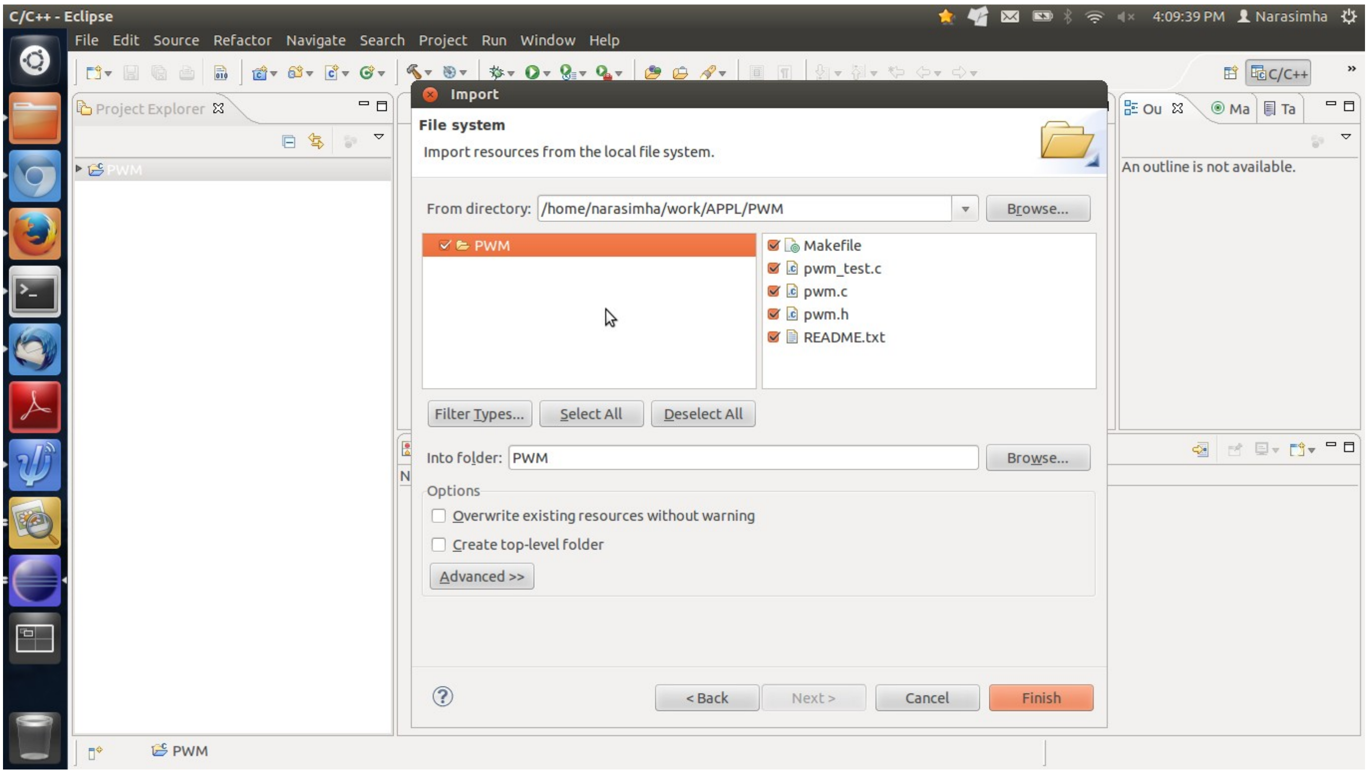
Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the pwm dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

```
   $ export PATH=$PATH:<the path of toolchain bin>
   $ cd <code_base>/app/pwm
   $ make clean
 [Syntax to Compile: $ make CC=<compiler>]
   $ make CC=arm-cortexa8-linux-gnueabi-gcc

   $ scp pwm-check root@<BOARD_NAME>:/home/
   $ ssh root@<BOARD IP-address>

[Syntax to Run: $ ./pwm-check <freq> <duty-cycle> <interface>]

   $ ./pwm_test 50 10 ecap.2

   $ exit
```

6. **Test Procedure of PWM on <BOARD_NAME> using Eclipse IDE**:

1. After importing PWM-source from existing location(local storage), select all files as below and click Finish.



Import

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/PWM

☑ PWM

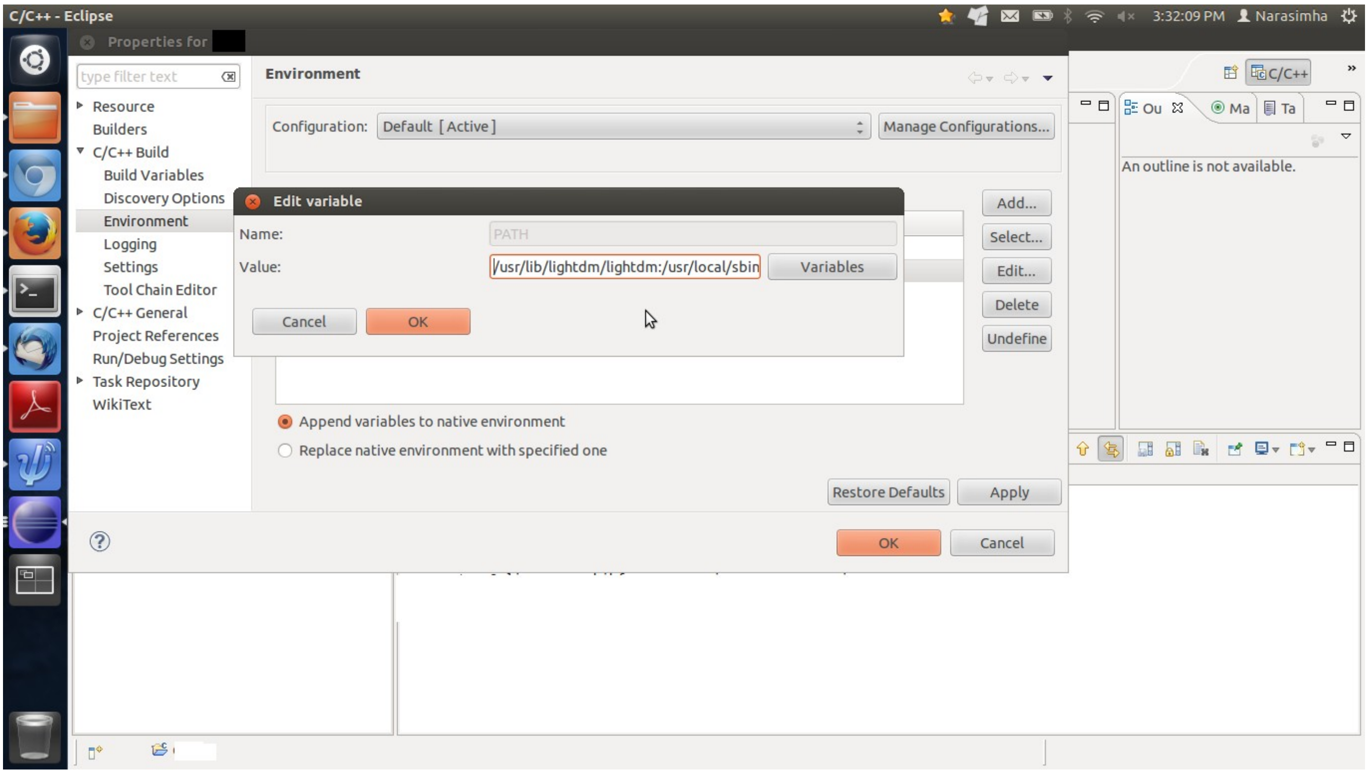
☑ Makefile
☑ pwm_test.c
☑ pwm.c
☑ pwm.h
☑ README.txt

Into folder: PWM

Options

□ Overwrite existing resources without warning
□ Create top-level folder

2. Select/Set Tool-chain PATH as shown below:



Properties for [redacted]

Environment

Configuration: Default [Active]

Edit variable

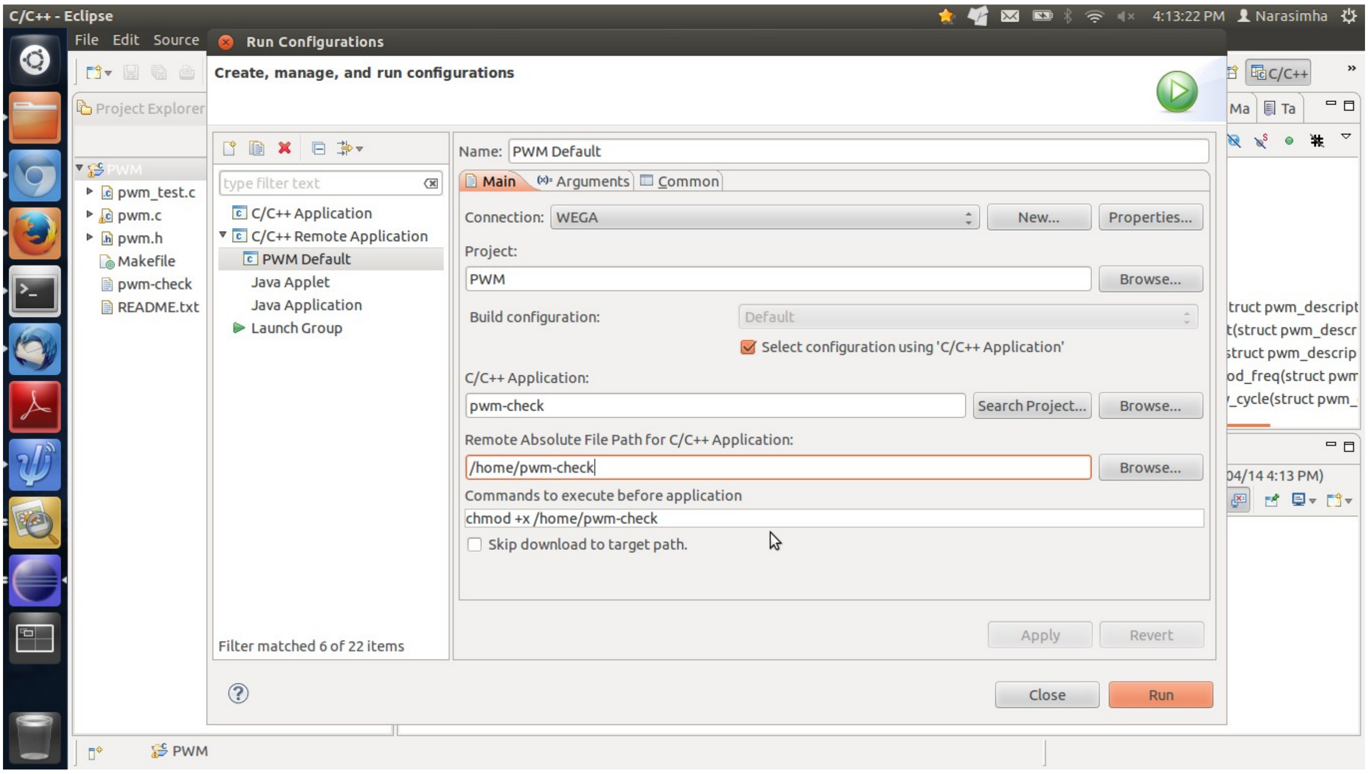
Name: PATH

Value: /usr/lib/lightdm/lightdm:/usr/local/sbin

● Append variables to native environment
○ Replace native environment with specified one

3. Right-Click on **PWM** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**PWM &** C/C++ Application:**pwm-check** etc.,**]**



Run Configurations

Create, manage, and run configurations

Name: PWM Default

Connection: WEGA

Project:

PWM

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application:

pwm-check

Remote Absolute File Path for C/C++ Application:

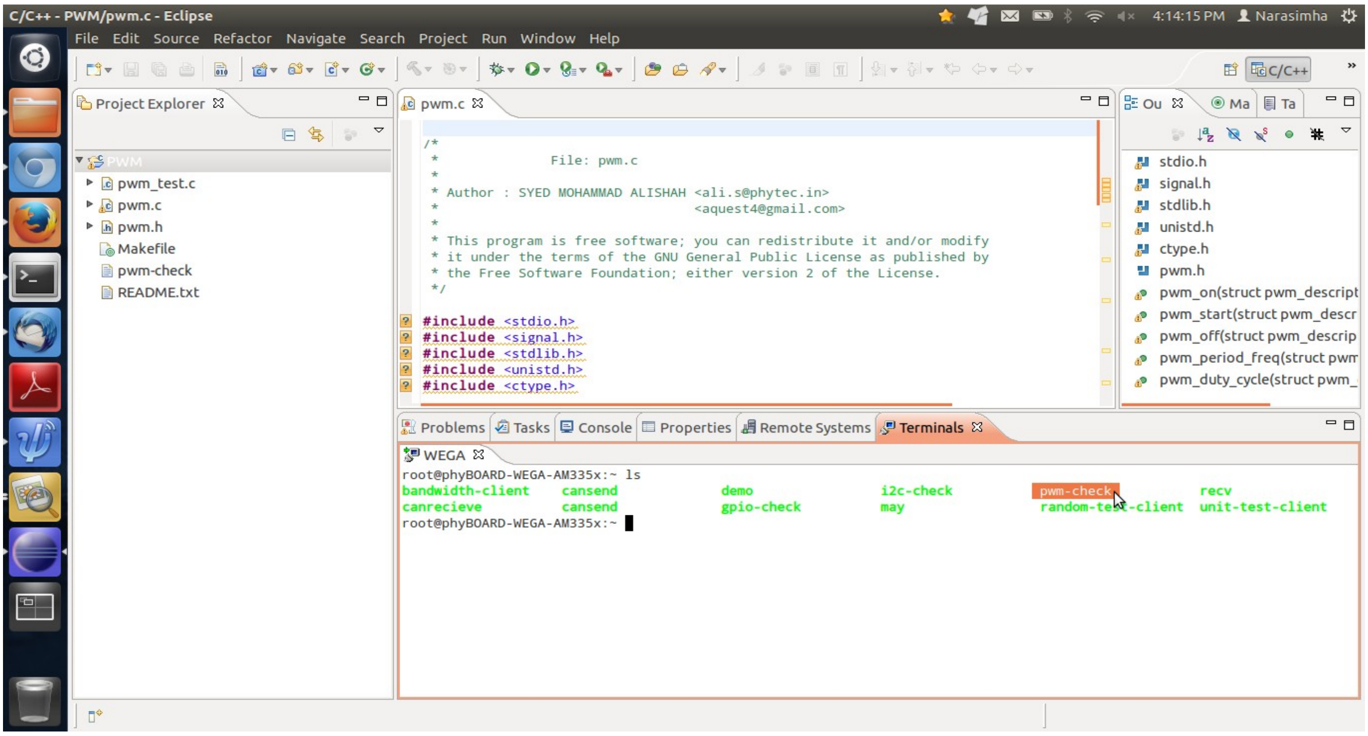
/home/pwm-check

Commands to execute before application

chmod +x /home/pwm-check

☐ Skip download to target path.

4. Below figure gives details about pwm-source, pwm-binary and the Remote Console-output.



```
/*
 *              File: pwm.c
 *
 * Author : SYED MOHAMMAD ALISHAH <ali.s@phytec.in>
 *                               <aquest4@gmail.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend          demo             i2c-check        pwm-check        recv
canrecieve          cansend          gpio-check       may              random-test-client  unit-test-client
root@phyBOARD-WEGA-AM335x:~ 
```

6. WATCHDOG APPLICATION

Contents:

1. WATCHDOG Introduction:

A Watchdog Timer(WDT) is a hardware circuit that can reset the computer system in case of a software fault.

<Board_Name> Linux-Kernel comes with WATCHDOG Driver for user-selection. For more details on WATCHDOG see the <Board_Name>_Hardware_Manual.pdf

2. WATCHDOG - Driver Configuration:

<Board_Name>_Board has a 32-bit Watchdog Timer, when the **/dev/watchdog** is opened it will reboot the system unless a userspace daemon resets the timer at regular intervals under certain timeout-period.

Default timeout of this Driver is *60 seconds*.

Follow WATCHDOG Section of<Board_Name>_System_Development_Guide.pdf

3. WATCHDOG access from shell:

The WATCHDOG can be using sysfs from below instructions.

Entry: /sys/class/WATCHDOG/Watdhdog0

```
$ ls /sys/class/watchdog/
```

```
$ ls /sys/class/watchdog/watchdog0
```

Note: Above commands can be used to access WATCHDOG.

4. WATCHDOG access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

WATCHDOG library and test program-files:

File-Name	Description
wdt.c	Library file
wdt.h	Library header
wdt_test.c	Test application for WATCHDOG Library
Makefile	To build the WATCHDOG test program.

WATCHDOG API's for user programming :

Function Name	Description
wdt_open	To Open the WATCHDOG for operations
wdt_config	To Configure the Watchdog Timer To set the timeout on the with the SETTIMEOUT ioctl, used value - **WDIOC_SETTIMEOUT** To query the current timeout using the GETTIMEOUT ioctl, used value - **WDIOC_GETTIMEOUT**
wdt_write	To write the new Watchdog-Timer Value
wdt_close	To close the WATCHDOG at the end of WATCHDOG Operations.

Code-Snippet:

```
struct wdt_descriptor_t wdt_desc;

Functions:
    wdt_open(&wdt_desc);

    wdt_config(&wdt_desc);

    wdt_write(&wdt_desc);

    wdt_close(&wdt_desc);
```

5. Test Procedure of WATCHDOG on <BOARD_NAME> using command line:

Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the WATCHDOG dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/watchdog
    $ make clean
    $ make
    $ scp wdt-check root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
```

Syntax to Run:

```
    $ ./wdt-check <a value less-than the watchdog reset time>
                              (or)
    $ ./wdt-check <a value greater-than the watchdog reset time>
```

For EX:

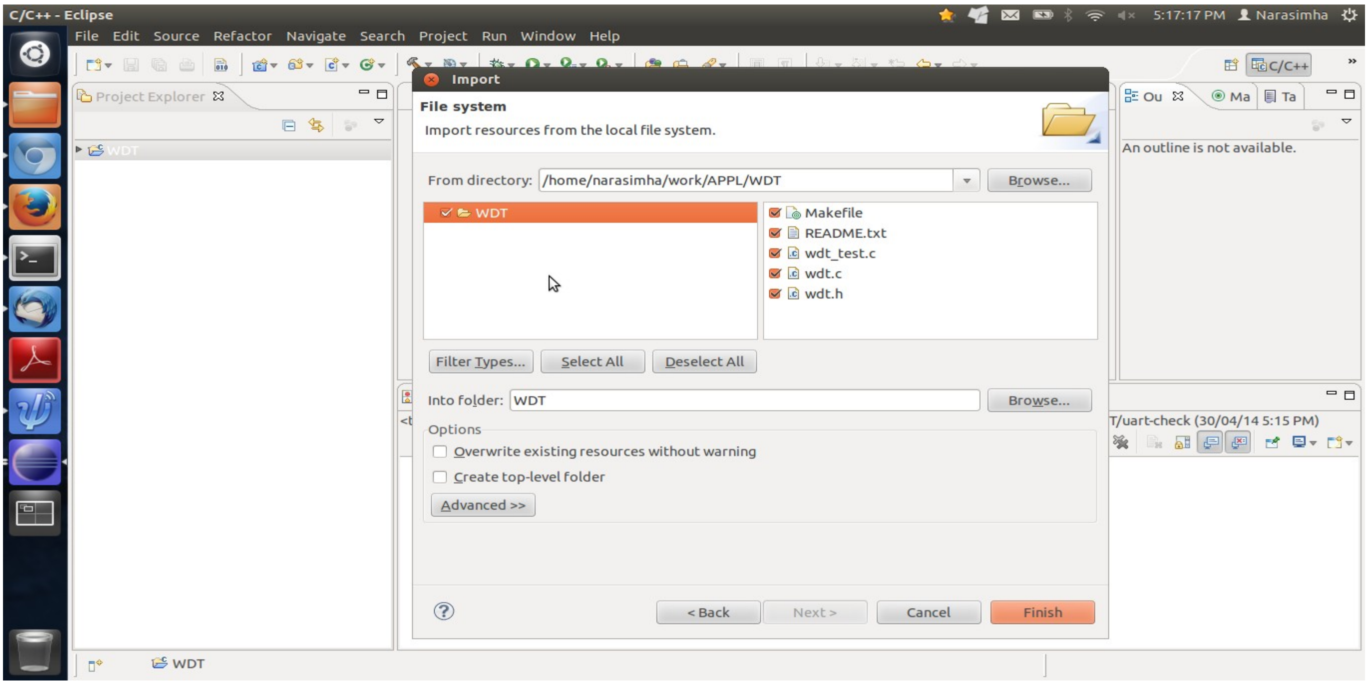
```
    $ ./wdt-check 6
```

[Note: “6” is greater-than the set-value(5), so will reboot the <Board>]

```
    $ exit
```

6. Test Procedure of WATCHDOG on <BOARD_NAME> using Eclipse IDE:

1. After importing Watchdog-source from existing location(local storage), select all files as below and click Finish.


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
5:17:17 PM Narasimha
Project Explorer
WDT
Ou Ma Ta
C/C++
An outline is not available.
T/uart-check (30/04/14 5:15 PM)

Import

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/WDT

☑ WDT

☑ Makefile
☑ README.txt
☑ wdt_test.c
☑ wdt.c
☑ wdt.h

Filter Types... Select All Deselect All
Browse...

Into folder: WDT

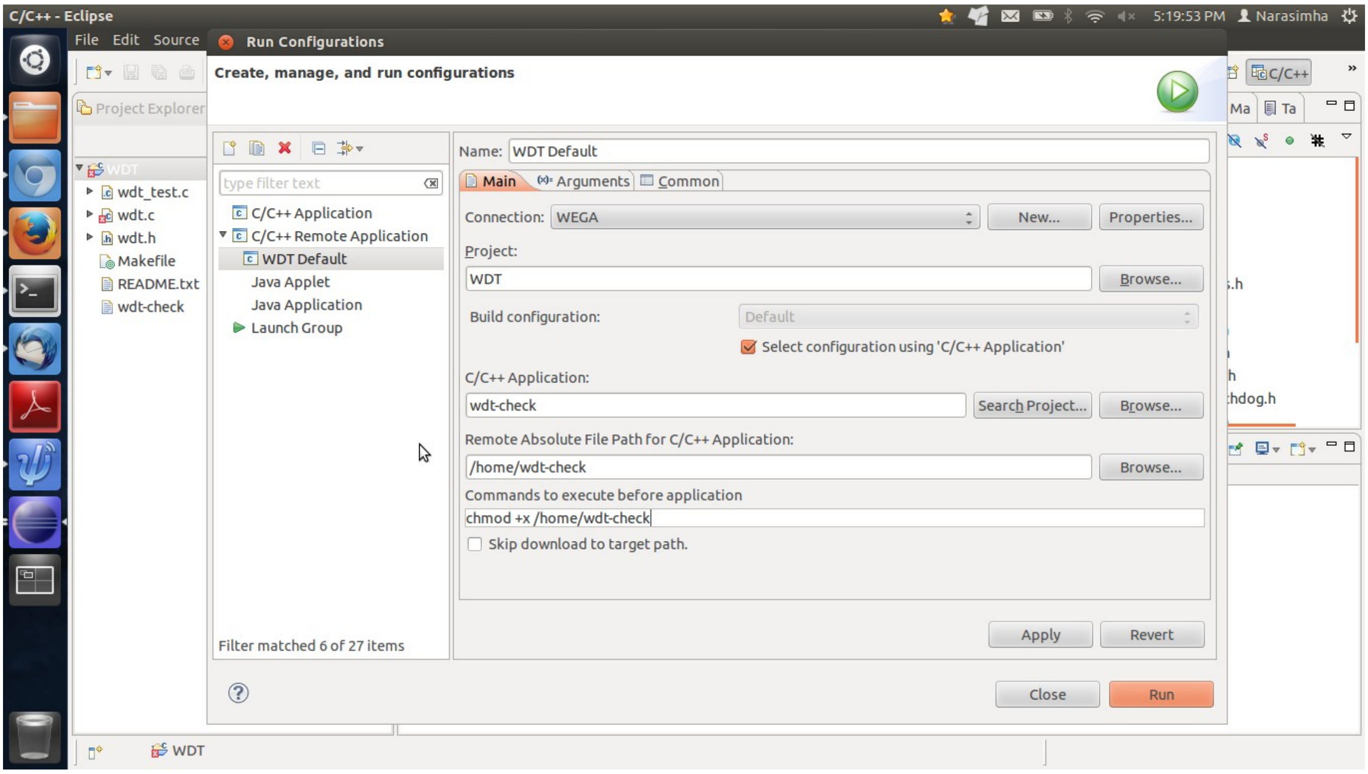
Options

□ Overwrite existing resources without warning

□ Create top-level folder

Advanced >>
< Back Next > Cancel Finish

2. Select/Set Tool-chain PATH as shown below:



Run Configurations

Create, manage, and run configurations

Filter matched 6 of 27 items

Name: WDT Default

Connection: WEGA

Project: WDT

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: wdt-check

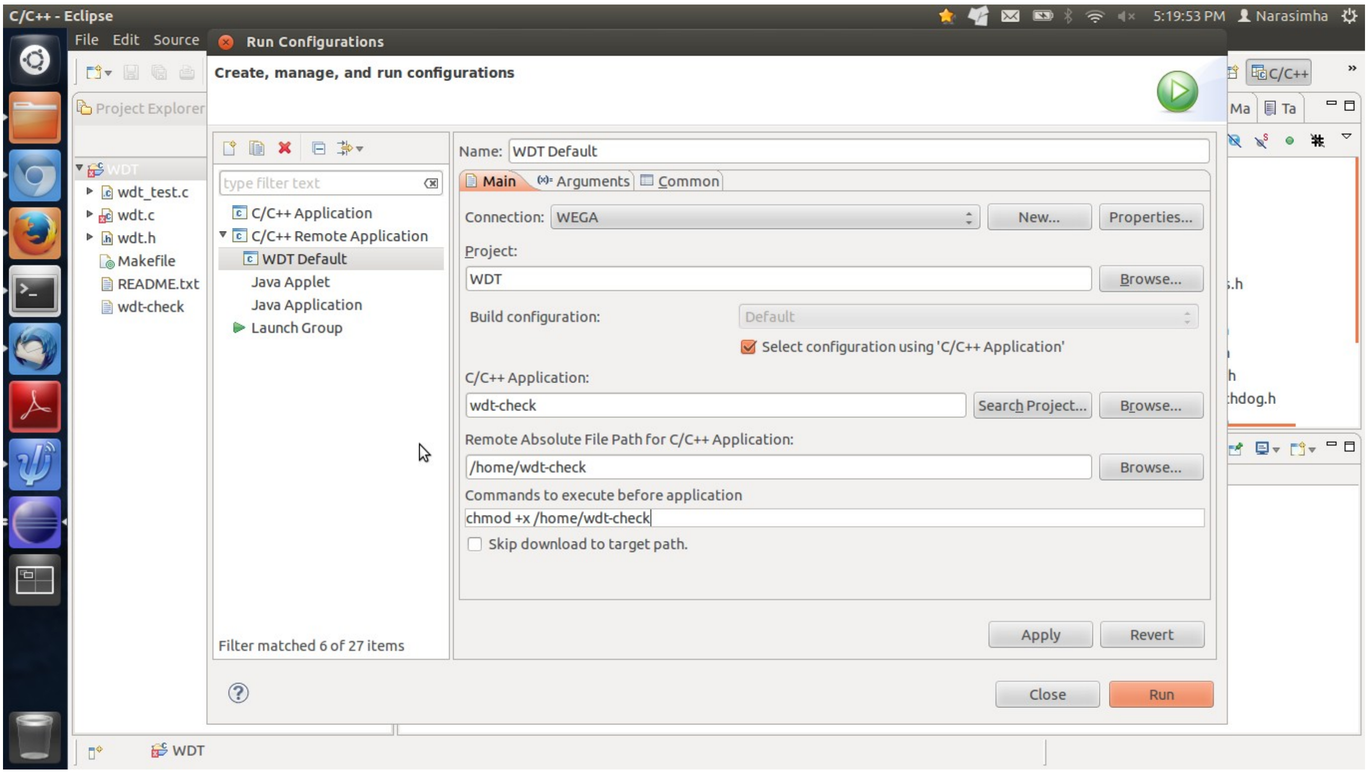
Remote Absolute File Path for C/C++ Application: /home/wdt-check

Commands to execute before application: chmod +x /home/wdt-check

□ Skip download to target path.

3. Right-Click on **WDT** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**WDT &** C/C++ Application:**wdt-check** etc.,**]**



Run Configurations

Create, manage, and run configurations

Name: WDT Default

Main | Arguments | Common

Connection: WEGA

Project: WDT

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: wdt-check

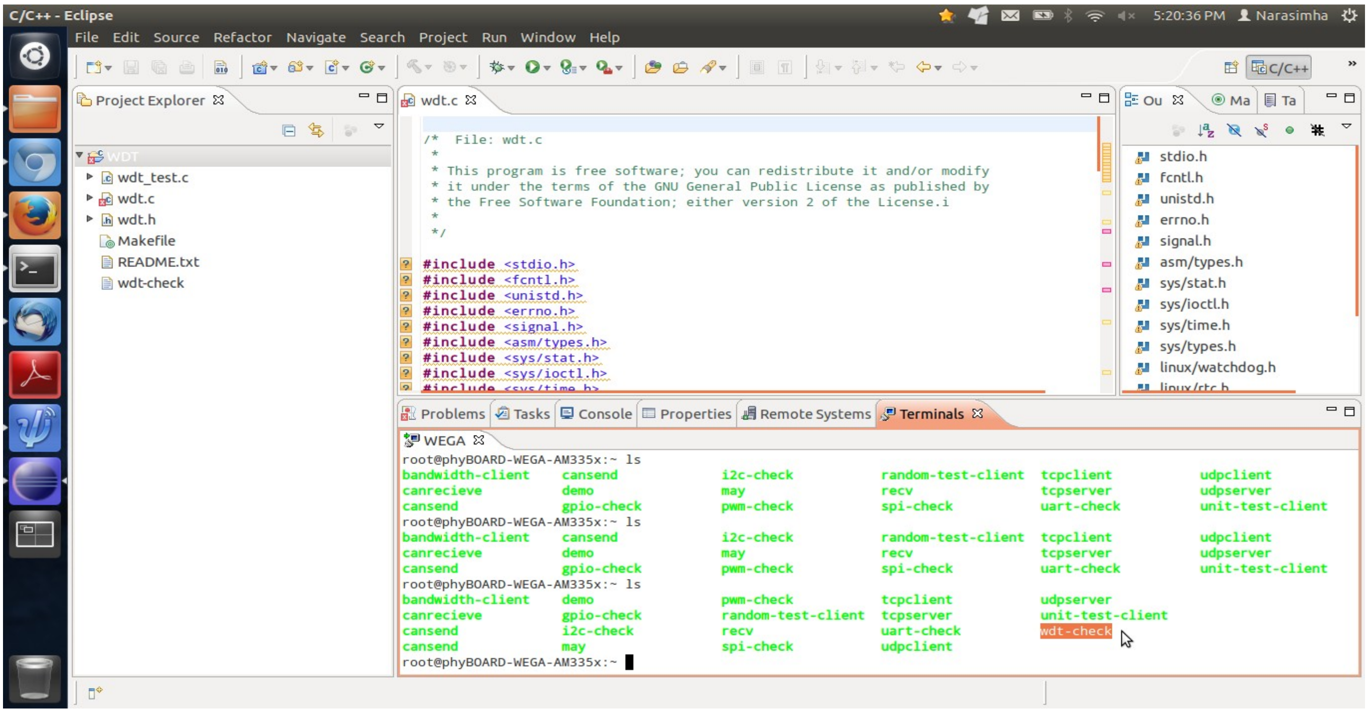
Remote Absolute File Path for C/C++ Application: /home/wdt-check

Commands to execute before application: chmod +x /home/wdt-check

□ Skip download to target path.

Filter matched 6 of 27 items

4. This step provides info about Project-Building, and details about the watchdog-source, watchdog-binary and Remote Console-output.



```
/*  File: wdt.c
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.i
 *
 */

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <asm/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/time.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpclient           udpclient
canrecieve          demo                may                 recv                tcpserver           udpserver
cansend             gpio-check          pwm-check           spi-check           uart-check          unit-test-client
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpclient           udpclient
canrecieve          demo                may                 recv                tcpserver           udpserver
cansend             gpio-check          pwm-check           spi-check           uart-check          unit-test-client
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    demo                pwm-check           tcpclient           udpserver
canrecieve          gpio-check          random-test-client  tcpserver           unit-test-client
cansend             i2c-check           recv                uart-check          wdt-check
cansend             may                 spi-check           udpclient
root@phyBOARD-WEGA-AM335x:~ █
```

7. TCP-SOCKET APPLICATION

Contents:

1. SOCKET Introduction:

A Socket is an end point of communication between two systems on a network. To be a bit precise, a socket is a combination of IP address and port on one system.

<Board_Name> Linux-Kernel comes with SOCKET Driver for user-selection. For more details of Ethernet(10/100 MB/s) for SOCKET Connections, see the <Board_Name>_Hardware_Manual.pdf

2. SOCKET - Driver Configuration:

For **[Ethernet(10/100) - RMII]** selection, the pin-muxing in kernel board file need to be done. Follow the SOCKET Section of <Board_Name>_System_Development_Guide.pdf

3. TCP-SOCKET access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

TCP-SOCKET library and test program-files:

File-Name	Description
tcpserver.c	Server file
tcplient.c	Client file
Makefile	To build the SOCKET Test Programs.

TCP-SOCKET API's for user programming:

[server-side]

Function Name	Description
SOCKET	To create the SOCKET
BIND	Bind a name to a SOCKET
LISTEN	Listen for connections on a SOCKET
ACCEPT	Accept a connection on a SOCKET
RECV	Receive a message from a Client
CLOSE	Close the SERVER-SOCKET

[client-side]

Function Name	Description
SOCKET	To create the SOCKET
CONNECT	Initiate a connection on a SOCKET
SEND	Send a message to Server
CLOSE	Close the CLIENT-SOCKET

Code-Snippet:

```
Server:
 Sd = socket(PF_INET,SOCK_STREAM,0)
 bind(Sd,(struct sockaddr  *)&server,sizeof(server))
 listen(Sd,<Backlog>)
 /* Backlog defines maximum length -- queue of pending Connections */

 accept(sd,(struct sockaddr *)&client,&length))
 recv(<socket-desc>,<buffer>,<buff-len>,<flag>))
 close(Sd)

Client:
 Sd = socket(PF_INET,SOCK_STREAM,0)
 connect(Sd,(struct sockaddr *)&server,sizeof(server))
 send(<socket-desc>,<buff>,<buff-length>,<flag>)
 close(Sd)
```

4. Test Procedure of TCP-SOCKET on <BOARD_NAME> using command line:

Procedure:

a. **Set the tool-chain path**

b. **Switch to the TCP-SOCKET dir and run make command**

c. **Transfer the bin to the target using scp**

d. **Open the target shell and execute it.**

e. **Exit the target shell**

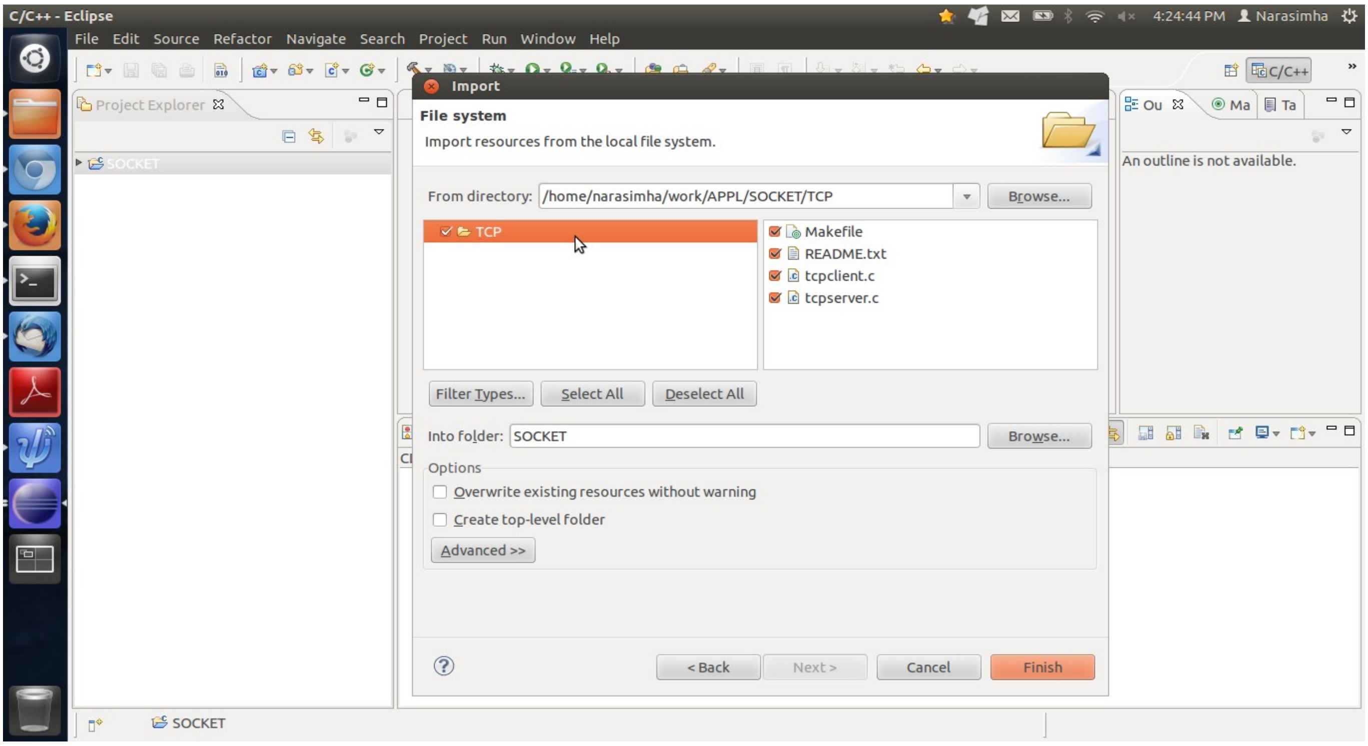
```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/TCP-SOCKET
    $ make clean
    $ make
    $ scp tcpserver tcpclient root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
```

For TCP:

```
    $ ./tcpserver &
    $ ./tcpclient
    $ exit
```

5. **Test Procedure of SOCKET on <BOARD_NAME> using Eclipse IDE:**

1. Select all files as below and click Finish.



File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/SOCKET/TCP

☑ TCP

☑ Makefile
☑ README.txt
☑ tcpclient.c
☑ tcpserver.c

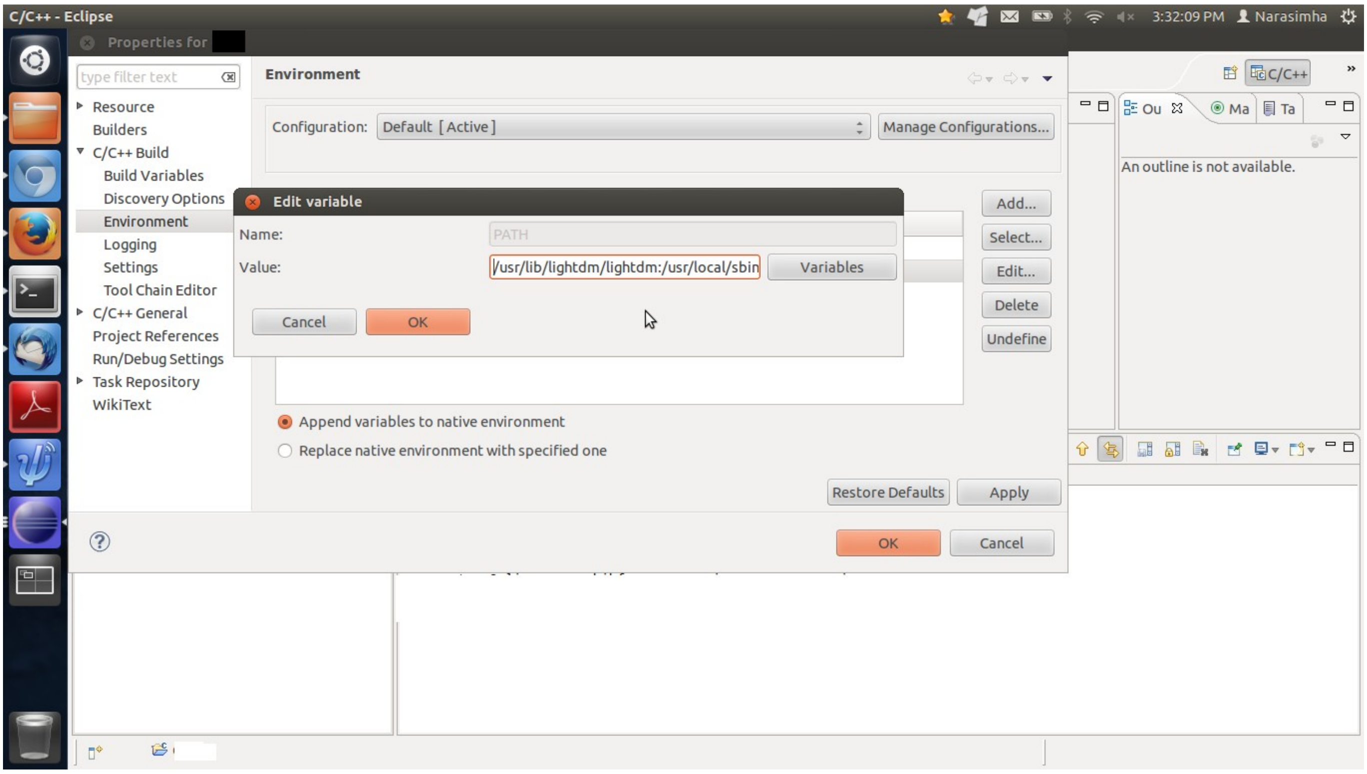
Into folder: SOCKET

Options

□ Overwrite existing resources without warning

□ Create top-level folder

2. Select/Set Tool-chain PATH as shown below:



Properties for [redacted]

Environment

Configuration: Default [Active]

Edit variable

Name: PATH

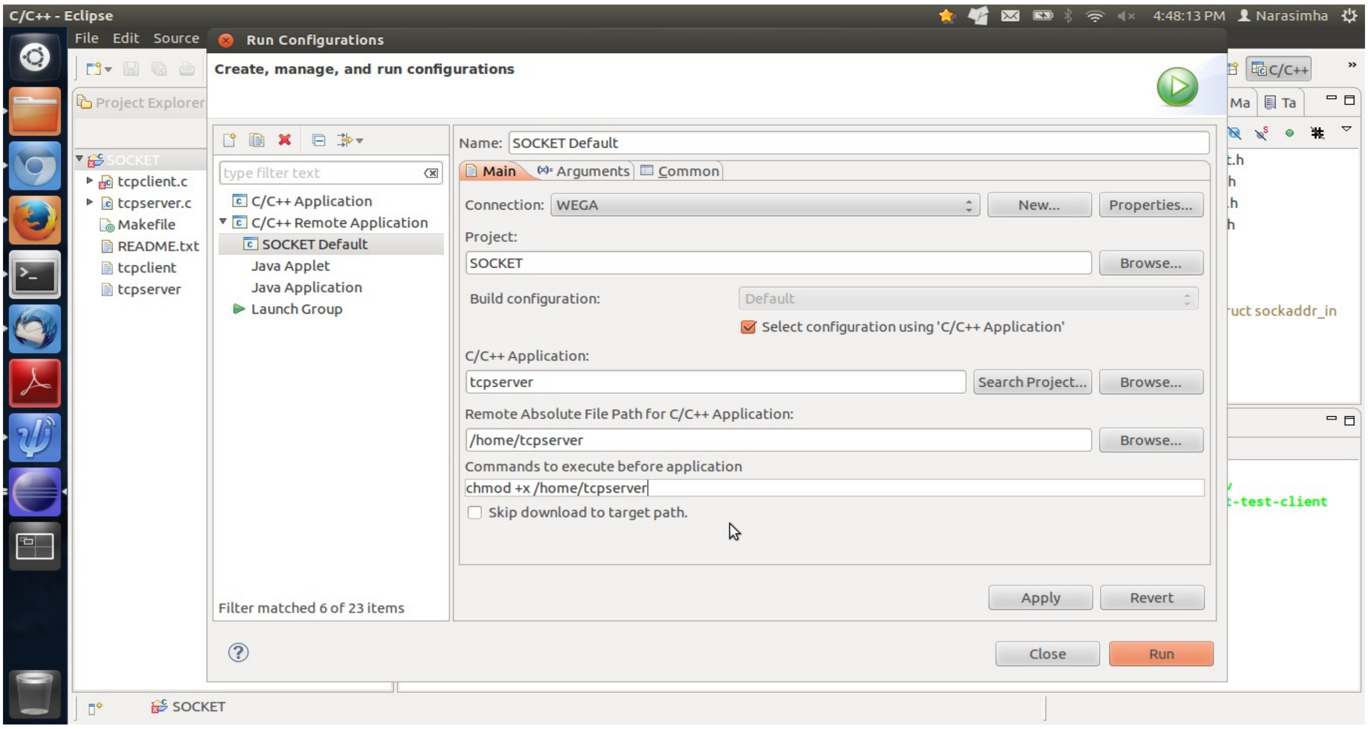
Value: /usr/lib/lightdm/lightdm:/usr/local/sbin

● Append variables to native environment

○ Replace native environment with specified one

3. Right-Click on **SOCKET** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**SOCKET** **&** C/C++ Application:**tcpserver** etc.,**]**



Run Configurations

Create, manage, and run configurations

Name: SOCKET Default

Connection: WEGA

Project: SOCKET

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

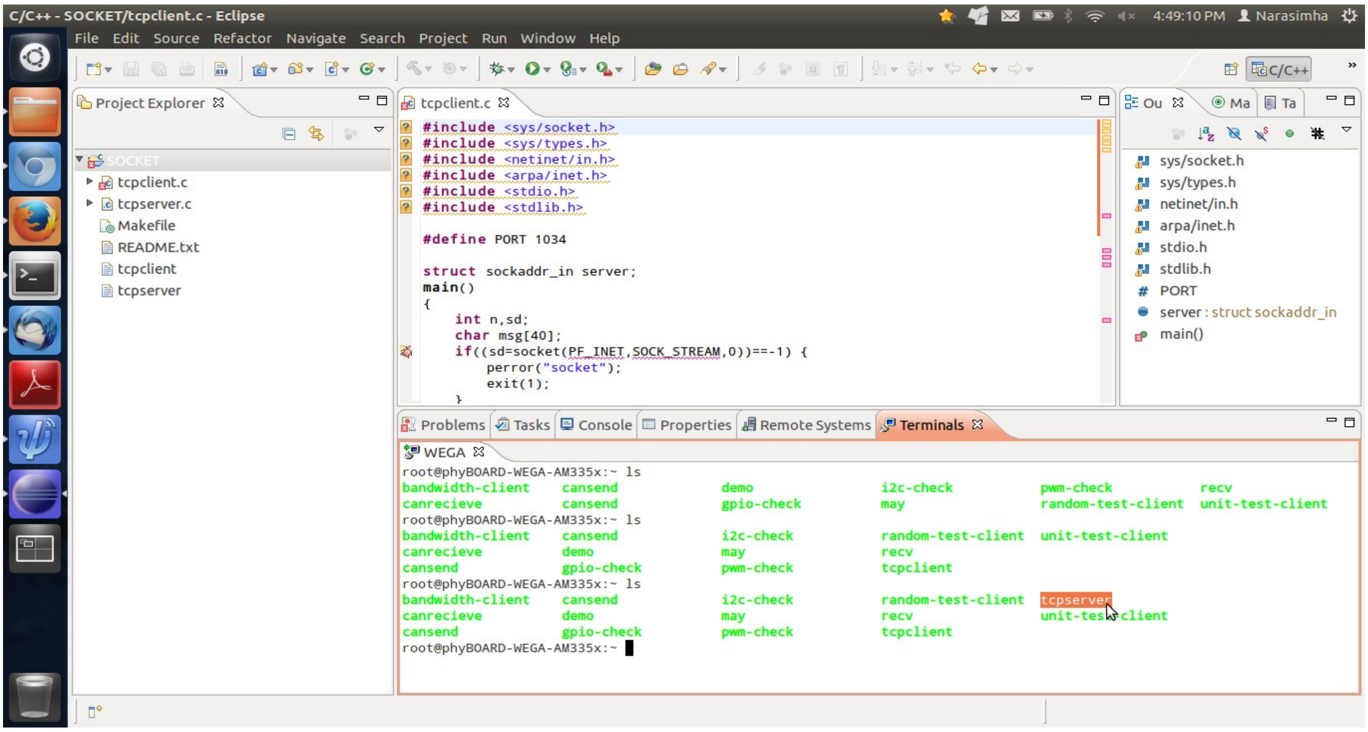
C/C++ Application: tcpserver

Remote Absolute File Path for C/C++ Application: /home/tcpserver

Commands to execute before application: chmod +x /home/tcpserver

☐ Skip download to target path.

4. Below figure gives details about tcpserver-source, tcpserver-binary and the Remote Console-output.

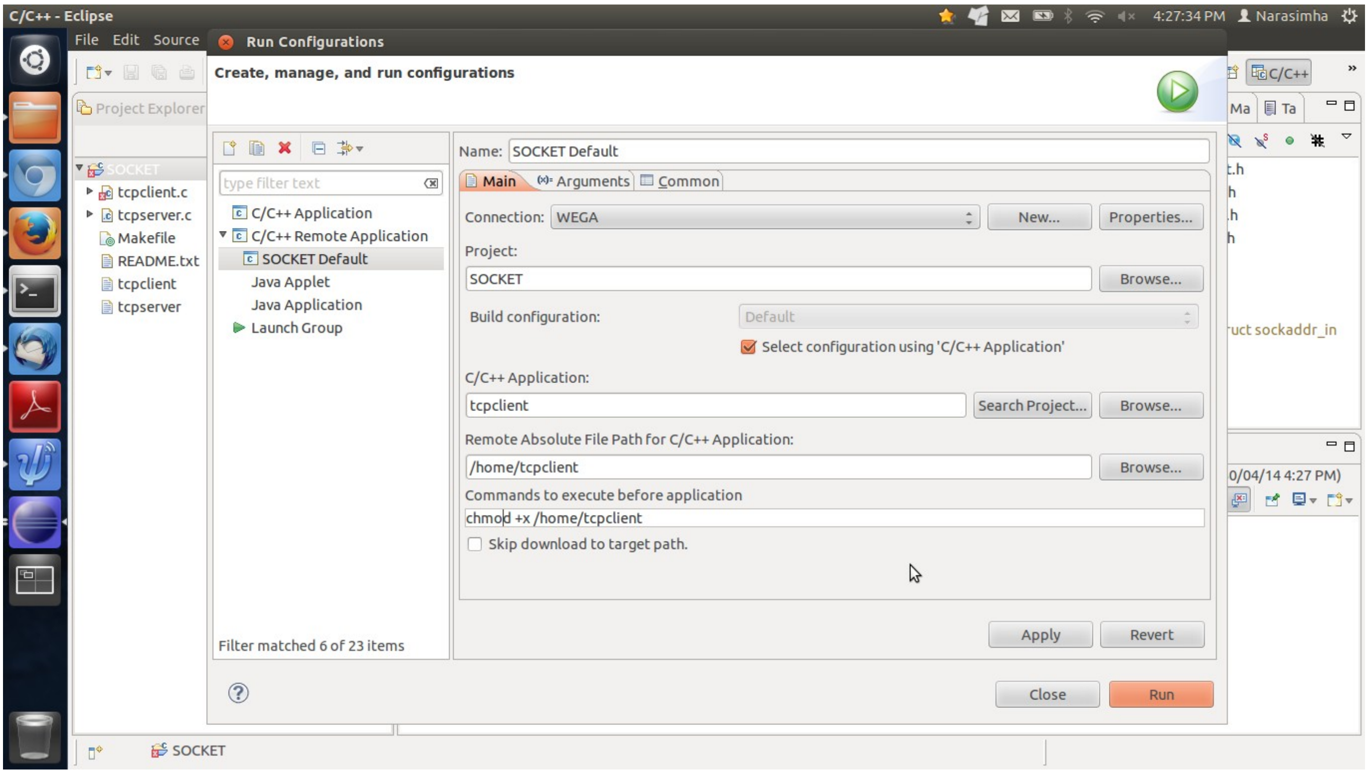


```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

#define PORT 1034

struct sockaddr_in server;
main()
{
    int n,sd;
    char msg[40];
    if((sd=socket(PF_INET,SOCK_STREAM,0))==-1) {
        perror("socket");
        exit(1);
    }
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             demo                i2c-check           pwm-check           recv
canrecieve          cansend             gpio-check          may                 random-test-client  unit-test-client
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  unit-test-client
canrecieve          demo                may                 recv
cansend             gpio-check          pwm-check           tcpclient
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpserver
canrecieve          demo                may                 recv                unit-test-client
cansend             gpio-check          pwm-check           tcpclient
root@phyBOARD-WEGA-AM335x:~
```

5. Right-Click on **SOCKET** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**SOCKET &** C/C++ Application:**tcpclient** etc.,**]**



Create, manage, and run configurations

Name: SOCKET Default

Connection: WEGA

Project:
SOCKET

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

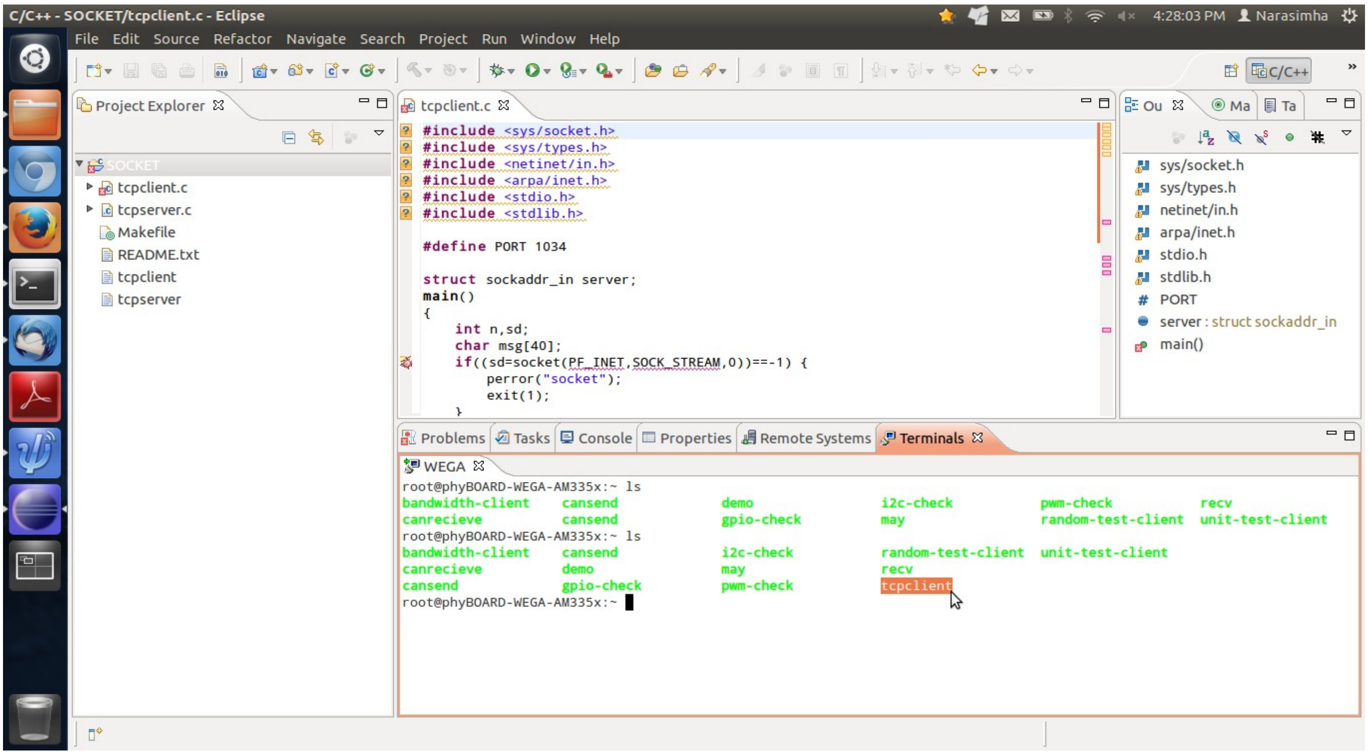
C/C++ Application:
tcpclient

Remote Absolute File Path for C/C++ Application:
/home/tcpclient

Commands to execute before application
chmod +x /home/tcpclient

☐ Skip download to target path.

6. Below figure gives details about tcpclient-source, tcpclient-binary and the Remote Console-output.



```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

#define PORT 1034

struct sockaddr_in server;
main()
{
    int n,sd;
    char msg[40];
    if((sd=socket(PF_INET,SOCK_STREAM,0))==-1) {
        perror("socket");
        exit(1);
    }
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             demo                i2c-check           pwm-check           recv
canrecieve          cansend             gpio-check          may                 random-test-client  unit-test-client
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  unit-test-client
canrecieve          demo                may                 recv
cansend             gpio-check          pwm-check           tcpclient
root@phyBOARD-WEGA-AM335x:~
```

8. UDP-SOCKET APPLICATION

Contents:

1. SOCKET Introduction:

A Socket is an end point of communication between two systems on a network. To be a bit precise, a socket is a combination of IP address and port on one system.

<Board_Name> Linux-Kernel comes with SOCKET Driver for user-selection. For more details of Ethernet(10/100 MB/s) for SOCKET Connections, see the <Board_Name>_Hardware_Manual.pdf

2. SOCKET - Driver Configuration:

For **[Ethernet(10/100) - RMII]** selection, the pin-muxing in kernel board file need to be done. Follow the SOCKET Section of <Board_Name>_System_Development_Guide.pdf

3. UDP-SOCKET access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

UDP-SOCKET library and test program-files:

File-Name	Description
udpserver.c	Server file
udpClient.c	Client file
Makefile	To build the SOCKET Test Programs.

UDP-SOCKET API's for user programming:

[server-side]

Function Name	Description
SOCKET	To create the SOCKET
BIND	Bind a name to a SOCKET
RECVFROM	Receive a message from a Client
SENDTO	Send a message to a Client
CLOSE	Close the UDP-SERVER-SOCKET

[client-side]

Function Name	Description
SOCKET	To create the SOCKET
SENDTO	Send a message to Server
RECVFROM	Receive a message to Server
CLOSE	Close the UDP-CLIENT-SOCKET

Code-Snippet:

```
Server:
 sock_sd = socket(PF_INET, SOCK_DGRAM, 0)
 bind(sd,(struct sockaddr  *)&server,sizeof(server))
 recvfrom(sock_sd, Buff, 100, 0,(struct sockaddr*)&client,&cli_len)
 sendto(sock_sd, Buff, 100, 0, (struct sockaddr *)&client,cli_len);
 close(sock_sd);

Client:
 sock_sd = socket(PF_INET, SOCK_DGRAM, 0)
 sendto(sock_sd,Buff,100,0,(struct sockaddr *)&client,sizeof(server)
 recvfrom(sock_sd, Buff, 100, 0, (struct sockaddr *)&server,&cli_len);
 close(sock_sd)
```

4. Test Procedure of UDP-SOCKET on <BOARD_NAME> using command line:

Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the UDP-SOCKET dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

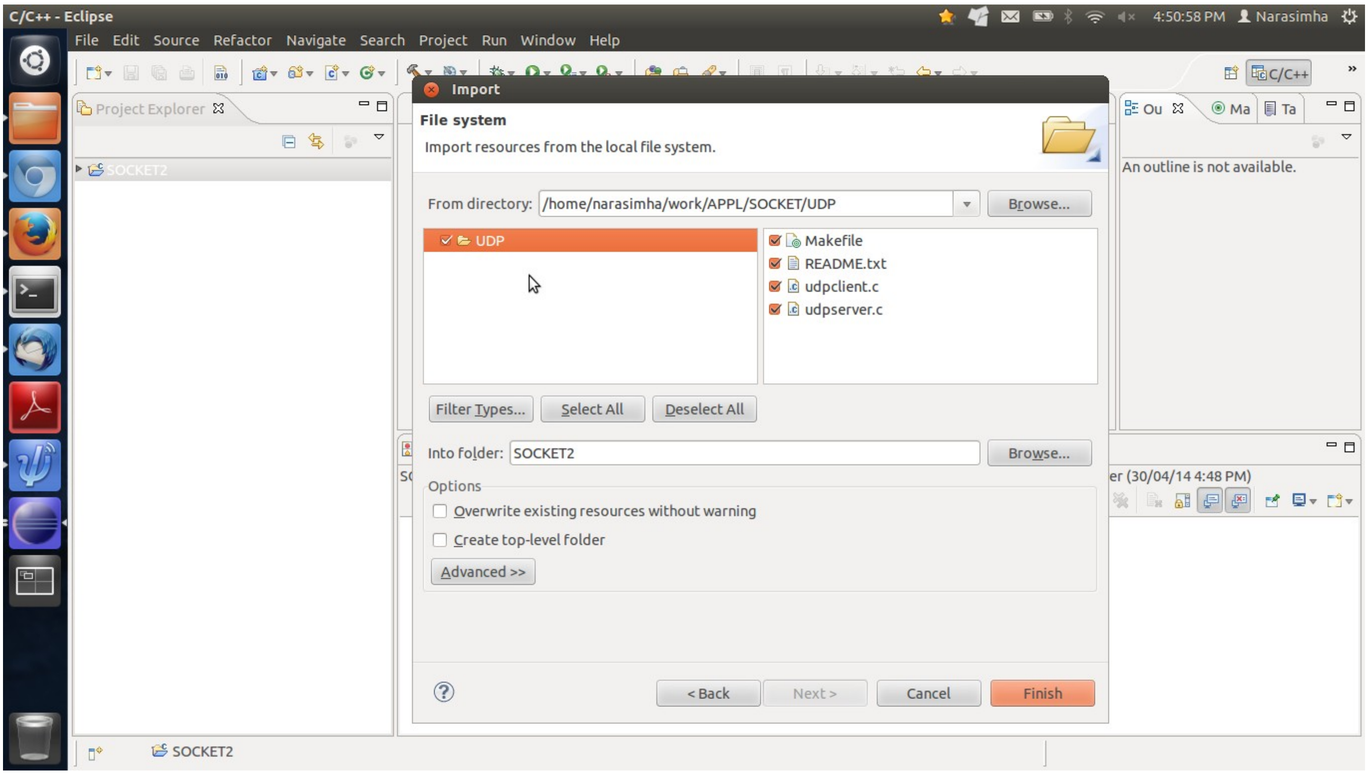
```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/UDP-SOCKET
    $ make clean
    $ make
    $ scp udpserver udpclient root@<BOARD_NAME>:/home/
    $ ssh root@<BOARD IP-address>
```

For UDP:

```
    $ ./udpserver &
    $ ./udpclient
    $ exit
```

5. Test Procedure of SOCKET on <BOARD_NAME> using Eclipse IDE:

1. Select all files as below and click Finish.



File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/SOCKET/UDP

☑ UDP

☑ Makefile
☑ README.txt
☑ udpclient.c
☑ udpserver.c

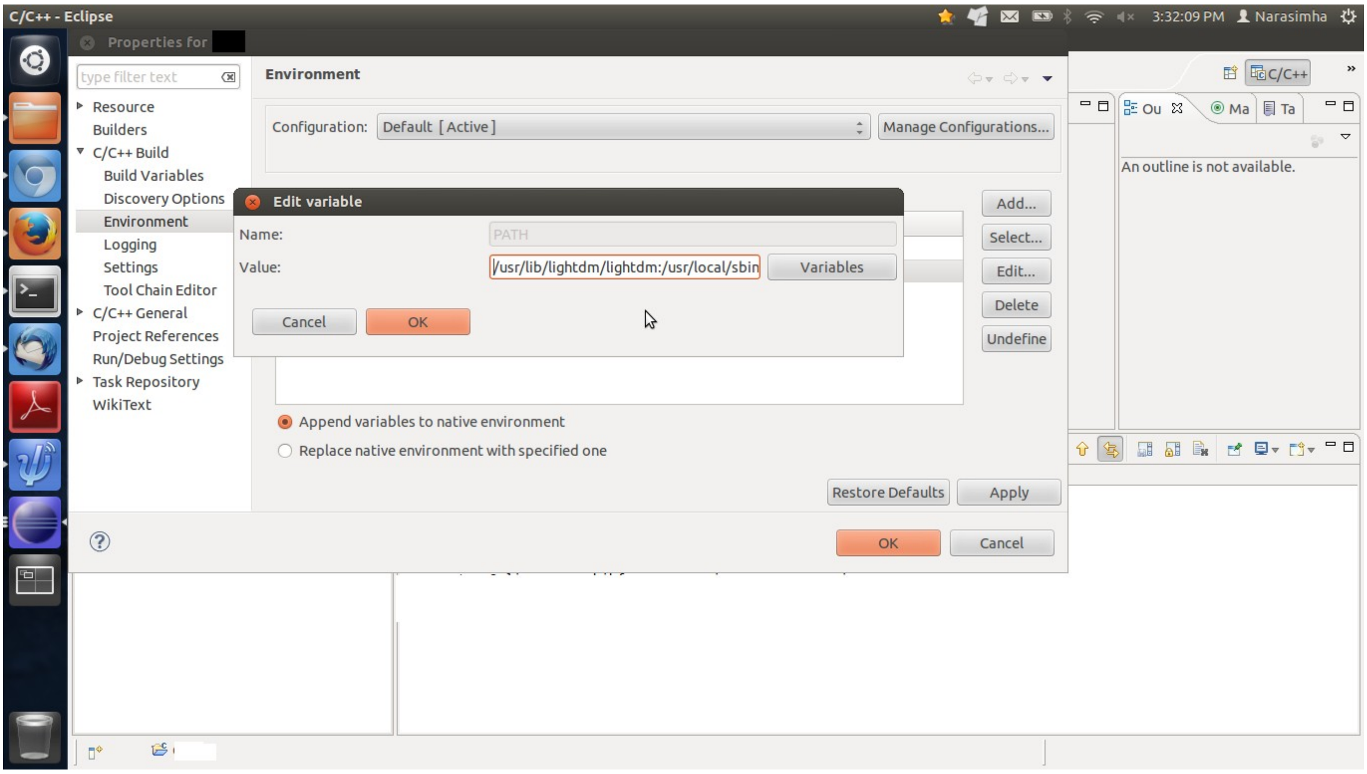
Into folder: SOCKET2

Options

☐ Overwrite existing resources without warning

☐ Create top-level folder

2. Select/Set Tool-chain PATH as shown below:



Environment

Configuration: Default [Active]

Edit variable

Name: PATH

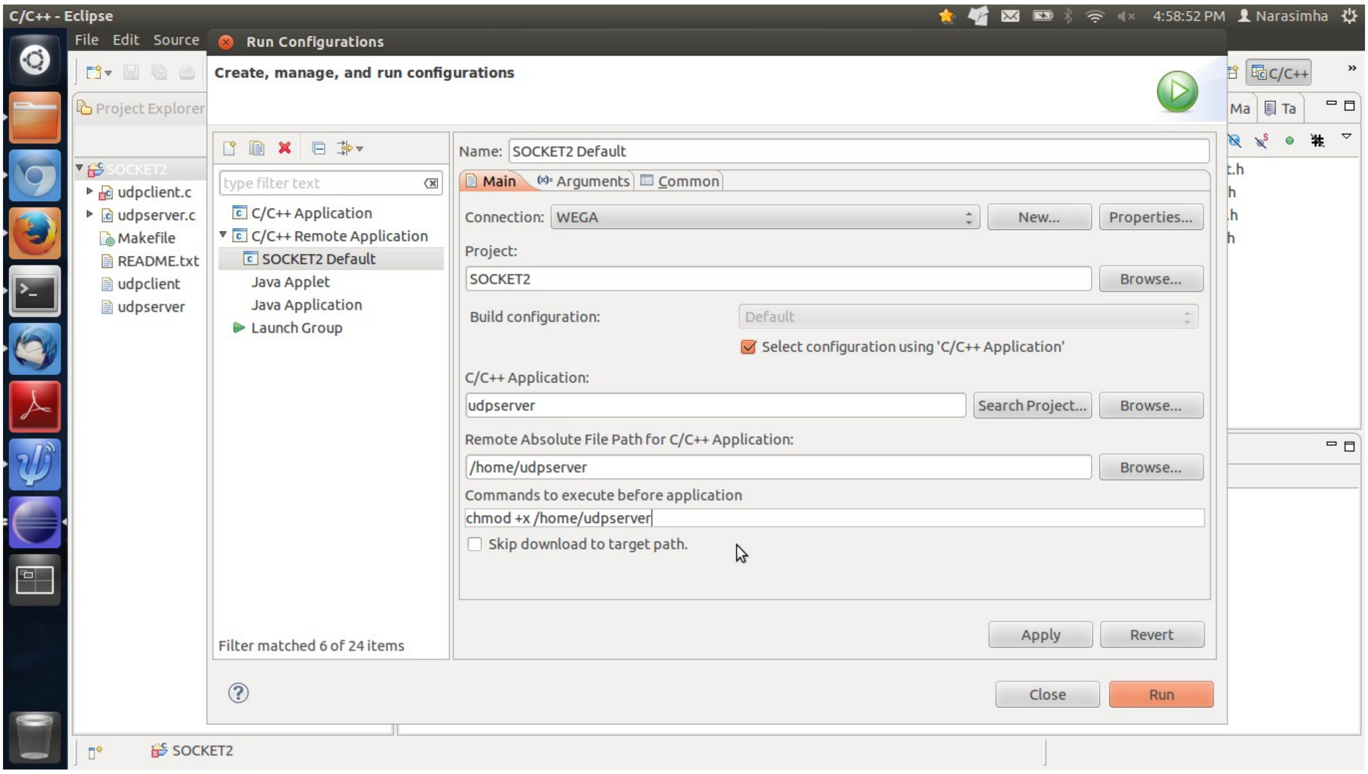
Value: /usr/lib/lightdm/lightdm:/usr/local/sbin

● Append variables to native environment

○ Replace native environment with specified one

3. Right-Click on **SOCKET2** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings as below:

[Note: Connection:**WEGA**, Project:**SOCKET2 &** C/C++ Application:**udpserver** etc.,**]**



Create, manage, and run configurations

Name: SOCKET2 Default

Connection: WEGA

Project: SOCKET2

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: udpserver

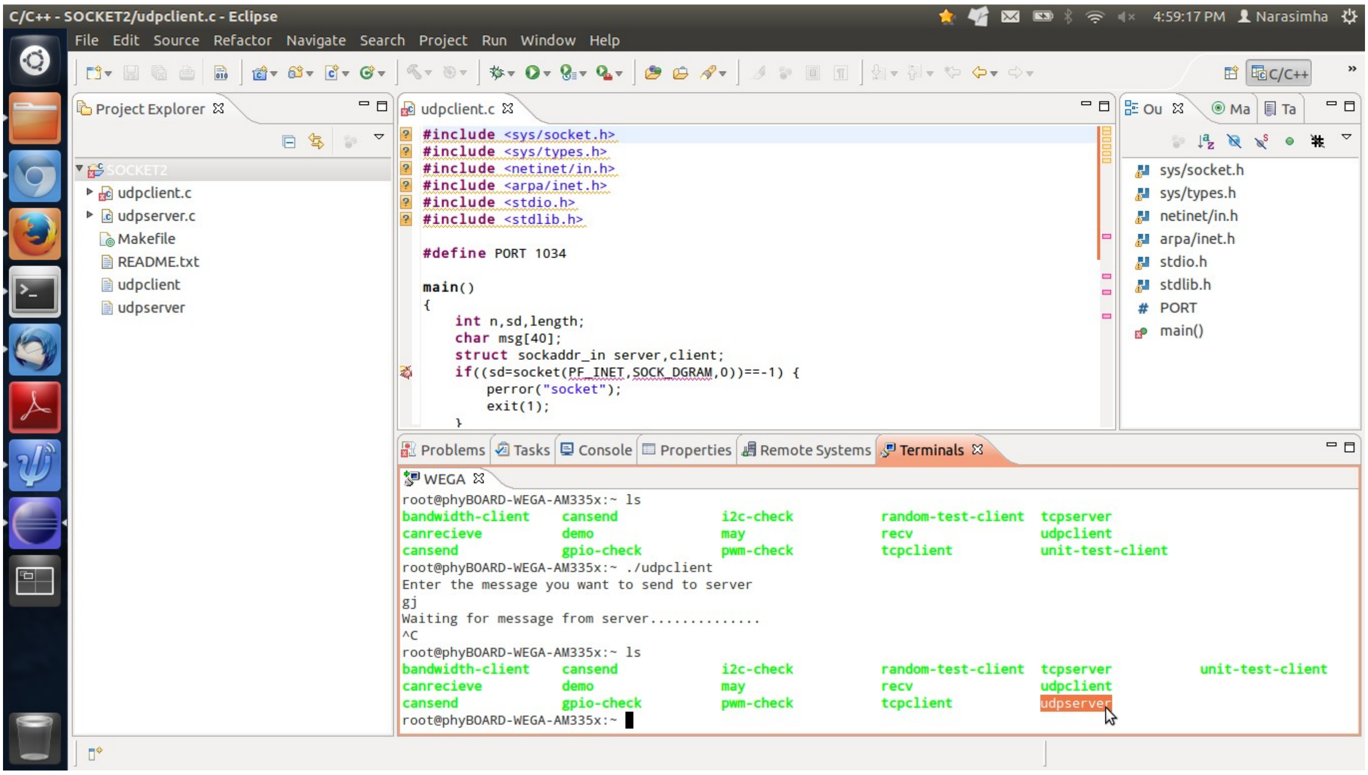
Remote Absolute File Path for C/C++ Application: /home/udpserver

Commands to execute before application: chmod +x /home/udpserver

☐ Skip download to target path.

Filter matched 6 of 24 items

4. Below figure gives details about udpserver-source, udpserver-binary and the Remote Console-output.

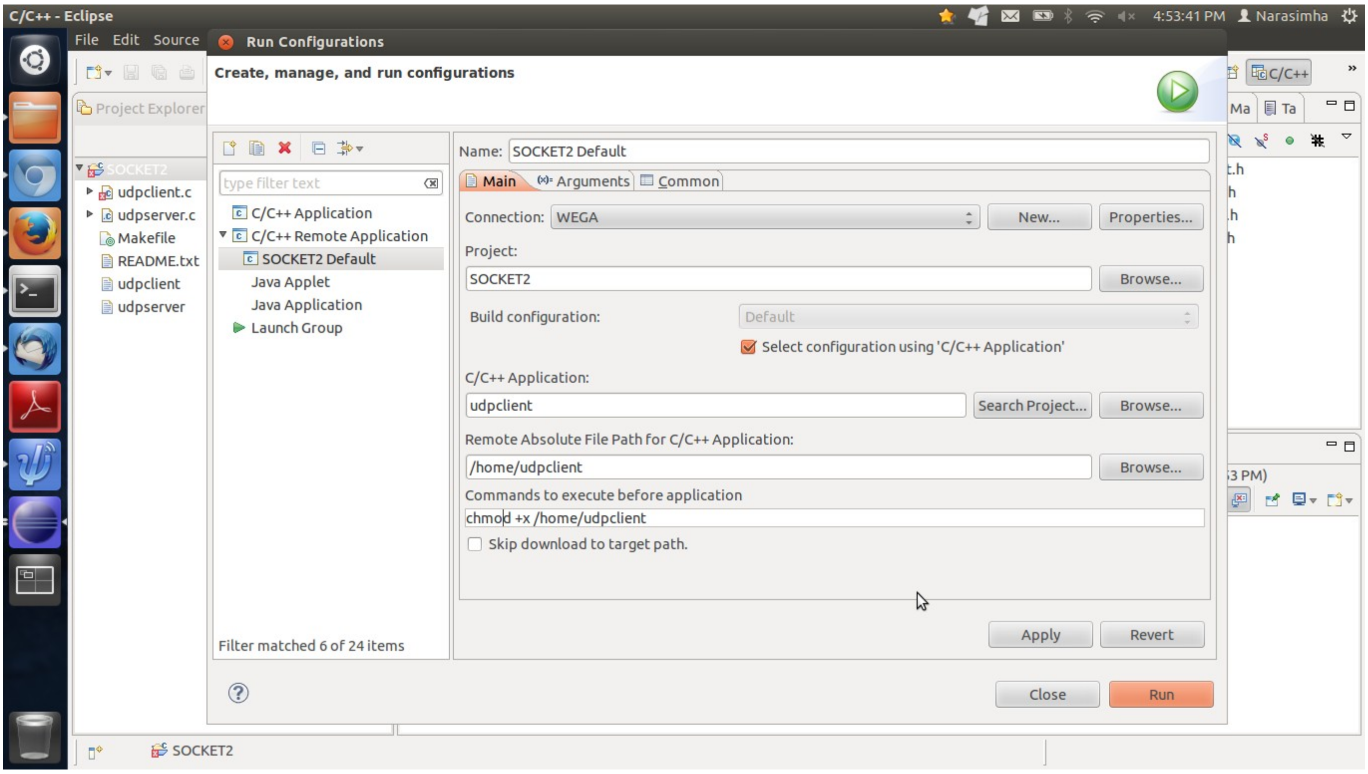


```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

#define PORT 1034

main()
{
    int n,sd,length;
    char msg[40];
    struct sockaddr_in server,client;
    if((sd=socket(PF_INET,SOCK_DGRAM,0))==-1) {
        perror("socket");
        exit(1);
    }
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpserver
canrecieve          demo                may                 recv                udpclient
cansend             gpio-check          pwm-check           tcpclient           unit-test-client
root@phyBOARD-WEGA-AM335x:~ ./udpclient
Enter the message you want to send to server
gj
Waiting for message from server.............
^C
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpserver           unit-test-client
canrecieve          demo                may                 recv                udpclient
cansend             gpio-check          pwm-check           tcpclient           udpserver
root@phyBOARD-WEGA-AM335x:~ 
```

5. Right-Click on **SOCKET2** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings as below:

[Note: Connection:**WEGA**, Project:**SOCKET2 & C/C++** Application:**udpclient** etc.,**]**



Create, manage, and run configurations

Name: SOCKET2 Default

Connection: WEGA

Project: SOCKET2

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

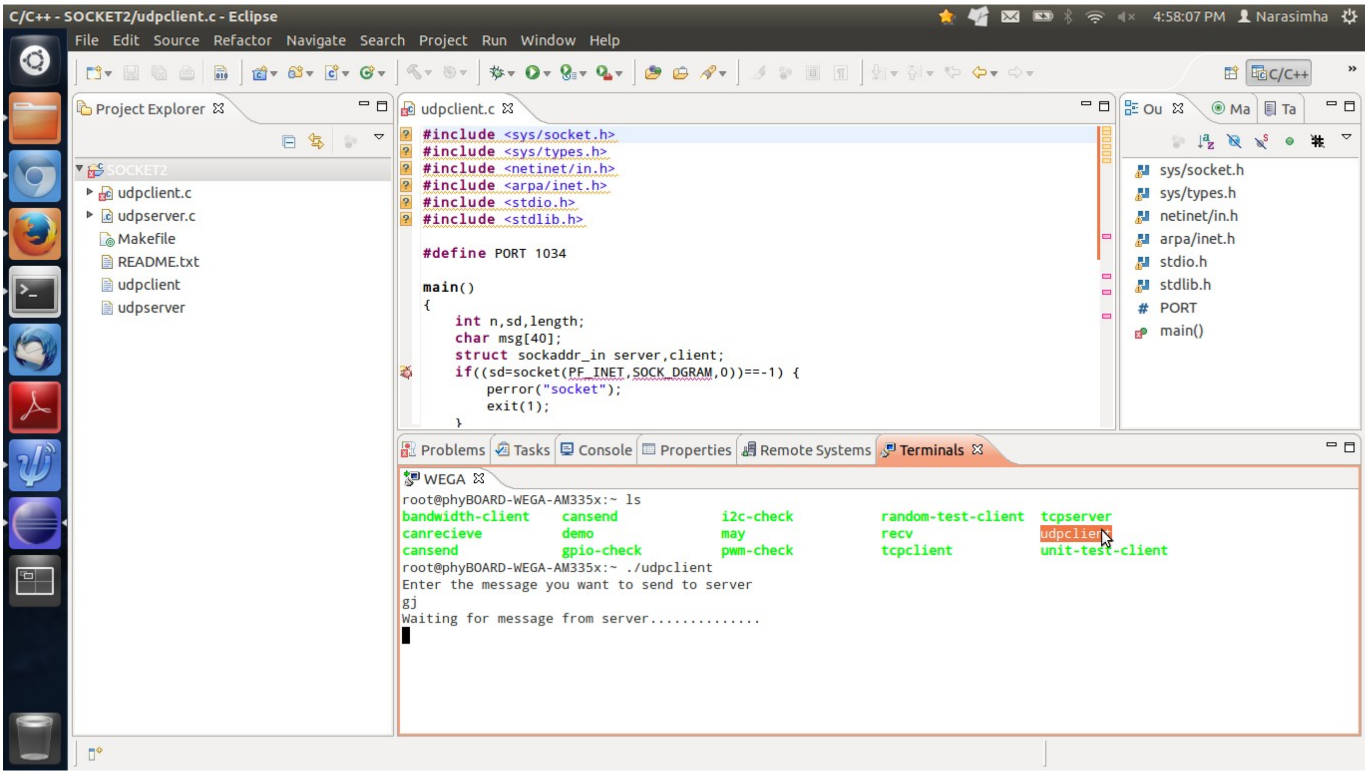
C/C++ Application: udpclient

Remote Absolute File Path for C/C++ Application: /home/udpclient

Commands to execute before application: chmod +x /home/udpclient

□ Skip download to target path.

6. Below figure gives details about udpclient-source, udpclient-binary and the Remote Console-output.



```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

#define PORT 1034

main()
{
    int n,sd,length;
    char msg[40];
    struct sockaddr_in server,client;
    if((sd=socket(PF_INET,SOCK_DGRAM,0))==-1) {
        perror("socket");
        exit(1);
    }
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             i2c-check           random-test-client  tcpserver
canrecieve          demo                may                 recv                udpclient
cansend             gpio-check          pwm-check           tcpclient           unit-test-client
root@phyBOARD-WEGA-AM335x:~ ./udpclient
Enter the message you want to send to server
gj
Waiting for message from server.............
```

9. CAN APPLICATION

Contents:

1. CAN Introduction:

CAN is a networking technology which has widespread use in automation, embedded devices, and automotive fields.

<BOARD_NAME> provides a CAN feature, which is supported by drivers using the proposed Linux standard CAN framework "Socket-CAN".

Using this framework, CAN interfaces can be programmed with the BSD socket API.

For more details of CAN pins on Expansion see the <Board_Name>_Hardware_Manual.pdf

2. CAN - Driver Configuration:

Socketcan interface provides a socket interface to user space applications and which builds upon the Linux network layer.

The pin-muxing for CAN selection in kernel board file need to be done.
Follow the CAN Section of
<Board_Name>_System_Development_Guide.pdf

3. CAN access from user application:

<Board_Name>_Board comes with sample library and test programs and also can be downloaded here.

CAN library and test program-files:

File-Name	Description
cansend.c	Sender file
Canreceive.c	Receiver file
Makefile	To build the CAN test program.

CAN API's for user programming :

Function Name	Description
SOCKET	To open/initialize the CAN
WRITE	To send the message
READ	To receive the message.

Code-Snippet:

```
/* can-send */
  s = socket(PF_CAN, SOCK_RAW, CAN_RAW)
  nbytes = write(s, &frame, sizeof(struct can_frame))



/* can-receive */
  s = socket(PF_CAN, SOCK_RAW, CAN_RAW)
  nbytes = read(s, &frame, sizeof(struct can_frame))
```

4. Test Procedure of CAN on <BOARD_NAME> using command line:

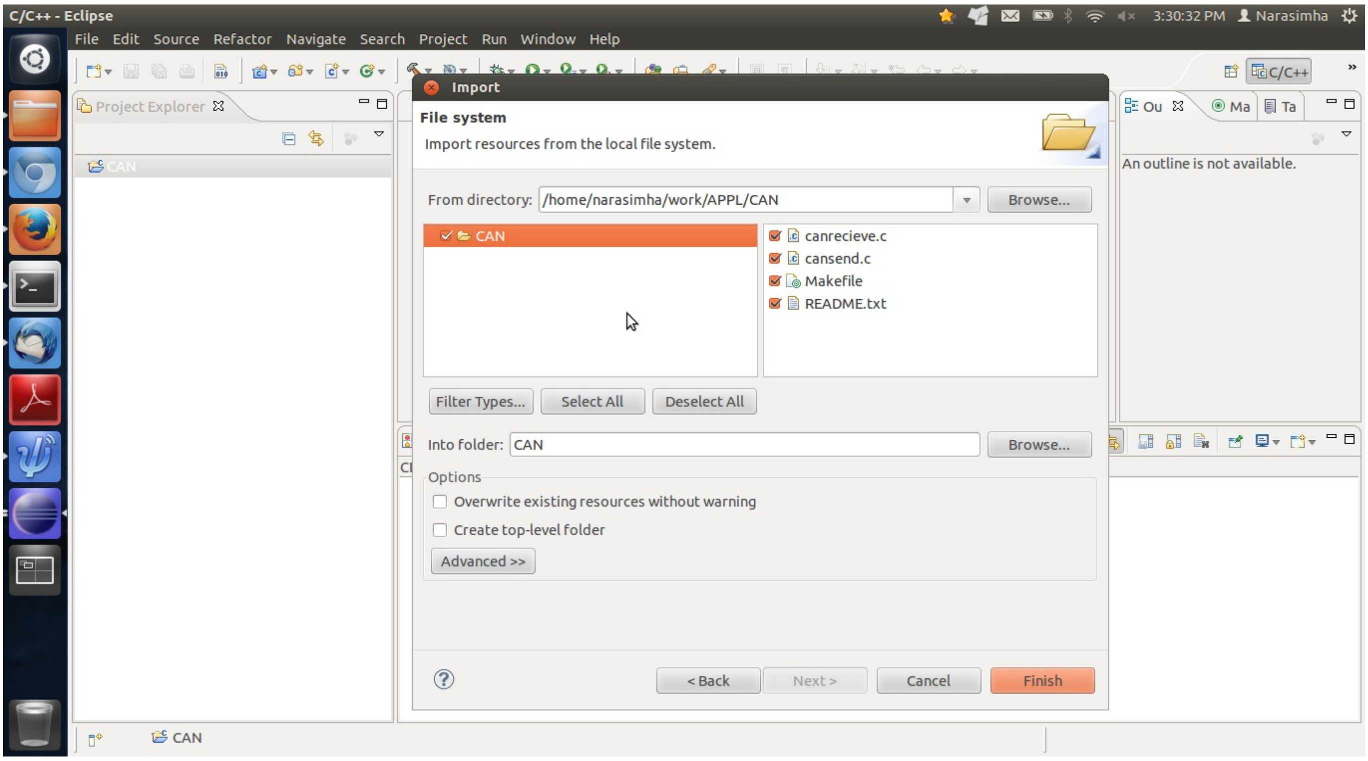
Procedure:

- **a. Set the tool-chain path**
- **b. Switch to the CAN dir and run make command**
- **c. Transfer the bin to the target using scp**
- **d. Open the target shell and execute it.**
- **e. Exit the target shell**

```
    $ export PATH=$PATH:<the path of toolchain bin>
    $ cd <code_base>/app/CAN
    $ make clean
    $ make
    $ scp cansend root@<BOARD_NAME-A>:/home/
    $ scp canreceive root@<BOARD_NAME-B>:/home/

    $ ssh root@<BOARD IP-address>

/* Configure CAN */
    $ canconfig can0 stop
    $ canconfig can0 bitrate 50000 ctrlmode triple-sampling on
                - [configure can to 50k B/s bitrate]
    $ canconfig can0 start

/* On Board-A : Transmitter */
    $ ./cansend
/* On Board-B : Receiver */
    $ ./canreceive

    $ exit
```

5. Test Procedure of CAN on <BOARD_NAME> using Eclipse IDE:

1. Select all files as below and click Finish.


C/C++ - Eclipse
3:30:32 PM
Narasimha
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
CAN
Import
Filter Types... Select All Deselect All
Browse...
Advanced >>
< Back Next > Cancel Finish
C/C++
Ou Ma Ta
An outline is not available.

File system

Import resources from the local file system.

From directory: /home/narasimha/work/APPL/CAN

☑ CAN

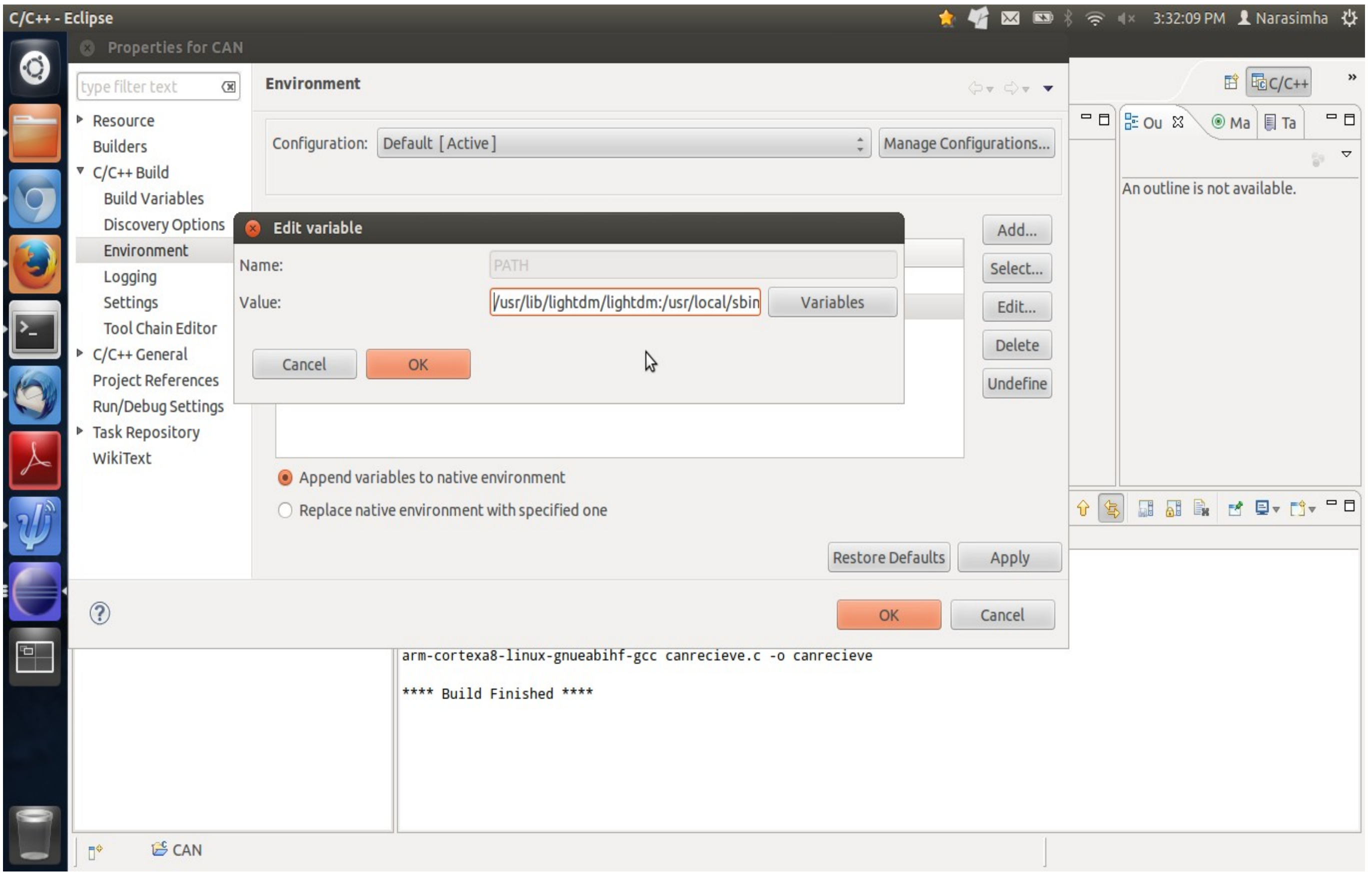
☑ canrecieve.c
☑ cansend.c
☑ Makefile
☑ README.txt

Into folder: CAN

Options

☐ Overwrite existing resources without warning
☐ Create top-level folder

2. Select/Set Tool-chain PATH as shown below:



Environment

Configuration: Default [Active]

Name: PATH

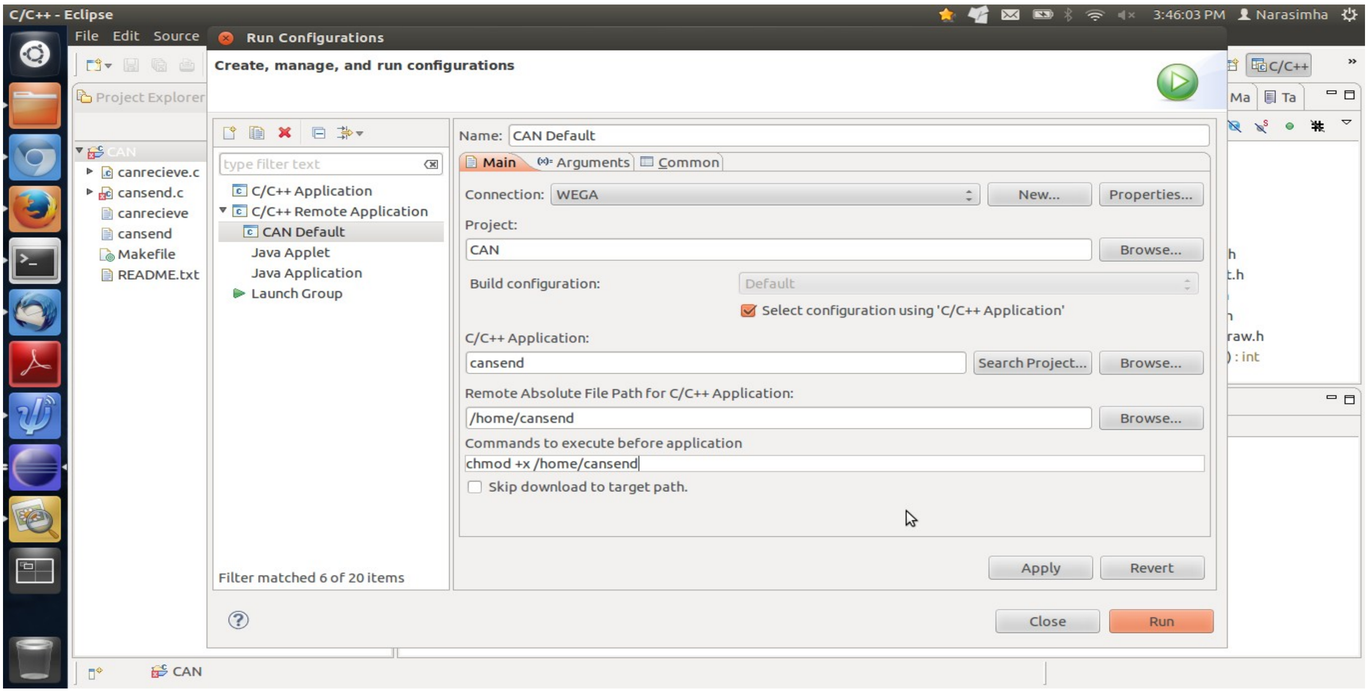
Value: /usr/lib/lightdm/lightdm:/usr/local/sbin

● Append variables to native environment
○ Replace native environment with specified one

```
arm-cortexa8-linux-gnueabihf-gcc canrecieve.c -o canrecieve

**** Build Finished ****
```

3. Right-Click on **CAN** from Project-Explorer & select Run-As(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection:**WEGA**, Project:**CAN &** C/C++ Application:**cansend** etc.,**]**



Name: CAN Default

Connection: WEGA

Project: CAN

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

C/C++ Application: cansend

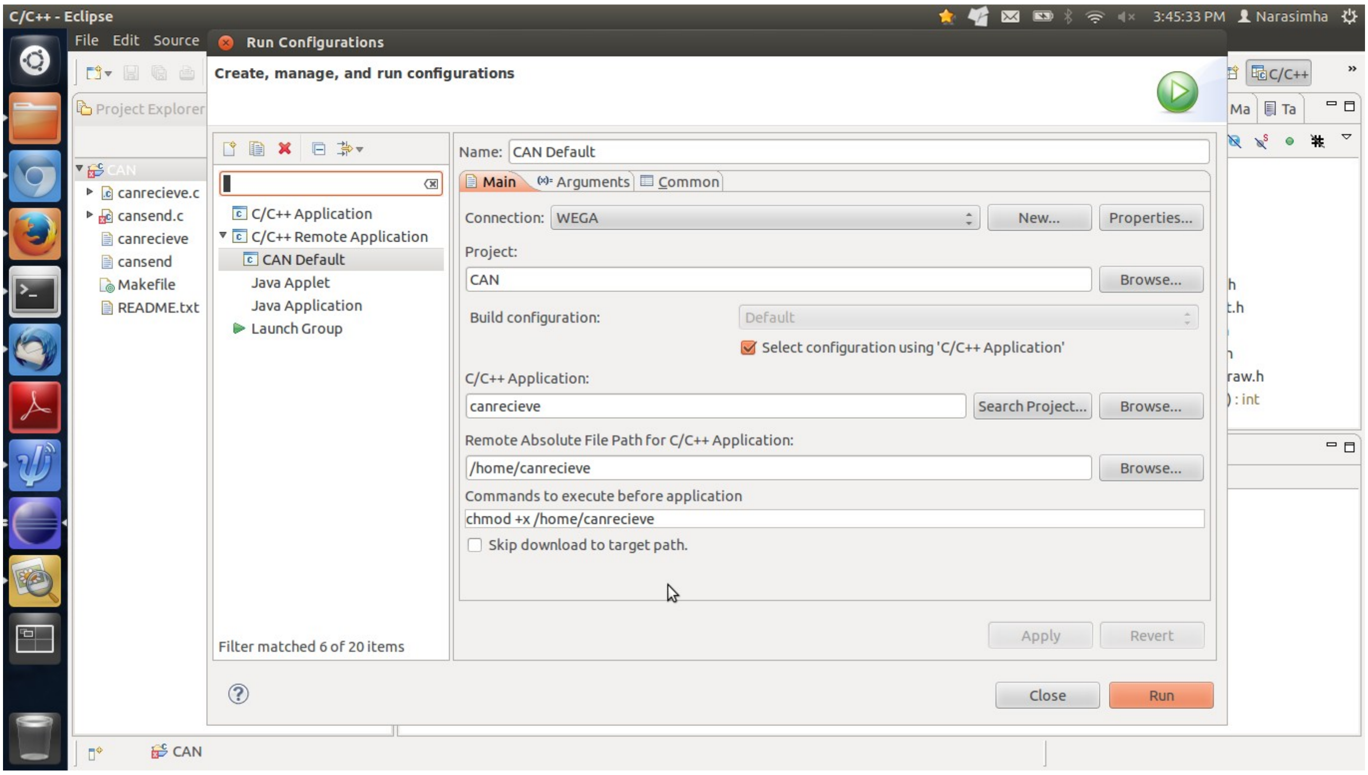
Remote Absolute File Path for C/C++ Application: /home/cansend

Commands to execute before application: chmod +x /home/cansend

□ Skip download to target path.

4. Right-Click on **CAN** from Project-Explorer & select RUN-AS(from Drop-Down-Menu). Then, do settings and below:

[Note: Connection: **WEGA**, Project: **CAN &** C/C++ Application: **canreceive** etc.,**]**



Name: CAN Default

Connection: WEGA

Project: CAN

Build configuration: Default

☑ Select configuration using 'C/C++ Application'

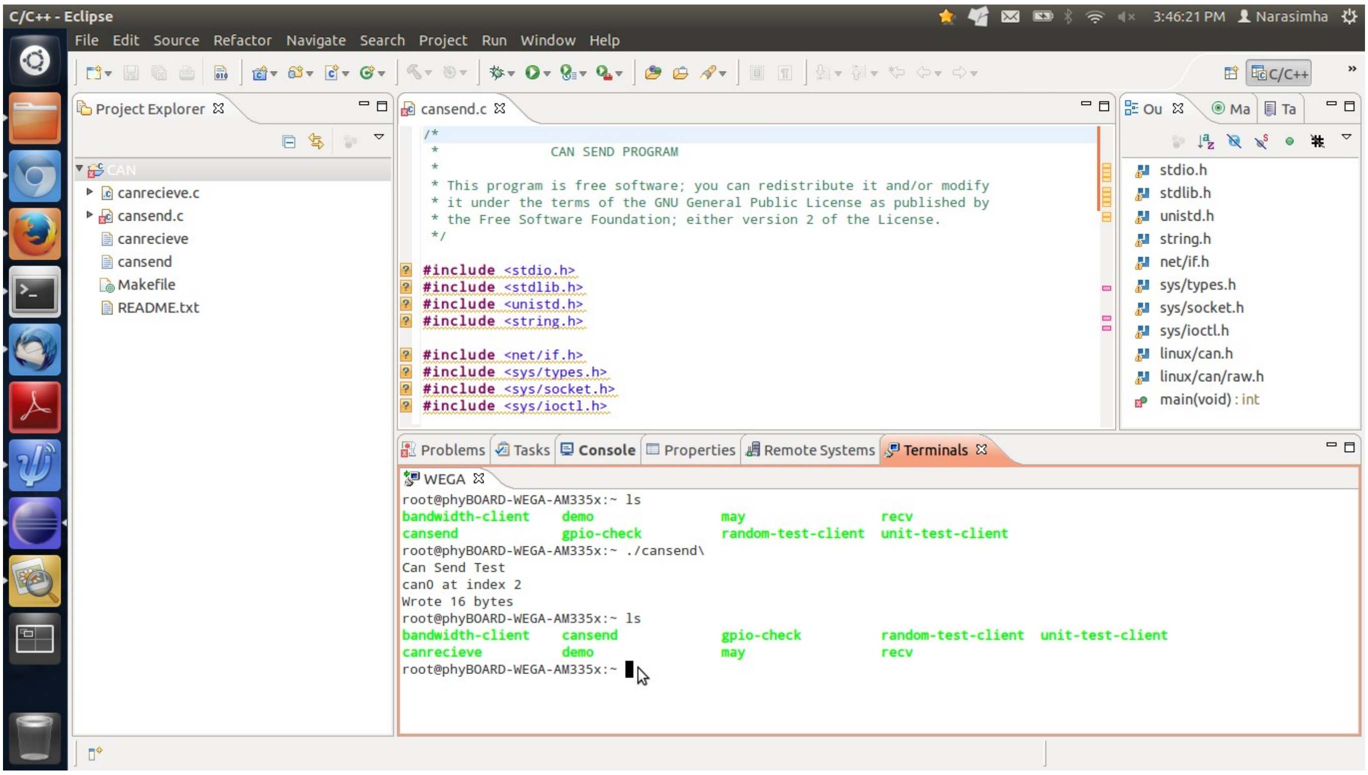
C/C++ Application: canrecieve

Remote Absolute File Path for C/C++ Application: /home/canrecieve

Commands to execute before application: chmod +x /home/canrecieve

□ Skip download to target path.

5. Below figure gives details about can-source, cansend/canreceive – binaries and the Remote Console-output.



```
/*
 *               CAN SEND PROGRAM
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

#include <net/if.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
```

```
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    demo                may                 recv
cansend             gpio-check          random-test-client  unit-test-client
root@phyBOARD-WEGA-AM335x:~ ./cansend\
Can Send Test
can0 at index 2
Wrote 16 bytes
root@phyBOARD-WEGA-AM335x:~ ls
bandwidth-client    cansend             gpio-check          random-test-client  unit-test-client
canrecieve          demo                may                 recv
root@phyBOARD-WEGA-AM335x:~
```

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 hear from you ...!*** ***