Raspberry PiOX Configuration Guide

Table of contents

Configuring the Remote Debugging Environment

Requirements

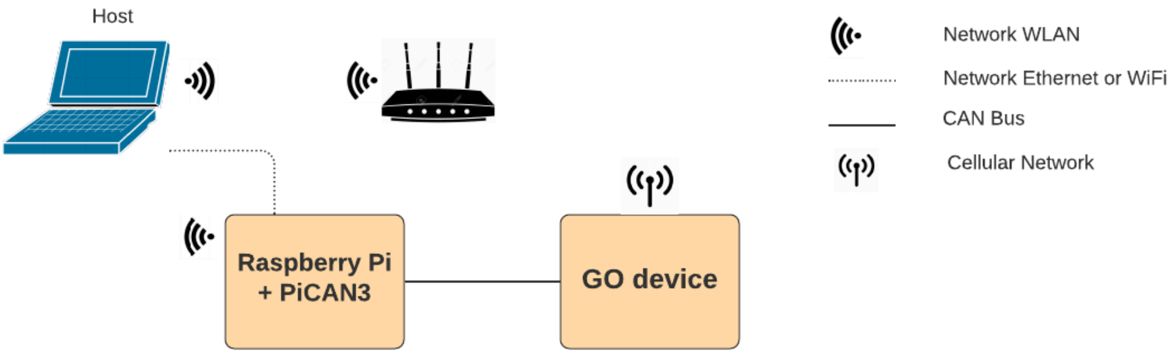
Hardware

1. Raspberry Pi 4 board for deployment and debugging
2. Host machine
3. GO device
4. CAN cable to connect between PiCAN3 and GO device
5. Ethernet cable to connect between RPi and host machine or Wi-Fi connection
6. PiCAN3 CAN Bus Board for Raspberry Pi 4 from: https://copperhilltech.com/pican3-can-bus-board-for-raspberry-pi-4-with-3a-smps-and-rtc/

Software

1. A Linux standalone or virtual machine (VM) desktop
2. Raspberry Pi OS
3. Cross compilation ARM tool chain
4. gdbserver
5. Java Development Kit (JDK)
6. Eclipse IDE
7. Linux can-utils
8. Raspberry PiOX Demo Githab source code

Hardware configuration:



For the **hardware** installation of the PiCAN3 CAN Bus Board on Raspberry Pi 4, refer to the following link. Note that the **software** installation part is covered later.

https://copperhilltech.com/content/PICAN3_UGA_10.pdf

Configuring the Raspberry Pi

General configuration

1. Download and launch the **Raspberry Pi Imager** from the following link: Download Raspberry Pi OS
2. Set up your Raspberry Pi by following the instructions in the following link: Setting up your Raspberry Pi
3. Raspberry Pi documentation (Quickstart, Installation, Configuration, etc.) can be found here: Raspberry Pi Documentation

Setting a static IP address

By default, the IP address of RPi is dynamic and is assigned by the DHCP server. However, when using the RPi as a remote target, and when access to the device will be through remote tools such as SSH (Secure Shell), it's useful and practical to assign the RPi a static private IP address.

In the RPi environment, open the command prompt and type the following command to open the dhcpcd.conf file:

```
sudo nano /etc/dhcpcd.conf
```

Scroll down to the end of the dhcpcd.conf file and manually add the static IP address (e.g. 192.168.2.66/24), the network gateway, and the domain name server with the following format:

Example configuration

```
interface eth0

static ip_address=192.168.2.66/24

static routers=192.168.2.1

static domain_name_servers=192.168.2.1
```

✱ NOTE: If your RPi is connected to the internet via an Ethernet or network cable, then enter the command `interface eth0`. If it is connected to the internet via Wi-Fi, then enter the command `interface wlan0`.

To save and exit the dhcpcd.conf file, press Ctrl+X, press Y, and then press Enter. Finally, reboot the device.

For more information, see the following link:

https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-give-your-raspberry-pi-a-static-ip-address-update

Setting Up a Cross-Compilation Toolchain (Linux)

As a prerequisite, you will need a Linux OS standalone or virtual machine (VM) desktop instance.

✱ NOTE: The following section is implemented on Virtual Box with Ubuntu 20.04.

Updating the Linux OS

✱ **NOTE:** If Linux OS was just installed, you can skip this step as the OS should already be the latest version.

Open terminal command prompt and type the following commands:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Updating the sources lists

You need to update the sources lists to include the cross-toolchain sources list, which makes a list of available cross-compilation packages.

Navigate to the sources.list.d directory by typing the following command:

```
cd /etc/apt/sources.list.d/
```

Create a new file crosstools.list in this directory by typing the following command:

```
sudo nano crosstools.list
```

Manually add the following to the crosstools.list file:

```
deb http://emdebian.org/tools/debian jessie main
```

Save and close the file. Press Ctrl+X, press Y, then press Enter.

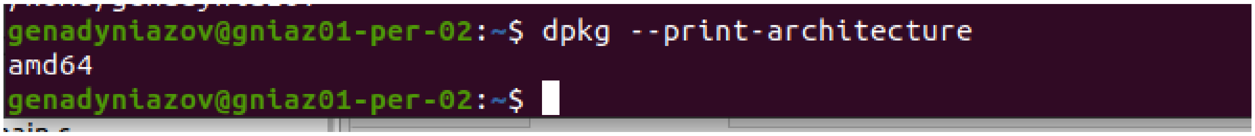
Adding armhf as a foreign architecture and updating the list of available packages

In the terminal, type the following command:

```
sudo dpkg --add-architecture armhf
```

Check the native architecture of the system by typing the following command in the terminal:

```
dpkg --print-architecture
```

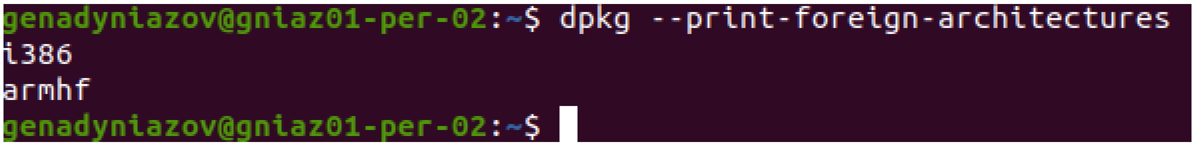


```
genadyniazov@gniaz01-per-02:~$ dpkg --print-architecture
amd64
genadyniazov@gniaz01-per-02:~$
```

Check the foreign architecture in the system by typing the following command in the terminal:

```
dpkg --print-foreign-architectures
```

Ensure that the output includes `armhf`.



```
genadyniazov@gniaz01-per-02:~$ dpkg --print-foreign-architectures
i386
armhf
genadyniazov@gniaz01-per-02:~$
```

Then type the following command:

```
sudo apt update
```

Installing the cross-build toolchain

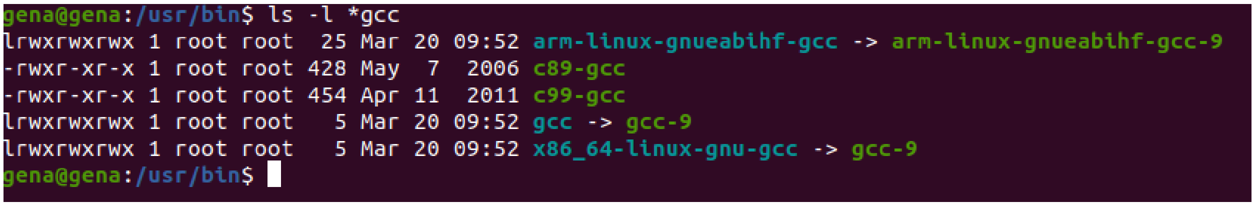
In the terminal, type the following command:

```
sudo apt install crossbuild-essential-armhf
```

Navigate to the bin folder and check the entry of gcc\g++ for natively compiling x86 code and arm-linux-gnueabihi-gcc\g++ entry for cross compiling armhf code:

```
cd /usr/bin
```

```
gena@desktop:/usr/bin$ ls -l *g++
```



```
gena@gena:/usr/bin$ ls -l *gcc
lrwxrwxrwx 1 root root  25 Mar 20 09:52 arm-linux-gnueabihf-gcc -> arm-linux-gnueabihf-gcc-9
-rwxr-xr-x 1 root root 428 May  7  2006 c89-gcc
-rwxr-xr-x 1 root root 454 Apr 11  2011 c99-gcc
lrwxrwxrwx 1 root root   5 Mar 20 09:52 gcc -> gcc-9
lrwxrwxrwx 1 root root   5 Mar 20 09:52 x86_64-linux-gnu-gcc -> gcc-9
gena@gena:/usr/bin$
```

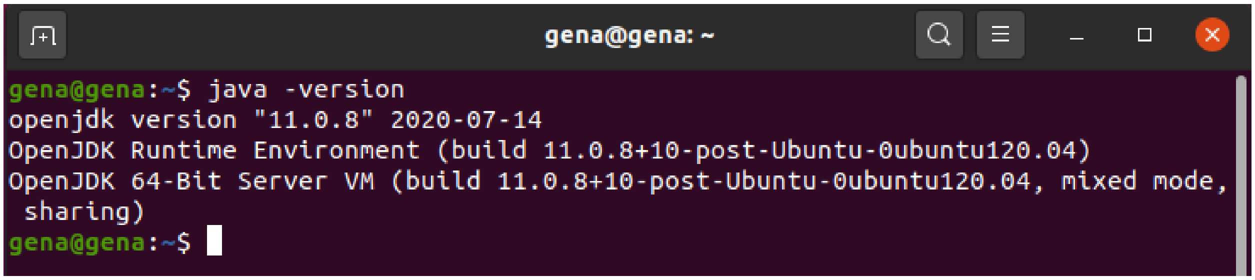
Installing and Configuring Eclipse for Remote Debugging

Installing the Java Development Kit (JDK)

To use Eclipse for C\C++ programming, you need to install the Java Development Kit (JDK) on your Linux machine.

1. First, update the apt package index with the following command:
   ```
   sudo apt update
   ```
2. Once the package index is updated install the default Java OpenJDK package with:
   ```
   sudo apt install default-jre
   ```
3. Verify the installation, by running the following command which will print the Java version:
   ```
   java -version
   ```

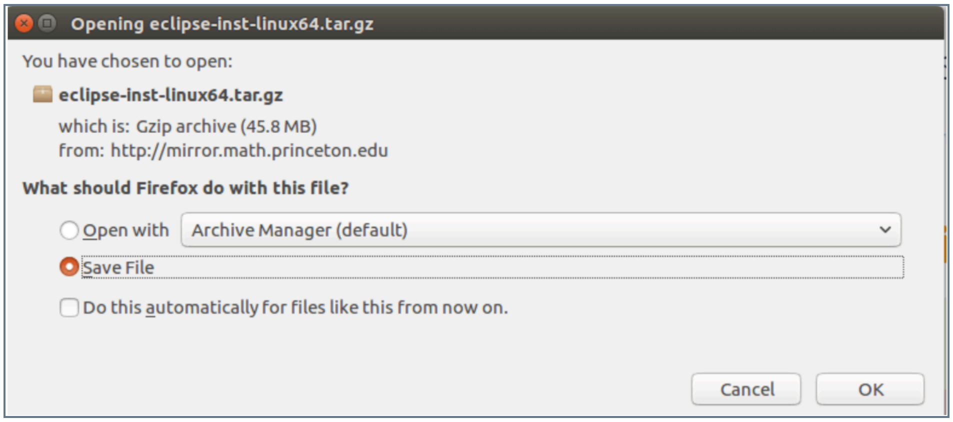
The output will look something like this:



```
gena@gena:~$ java -version
openjdk version "11.0.8" 2020-07-14
OpenJDK Runtime Environment (build 11.0.8+10-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.8+10-post-Ubuntu-0ubuntu120.04, mixed mode,
 sharing)
gena@gena:~$
```

Downloading Eclipse

Eclipse can be downloaded from the following link: https://www.eclipse.org/downloads/



Opening eclipse-inst-linux64.tar.gz

You have chosen to open:

eclipse-inst-linux64.tar.gz

which is: Gzip archive (45.8 MB)
from: http://mirror.math.princeton.edu

What should Firefox do with this file?

○ Open with Archive Manager (default)

● Save File

□ Do this automatically for files like this from now on.

Extract the downloaded package, by default Eclipse package should be downloaded into the ~/Downloads folder of your home directory.

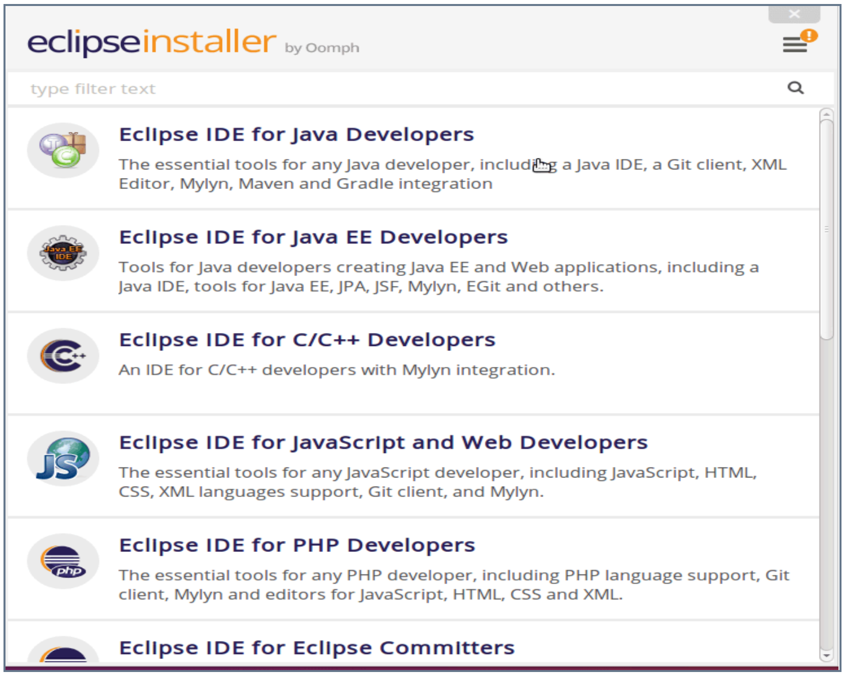
Installing Eclipse IDE

Use the commands below to extract the content in the ~/Downloads folder and then launch the installer:

```
tar xfz ~/Downloads/eclipse-inst-linux64.tar.gz
```

```
cd Downloads/eclipse-installer/eclipse-inst
```

Select the package **Eclipse IDE for C/C++ Developers** to install and continue:



Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Eclipse IDE for Java EE Developers
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.

Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.

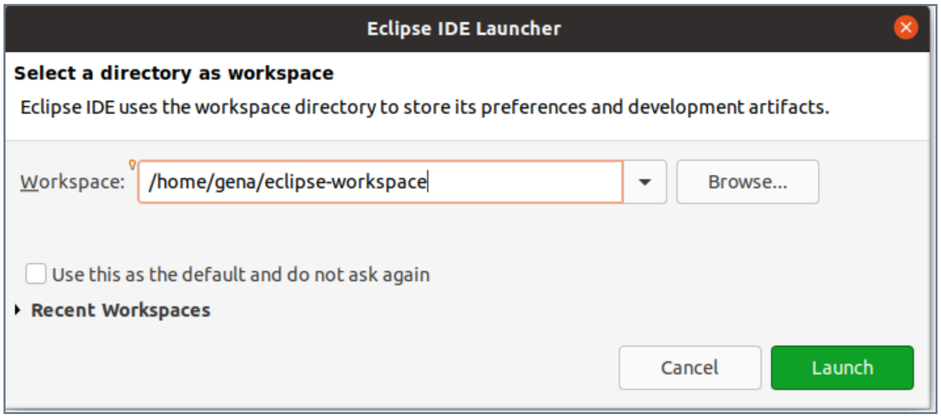
Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Eclipse IDE for Eclipse Committers

Use the onscreen instructions to complete the installer. Accept the default installation directory and continue. Next, accept the license terms and continue, wait for Eclipse installer to download and install all the packages.

Configuring Eclipse for cross-compilation

Start Eclipse IDE, specify the default Workspace directory, and then click **Launch**.



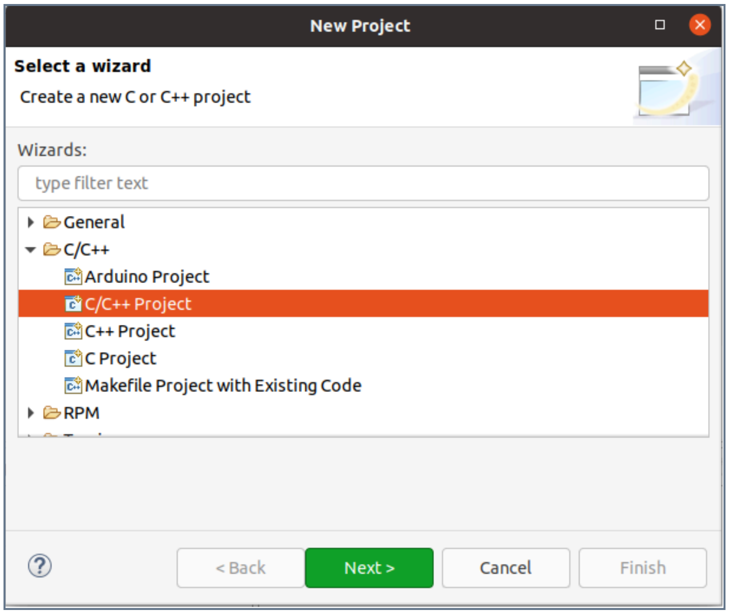
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/gena/eclipse-workspace

□ Use this as the default and do not ask again

Recent Workspaces

Create a new project by navigating to **File** > **New** > **Project** > **C/C++** > **C/C++ Project** and then click **Next**.



New Project

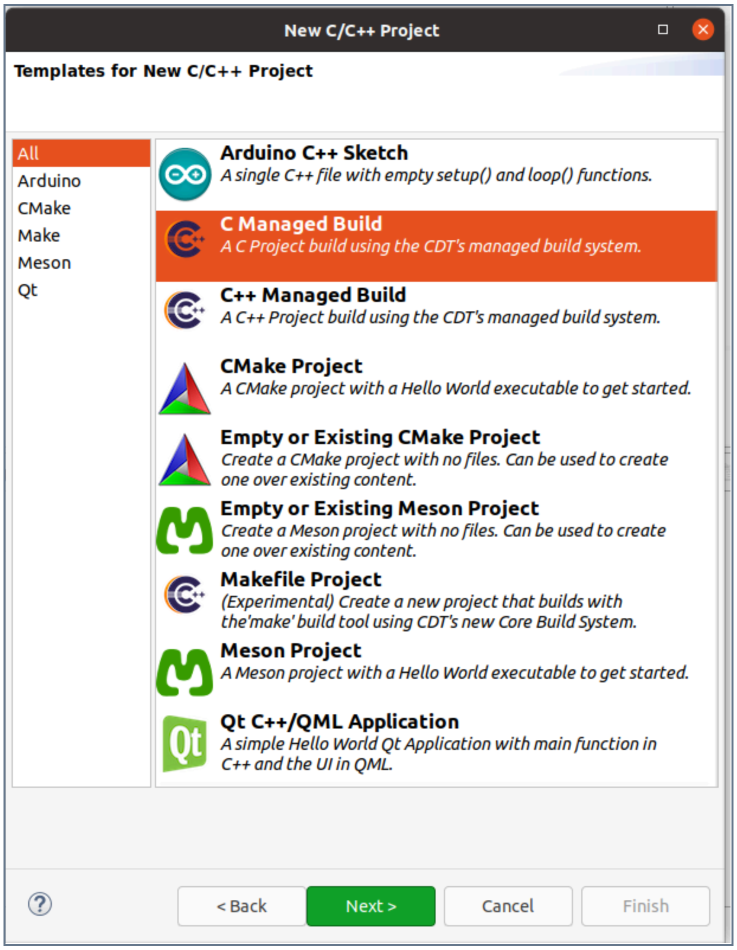
Select a wizard

Create a new C or C++ project

Wizards:

type filter text

Select **C Managed Build** as the template and then click **Next**.



New C/C++ Project

Templates for New C/C++ Project

Arduino C++ Sketch
A single C++ file with empty setup() and loop() functions.

C Managed Build
A C Project build using the CDT's managed build system.

C++ Managed Build
A C++ Project build using the CDT's managed build system.

CMake Project
A CMake project with a Hello World executable to get started.

Empty or Existing CMake Project
Create a CMake project with no files. Can be used to create one over existing content.

Empty or Existing Meson Project
Create a Meson project with no files. Can be used to create one over existing content.

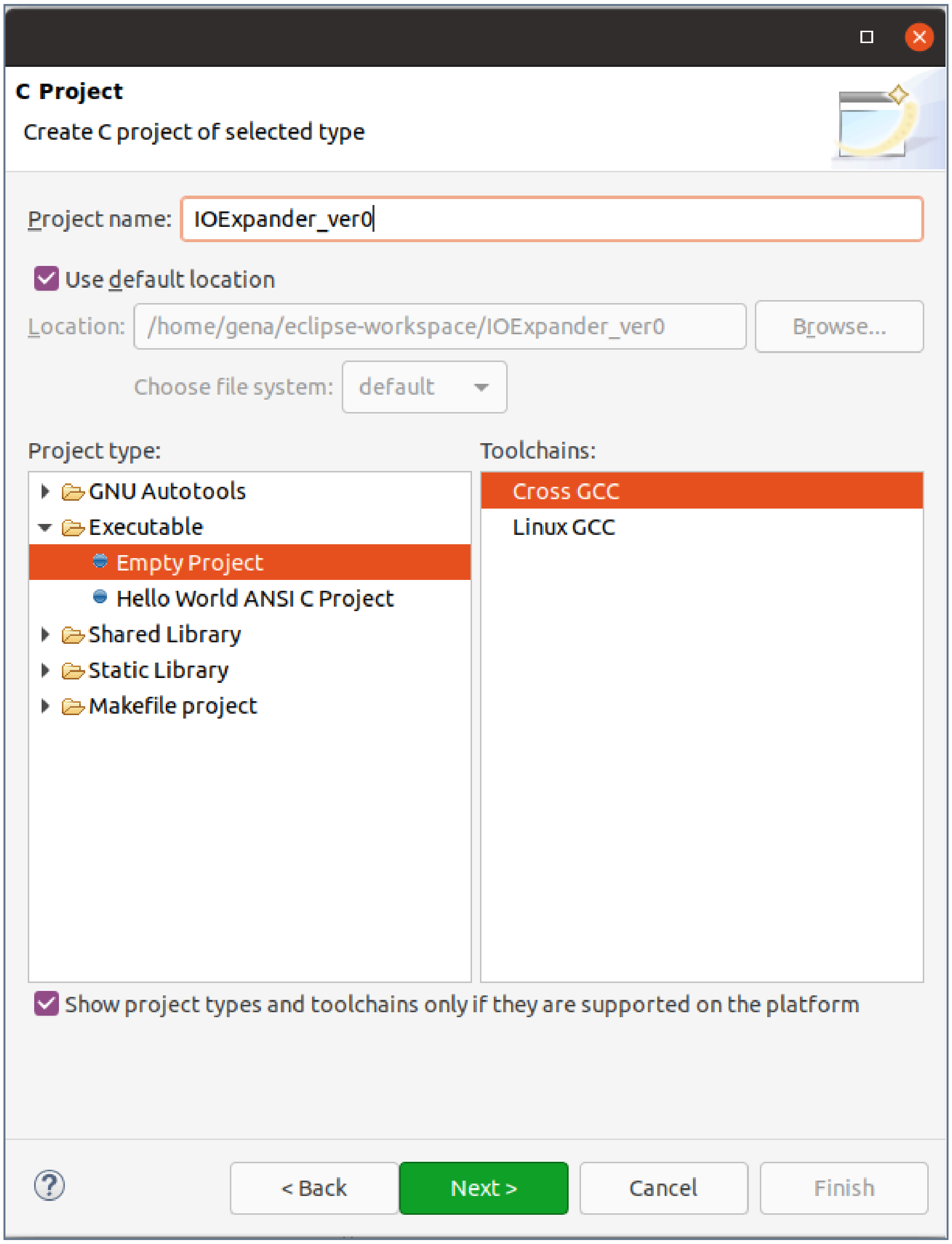
Makefile Project
(Experimental) Create a new project that builds with the'make' build tool using CDT's new Core Build System.

Meson Project
A Meson project with a Hello World executable to get started.

Qt C++/QML Application
A simple Hello World Qt Application with main function in C++ and the UI in QML.

Enter a project name, and then:

1. Under **Project type**, in the **Executable** folder, select **Empty Project**.
2. Under **Toolchains**, select **Cross GCC**.
3. Click **Next**.



C Project

Create C project of selected type

Project name: IOExpander_ver0

☑ Use default location

Location: /home/gena/eclipse-workspace/IOExpander_ver0

Choose file system: default

Project type:

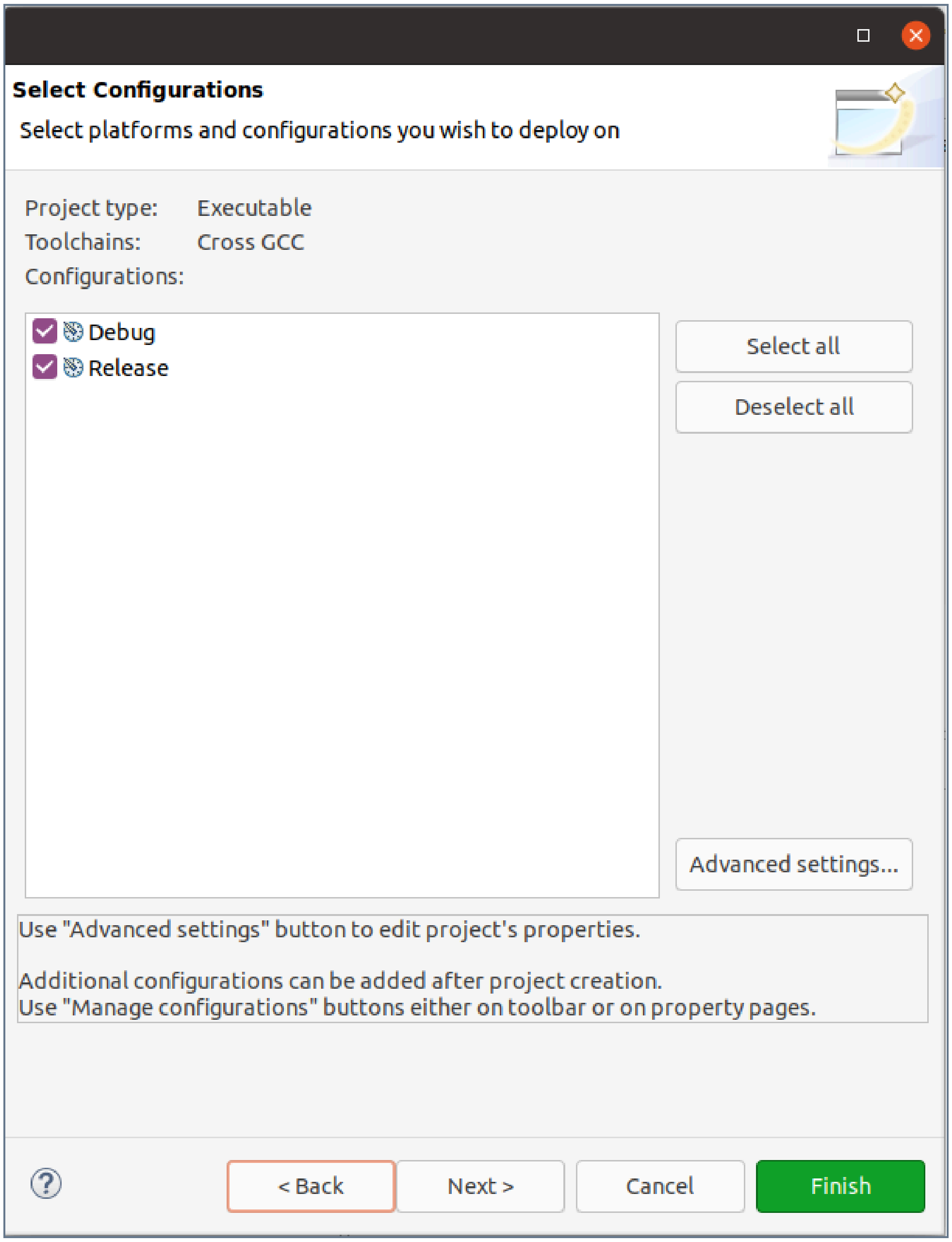
- GNU Autotools
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Shared Library
- Static Library
- Makefile project

Toolchains:

- Cross GCC
- Linux GCC

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

Keep all default settings on the **Select Configuration** screen. Click **Next**.



Select Configurations

Select platforms and configurations you wish to deploy on

Project type: Executable

Toolchains: Cross GCC

Configurations:

☑ Debug

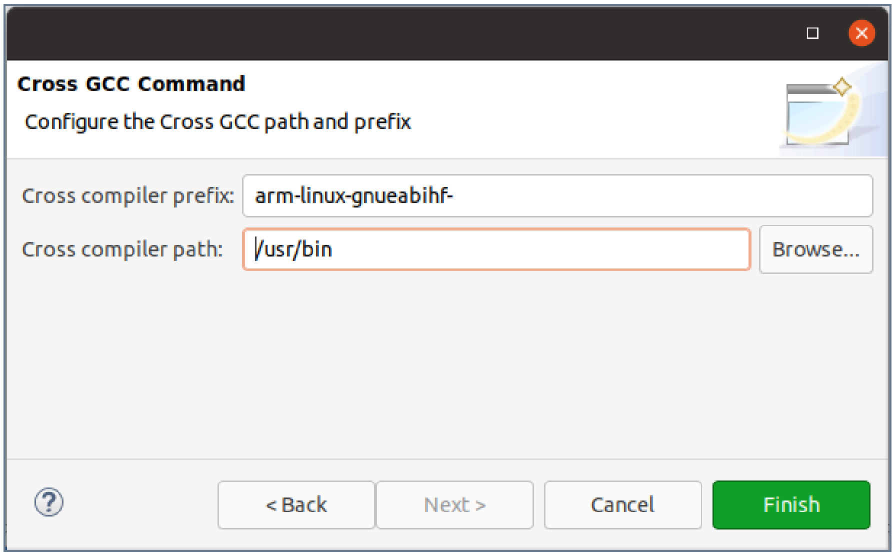
☑ Release

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.

On the **Cross GCC Command** screen:

1. In the **Cross compiler prefix** text box, enter `arm-linux-gnueabihf-`
2. In the **Cross compiler path** text box, enter `/usr/bin`
3. Click **Finish**.



Cross GCC Command

Configure the Cross GCC path and prefix

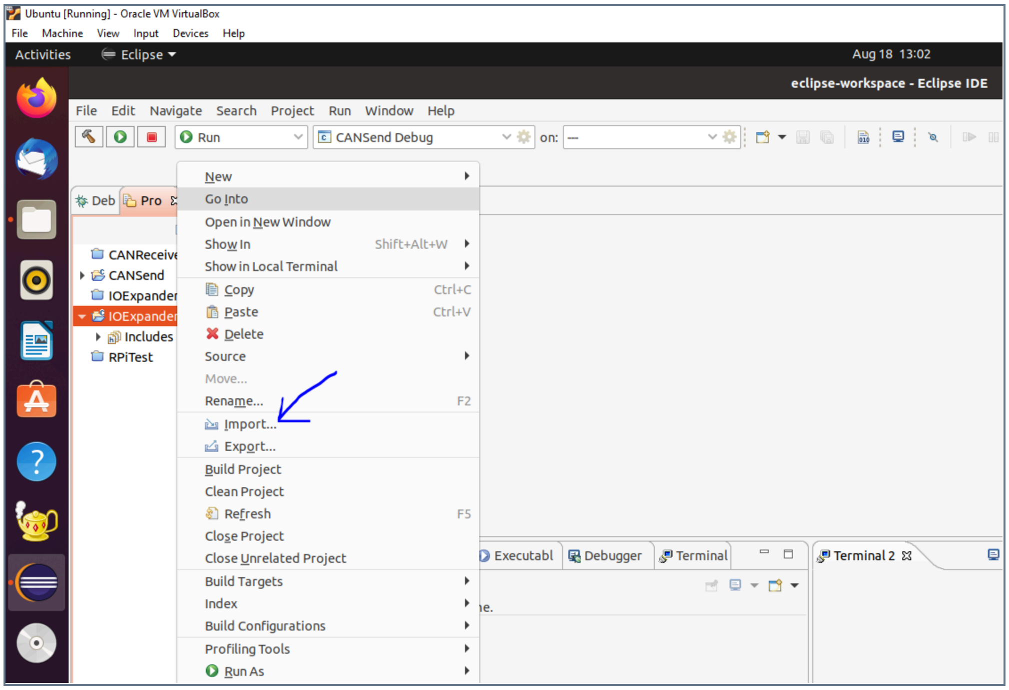
Cross compiler prefix: arm-linux-gnueabihf-

Cross compiler path: /usr/bin

The Eclipse IDE is now configured for cross-compilation using the cross-compilation toolchain. You can write your code or import source code from an existing project.

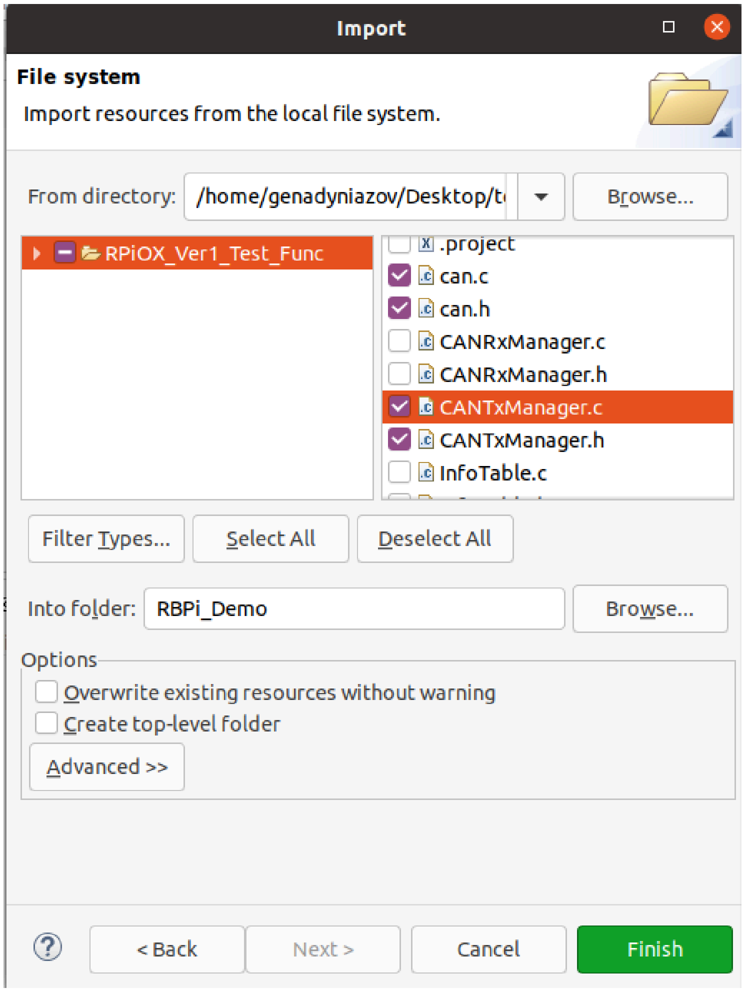
Importing existing source files to the project

Right click on the project name and then click **Import...**



1. Under **General**, select **File System** and then click **Next**.
2. Use the **Browse** button to navigate to the location of the source code folder.
3. From the list on the right, select all source file checkboxes (.c and .h) and then click **Finish**.

All selected source files are now moved into the project folder. See the right screenshot for an example — note how the **CANTxManager.c**, **CANTxManager.h**, **can.c**, and **can.h** files that were selected are now in the project folder.



Import

File system

Import resources from the local file system.

From directory: /home/genadyniazov/Desktop/t

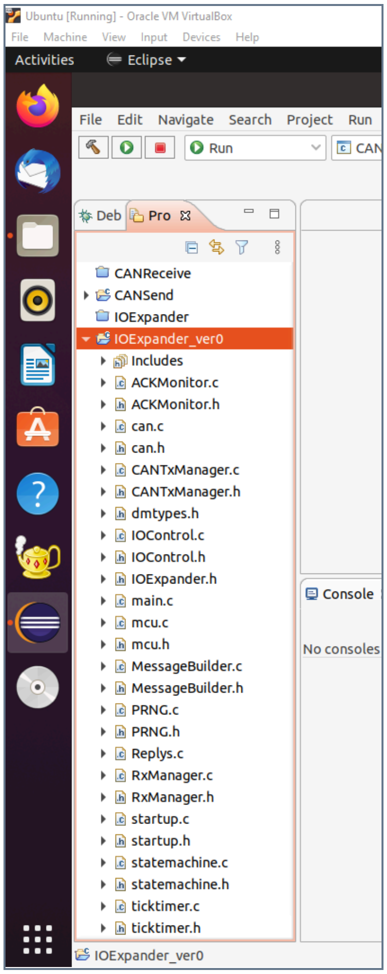
RPiOX_Ver1_Test_Func

- □ .project
- ☑ can.c
- ☑ can.h
- □ CANRxManager.c
- □ CANRxManager.h
- ☑ CANTxManager.c
- ☑ CANTxManager.h
- □ InfoTable.c

Into folder: RBPi_Demo

Options

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

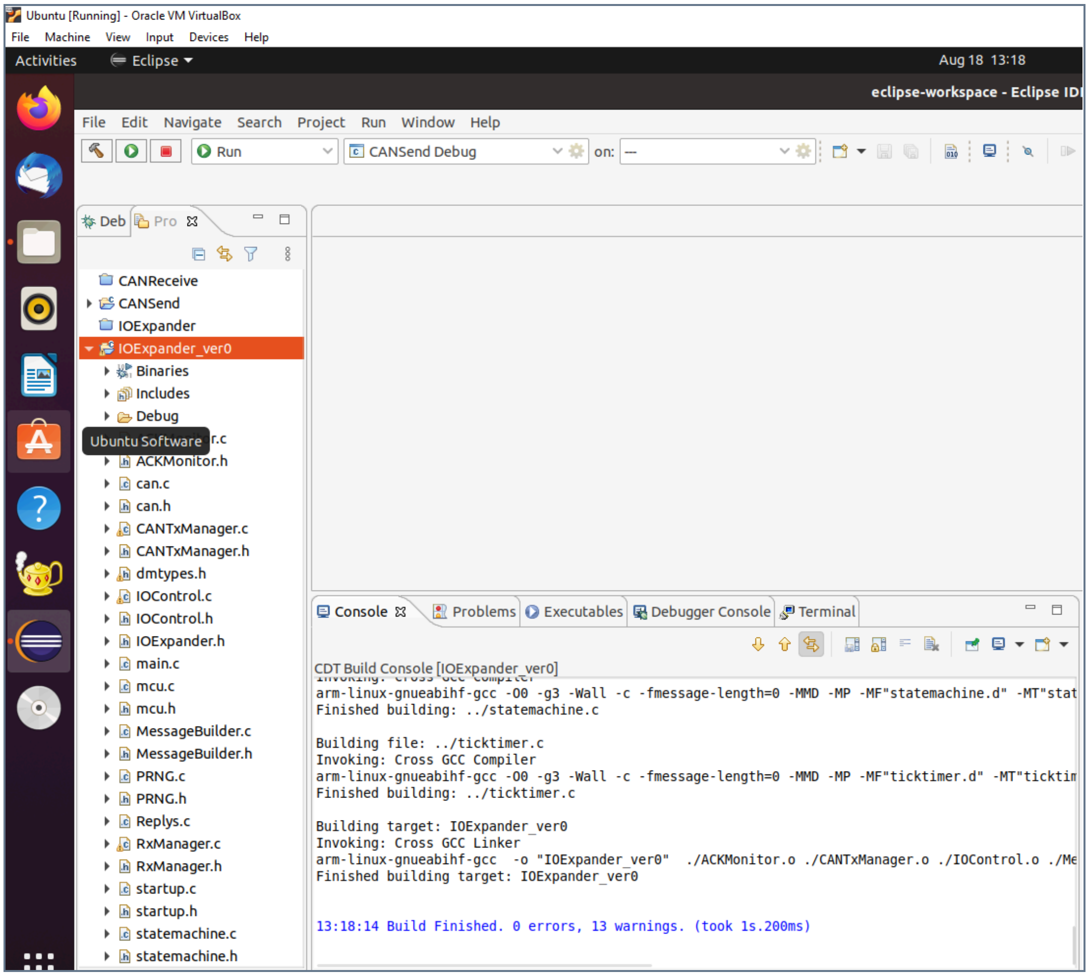


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Eclipse
File Edit Navigate Search Project Run
Run
CAN
Deb
Pro
Console
No consoles

- CANReceive
- CANSend
- IOExpander
- IOExpander_ver0
  - Includes
  - ACKMonitor.c
  - ACKMonitor.h
  - can.c
  - can.h
  - CANTxManager.c
  - CANTxManager.h
  - dmtypes.h
  - IOControl.c
  - IOControl.h
  - IOExpander.h
  - main.c
  - mcu.c
  - mcu.h
  - MessageBuilder.c
  - MessageBuilder.h
  - PRNG.c
  - PRNG.h
  - Replys.c
  - RxManager.c
  - RxManager.h
  - startup.c
  - startup.h
  - statemachine.c
  - statemachine.h
  - ticktimer.c
  - ticktimer.h

IOExpander_ver0

Next, you should be able to build the project. Right click the project name and click **Build Project**. If everything is configured properly, the build finishes with 0 errors.



CDT Build Console [IOExpander_ver0]

```
Invoking: Cross GCC Compiler
arm-linux-gnueabihf-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"statemachine.d" -MT"stat
Finished building: ../statemachine.c

Building file: ../ticktimer.c
Invoking: Cross GCC Compiler
arm-linux-gnueabihf-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"ticktimer.d" -MT"ticktim
Finished building: ../ticktimer.c

Building target: IOExpander_ver0
Invoking: Cross GCC Linker
arm-linux-gnueabihf-gcc  -o "IOExpander_ver0"  ./ACKMonitor.o ./CANTxManager.o ./IOControl.o ./Me
Finished building target: IOExpander_ver0


13:18:14 Build Finished. 0 errors, 13 warnings. (took 1s.200ms)
```

Remote debugging

To allow debugging communication between the host machine and Raspberry Pi, you must install gdbserver on Raspberry Pi. This section describes how to do this.

For more information about gdbserver, refer to the following link:
https://www.man7.org/linux/man-pages/man1/gdbserver.1.html

Open the terminal command prompt and enter the following command:

```
sudo apt-get install gdbserver
```

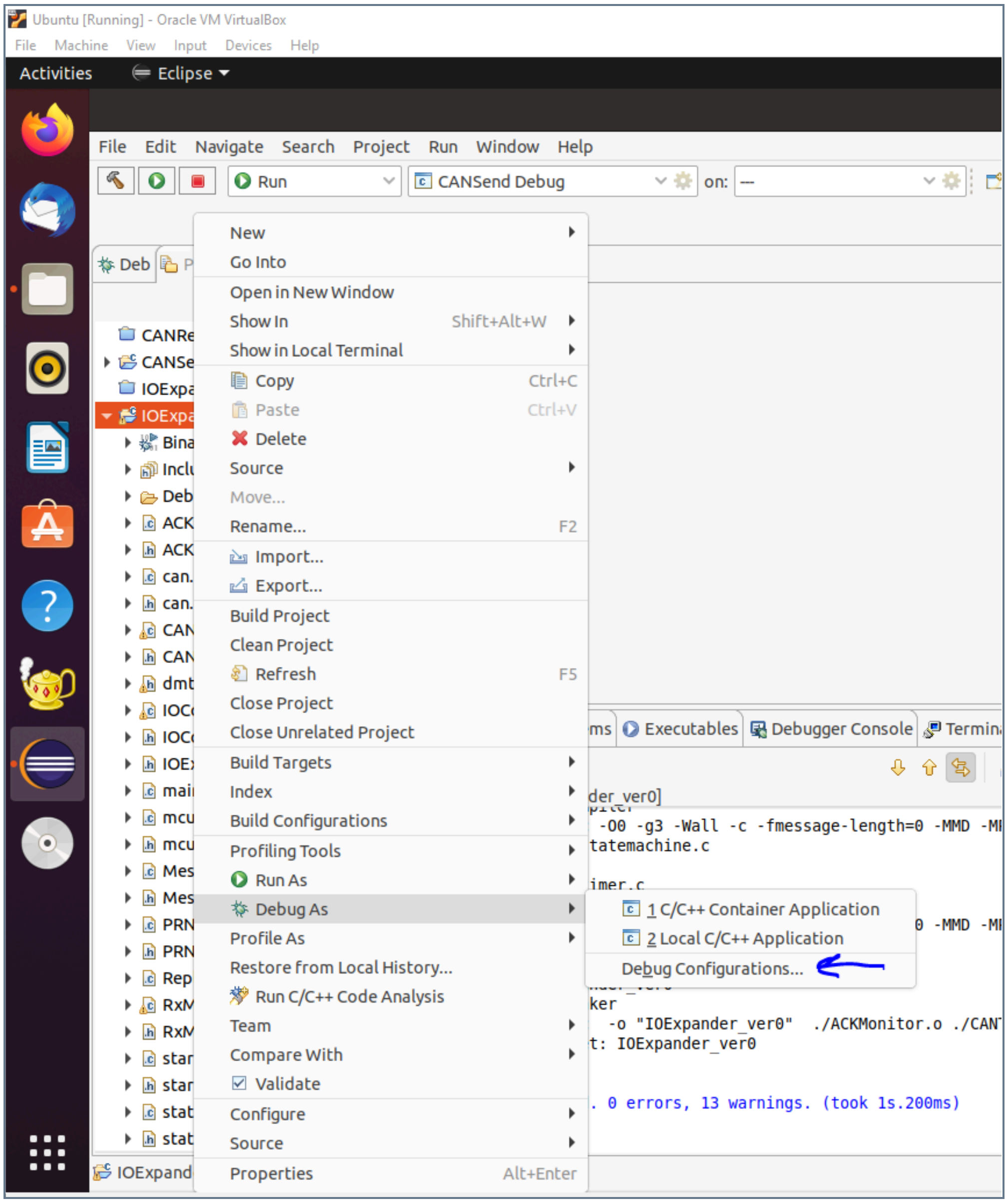
Now, go back to the host machine to configure Eclipse for remote debugging.

Open the terminal and install gdb-multiarch to support the target's ARM architecture on the host machine by typing the following command:

```
sudo apt-get install gdb-multiarch
```

Next, Eclipse must be configured so that it can connect to the RPi's gdb server.

In Eclipse, right click the project name and then select **Debug As** > **Debug Configurations...**



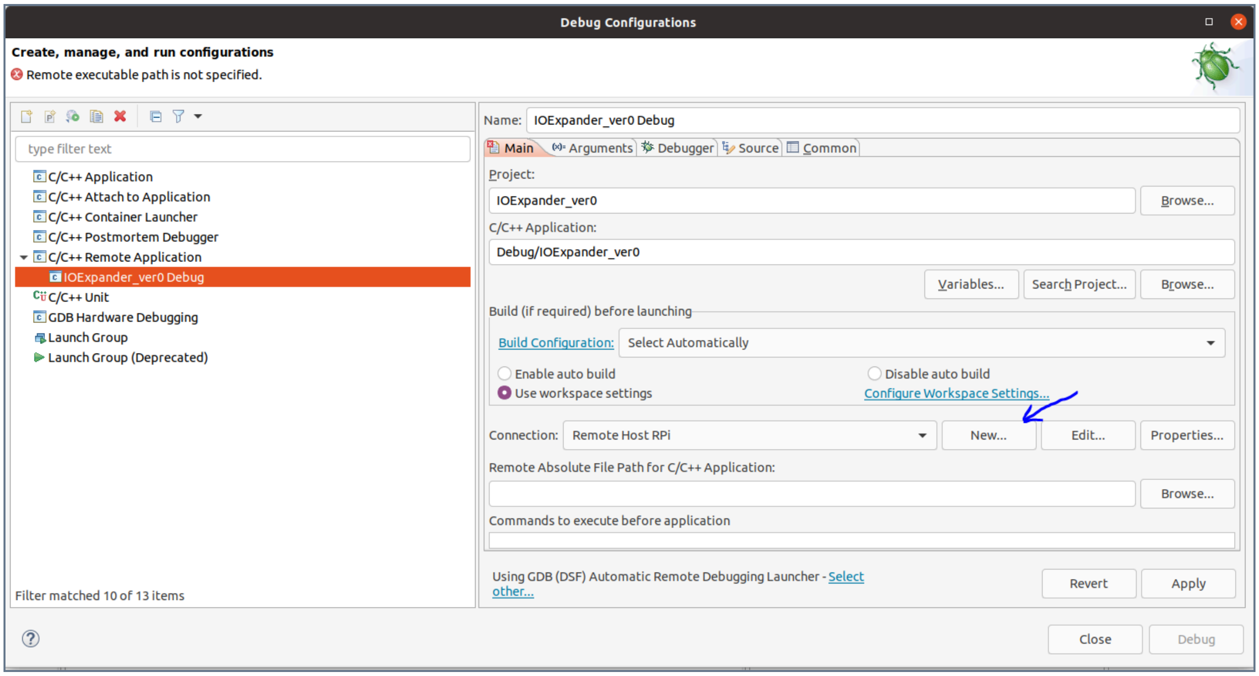
```
der_ver0]
 -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MI
tatemachine.c

imer.c
                                         0 -MMD -MI

ker
 -o "IOExpander_ver0"  ./ACKMonitor.o ./CAN
t: IOExpander_ver0

. 0 errors, 13 warnings. (took 1s.200ms)
```

In the **Debug Configurations** window, double-click **C/C++ Remote Application**. This will create a new remote debug configuration session. In the **Connection** section, click **New**.



Debug Configurations

Create, manage, and run configurations

Remote executable path is not specified.

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - IOExpander_ver0 Debug
- C/C++ Unit
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)

Filter matched 10 of 13 items

Name: IOExpander_ver0 Debug

Project: IOExpander_ver0

C/C++ Application: Debug/IOExpander_ver0

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build
○ Disable auto build
● Use workspace settings

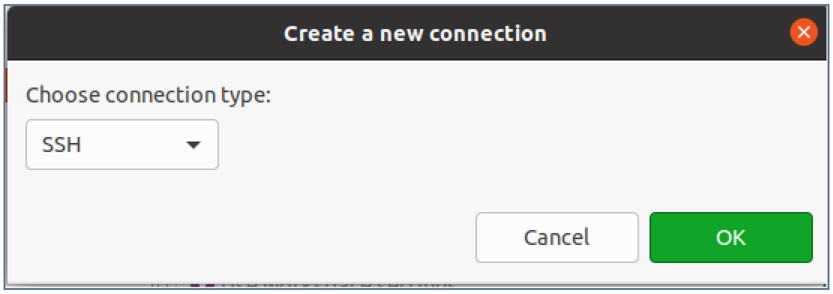
Connection: Remote Host RPi

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

Commands to execute before application

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

In the dialog that appears, select **SSH** from the **Choose connection type** drop-down menu. Click **OK**.

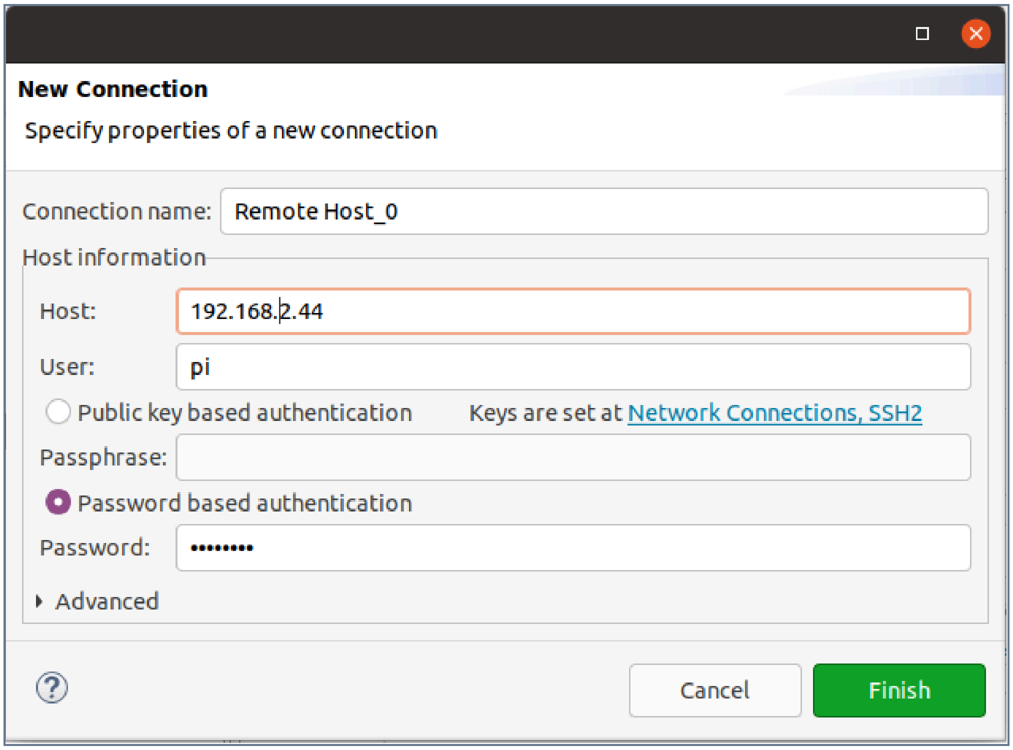


Create a new connection

Choose connection type: SSH

In the **New Connection** dialog, do the following:

1. In the **Host** text box, enter the IP address of the remote target (the Raspberri Pi).
2. In the **User** text box, enter the username of the remote target. This is “pi” by default.
3. Select the **Password based authentication** radio button and then enter the password of the remote target.
4. Click **Finish**.



New Connection

Specify properties of a new connection

Connection name: Remote Host_0

Host information

Host: 192.168.2.44

User: pi

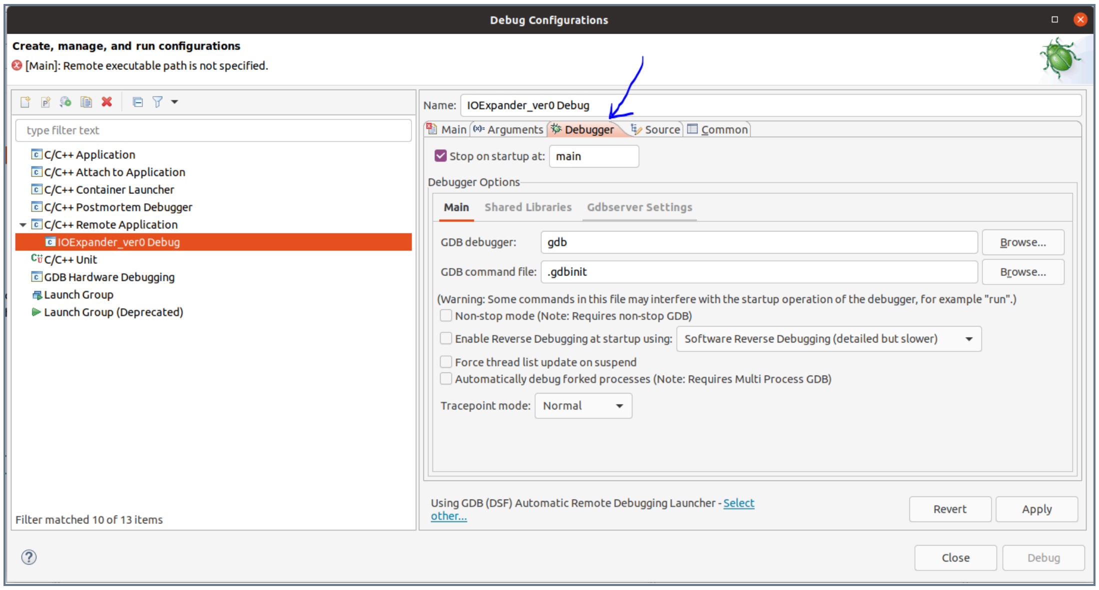
○ Public key based authentication   Keys are set at Network Connections, SSH2

Passphrase:

● Password based authentication

Password: ••••••••

Back in the **Debug Configurations** window, select the **Debugger** tab, change the GDB debugger text box from gdb to gdb-multiarch, and then click **Apply**.



Debug Configurations

Create, manage, and run configurations

[Main]: Remote executable path is not specified.

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - IOExpander_ver0 Debug
- C/C++ Unit
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)

Filter matched 10 of 13 items

Name: IOExpander_ver0 Debug

☑ Stop on startup at: main

Debugger Options

GDB debugger: gdb

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 using: Software Reverse Debugging (detailed but slower)

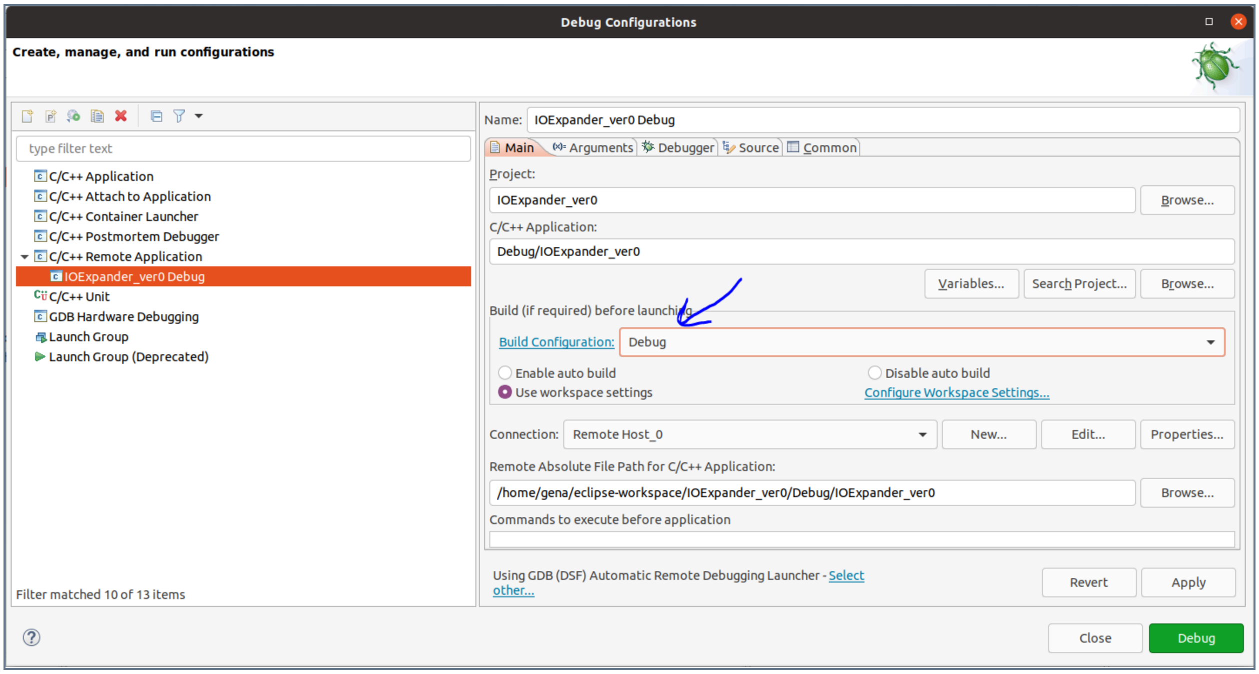
□ Force thread list update on suspend

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

Tracepoint mode: Normal

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

Still in **Debug Configurations**, go to the **Main** tab. In the **Build (if required before launching)** section, select "Debug" from the **Build Configuration** drop-down menu.



Debug Configurations

Create, manage, and run configurations

Name: IOExpander_ver0 Debug

Project: IOExpander_ver0

C/C++ Application: Debug/IOExpander_ver0

Build (if required) before launching

Build Configuration: Debug

○ Enable auto build ○ Disable auto build

● Use workspace settings

Connection: Remote Host_0

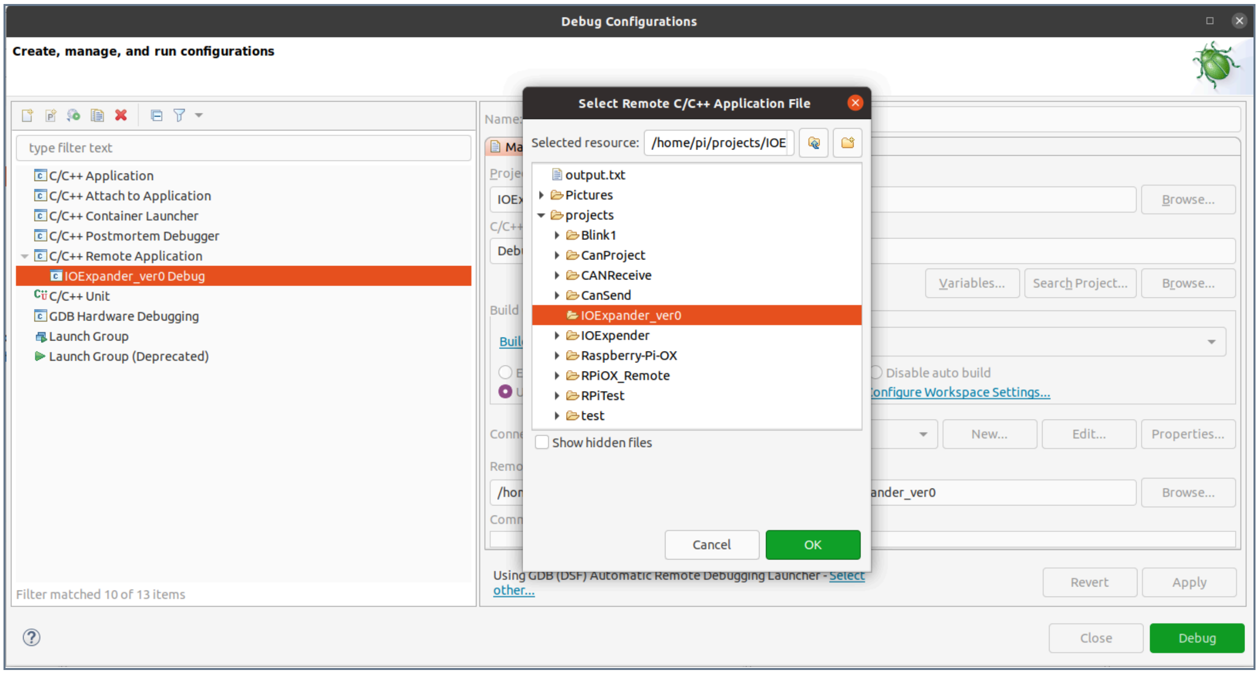
Remote Absolute File Path for C/C++ Application: /home/gena/eclipse-workspace/IOExpander_ver0/Debug/IOExpander_ver0

Commands to execute before application

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

Filter matched 10 of 13 items

Next, under **Remote Absolute File Path for C/C++ Application**, click **Browse** and navigate to the folder in the remote target (the RPi) where the executable file is located and then click **OK**.



Debug Configurations

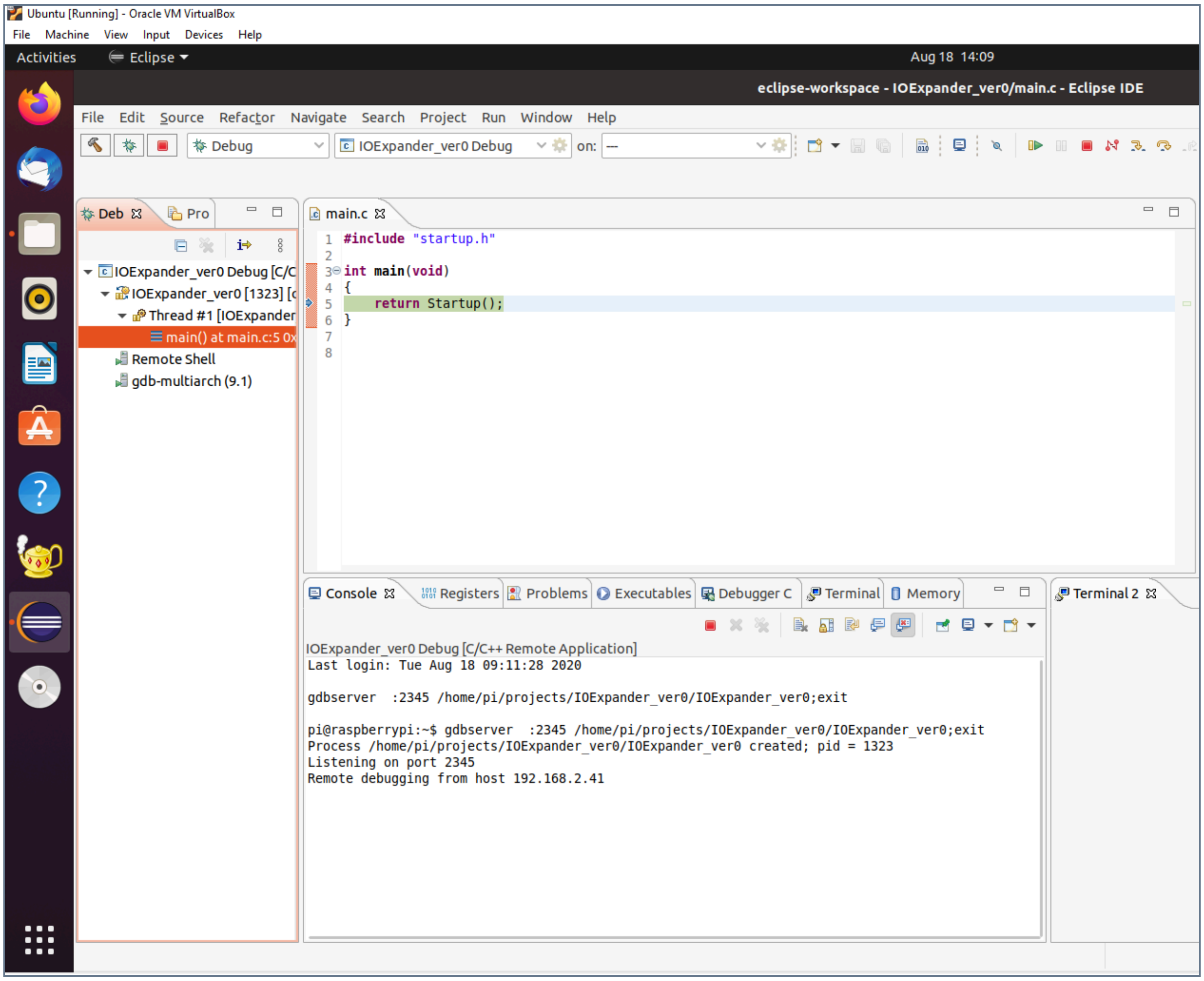
Create, manage, and run configurations

Select Remote C/C++ Application File

Selected resource: /home/pi/projects/IOE

□ Show hidden files

Finally, click **Apply** and then click **Debug** to start the debugging session with the Raspberry Pi.


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Eclipse
Aug 18 14:09
eclipse-workspace - IOExpander_ver0/main.c - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
IOExpander_ver0 Debug
on: —
Deb
Pro
IOExpander_ver0 Debug [C/C
IOExpander_ver0 [1323] [c
Thread #1 [IOExpander
main() at main.c:5 0x
Remote Shell
gdb-multiarch (9.1)
main.c
Console
Registers
Problems
Executables
Debugger C
Terminal
Memory
Terminal 2

```
#include "startup.h"

int main(void)
{
    return Startup();
}
```

IOExpander_ver0 Debug [C/C++ Remote Application]

```
Last login: Tue Aug 18 09:11:28 2020

gdbserver  :2345 /home/pi/projects/IOExpander_ver0/IOExpander_ver0;exit

pi@raspberrypi:~$ gdbserver  :2345 /home/pi/projects/IOExpander_ver0/IOExpander_ver0;exit
Process /home/pi/projects/IOExpander_ver0/IOExpander_ver0 created; pid = 1323
Listening on port 2345
Remote debugging from host 192.168.2.41
```

Configuring the CAN Communications

For software installation and configuration settings of PiCAN3 CAN Bus Board on Raspberry Pi 4, see:

https://copperhilltech.com/content/PICAN3_UGA_10.pdf

For a hardware schematic of the PiCAN3, see:

https://copperhilltech.com/content/pican3_rev_C.pdf