

Using GCC/GDB with Ubimote_HR

Version 1.0

© CDAC KP Bangalore

<http://www.cdac.in>

Contents

1. System Requirements
2. Install ARM toolchain
3. Install Eclipse
4. Testing Eclipse with tool chain
5. Configuring Build settings
6. Building Project
7. Setting up Debugger
8. Create Debug configuration Using GDB
9. Debugging using GDB
10. Clone Ubimote HR SDK
11. Loading Projects into Eclipse

1. System Requirements

- a. Ubuntu 12.04 and above

2. Install ARM tool chain

Add the repository for gcc-arm-embedded:

```
$ sudo add-apt-repository ppa:terry.guo/gcc-arm-embedded
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install gcc-arm-none-eabi
```

Then check your version:

```
$ arm-none-eabi-gcc --version
arm-none-eabi-gcc (4.8.2-14ubuntu1+6) 4.8.2
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.
```

3. Install Eclipse

Follow these instructions to install Eclipse and CDT on a Linux platform. Note that the Eclipse IDE is dependent on Java Runtime Environment being installed on the machine. Make sure JRE is installed, and that the installation path is added to your environment variables, before the installation of Eclipse. Note that both JRE and Eclipse must be downloaded for the same platform (the 64-bit version of Eclipse requires the 64-bit version of JRE).

3.1. Download **Eclipse CDT** for C/C++ developers from <http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/lunasr2>

Note: Do not install Eclipse using apt or Ubuntu software centre. It is outdated and the plugins require a more recent version of Eclipse

- 3.2. unzip eclipse in your favourite folder e.g., /opt/eclipse
- 3.3. Execute eclipse (#sudo ./eclipse from terminal) and go to Help->Install New Software.
- 3.4. Add gnu-arm eclipse plug-in url and install the plug-in. Use the following url.

<http://gnuarmeclipse.sourceforge.net/updates>

- 3.5. Restart eclipse when installation finishes

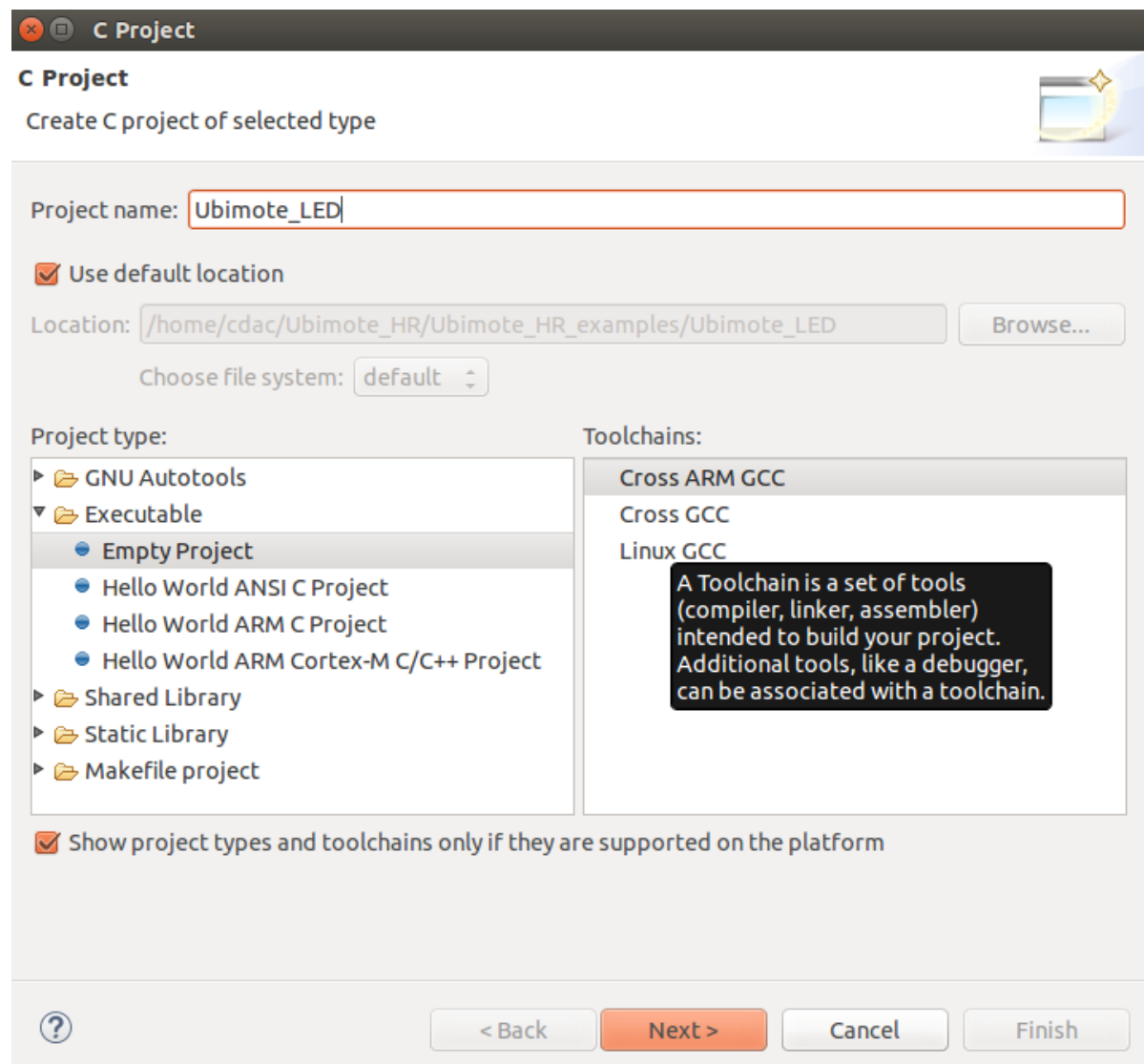
4. Testing Eclipse with tool chain

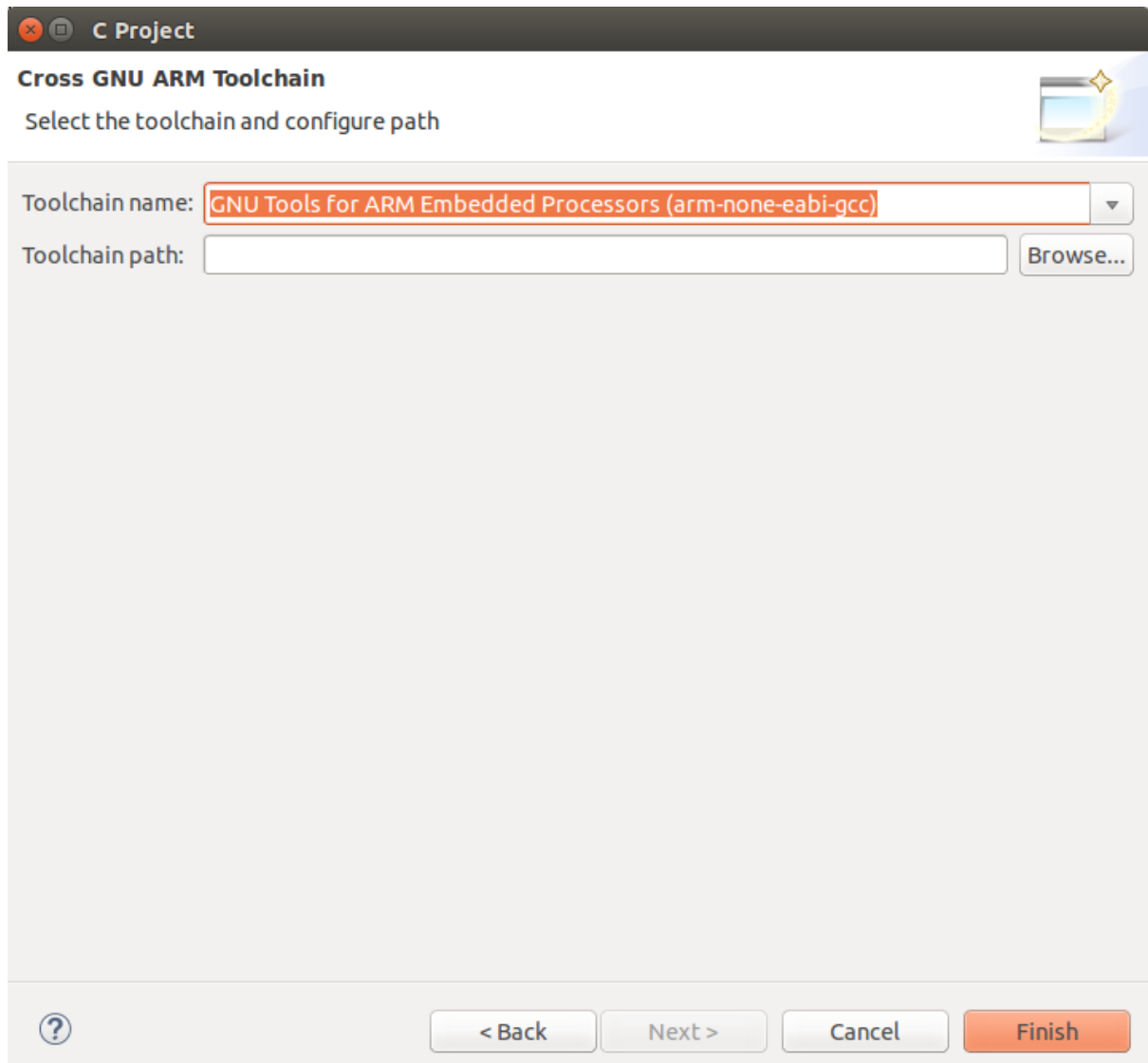
Once the gnu-arm plug-in is installed you can test that everything works by creating a test project.

To do so, click **File > New > C Project** and select **Project Type: Executable** and **Toolchains: Cross ARM GCC**

Give **Next >** select both the configuration **Debug** and **Release**.

Next > select tool chain as **GNU Tools for ARM Embedded Processors (arm-none-eabi-gcc)**



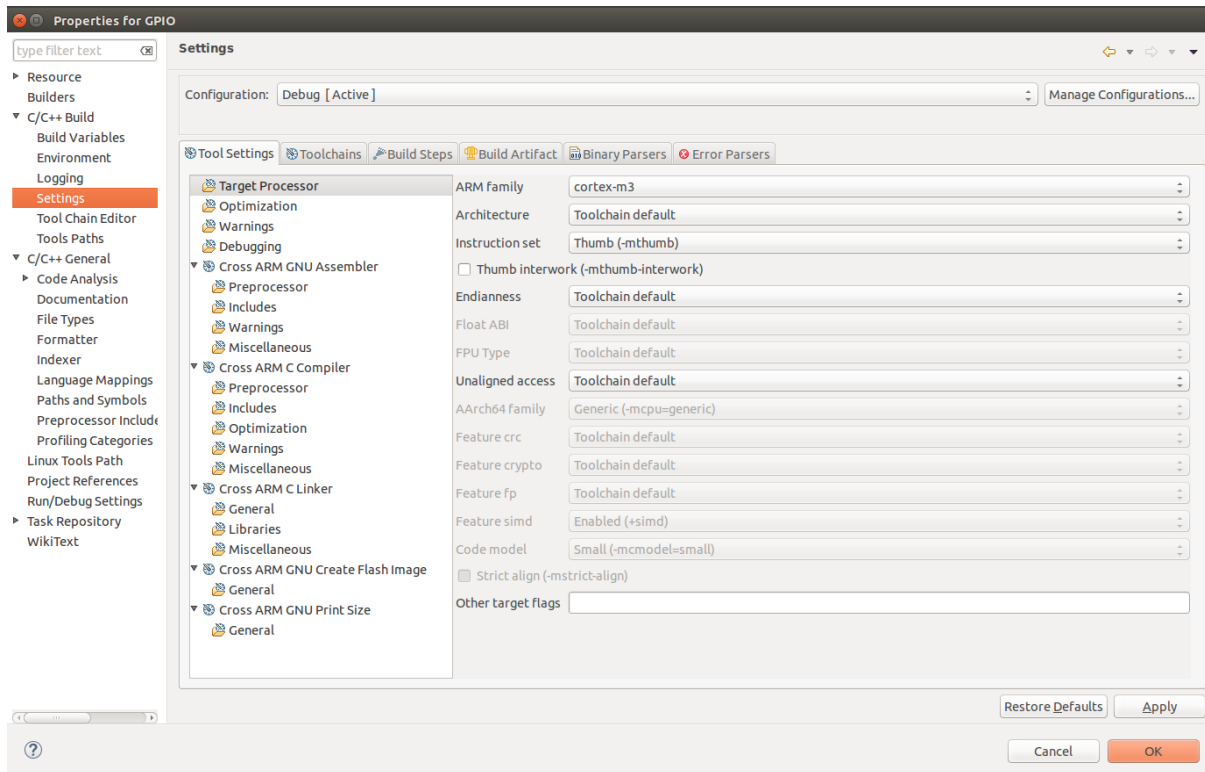


Name your project (e.g. "GPIO") and click Finish.

5. Configuring Build Settings

Right click on the project and select properties:

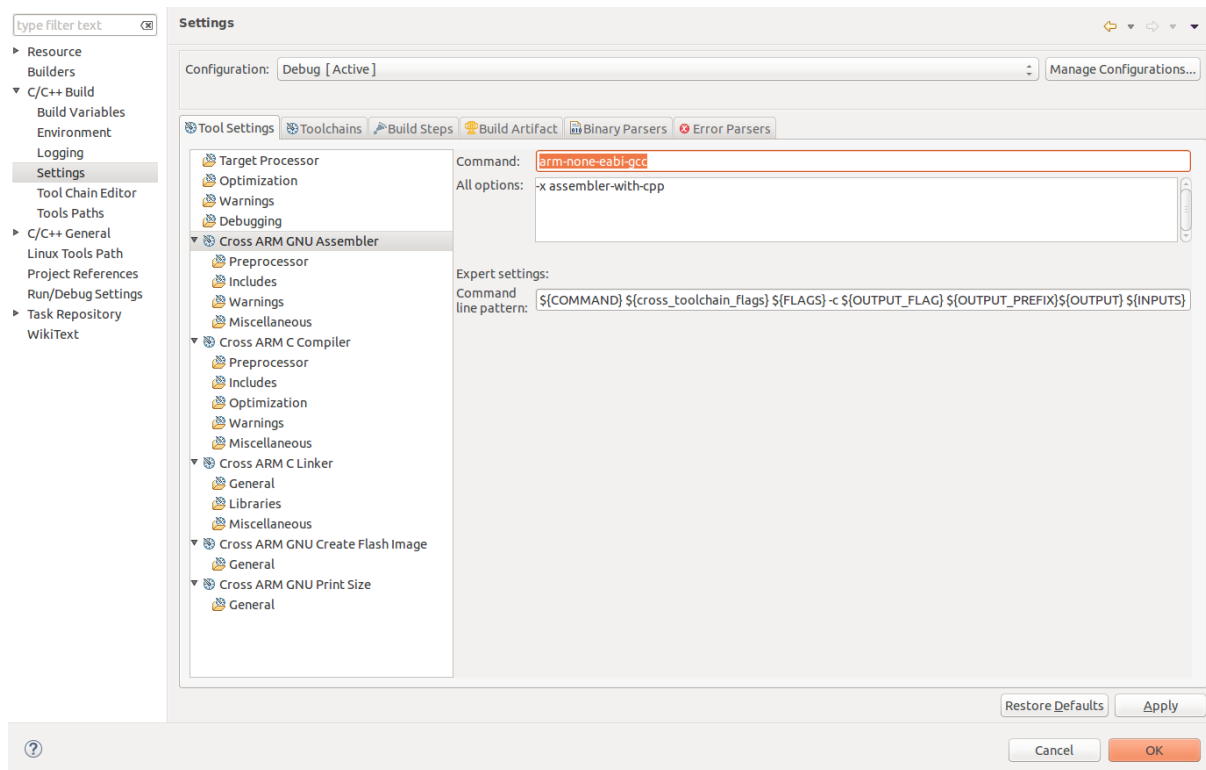
C/C++ Build -> Settings



Select the micro-controller architecture, for example Cortex M3

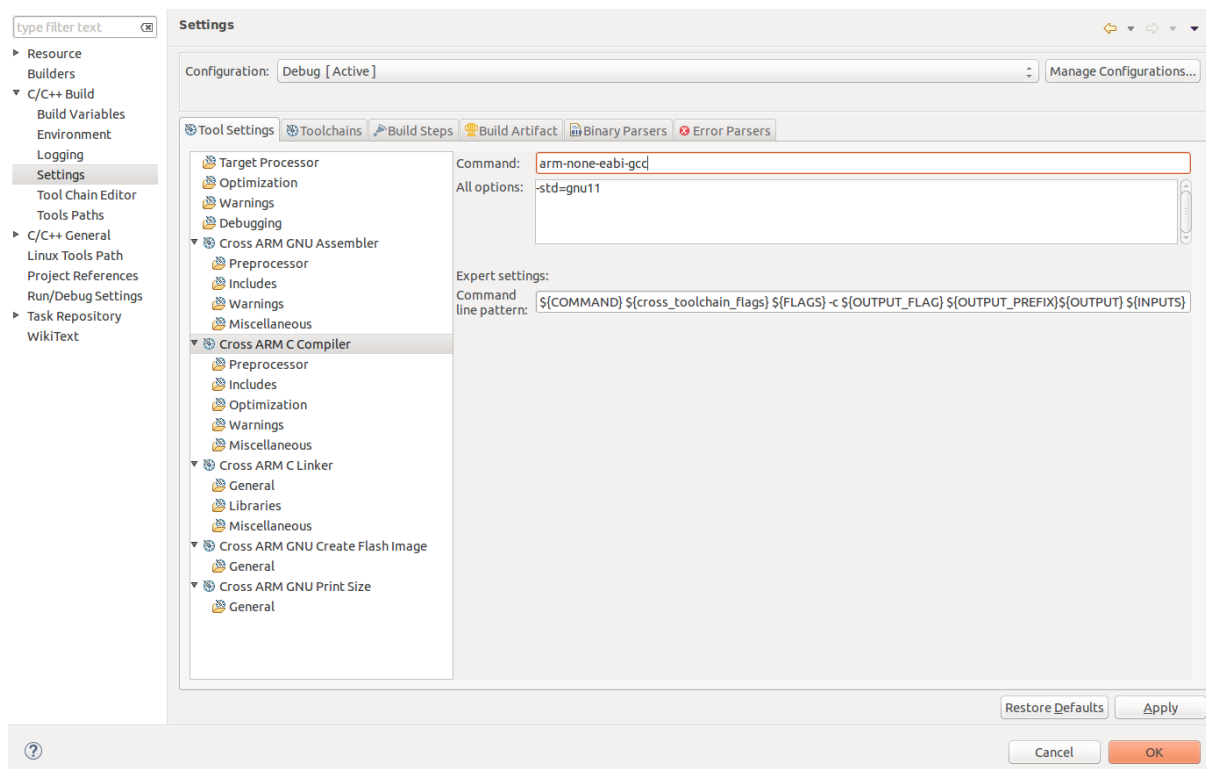
Move to the "assembler" option and set up the path to your gcc-arm-none/ directory, type arm-none-eabi-gcc

Note: If you have updated your computer PATH environment variable, you can just enter arm-none-eabi-gcc, without the full path.

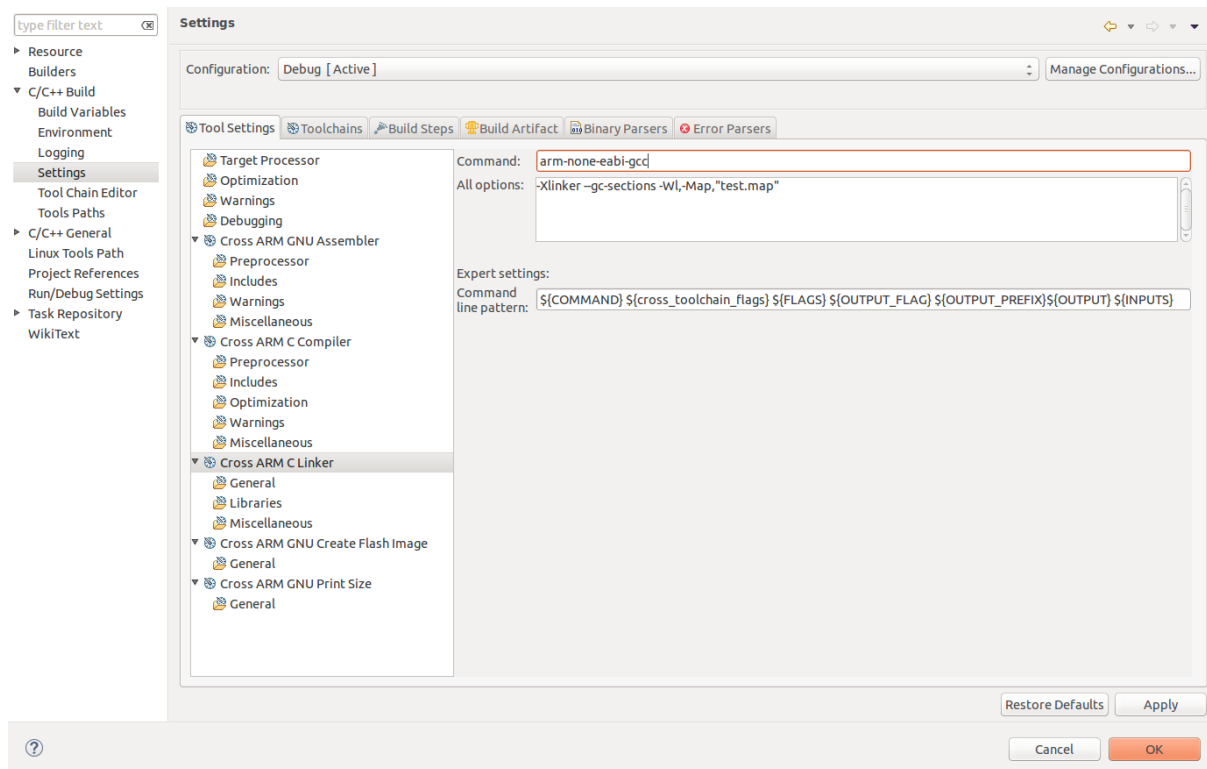


Move to the "Compiler" option and set up the path to your gcc-arm-none/ directory, type arm-none-eabi-gcc

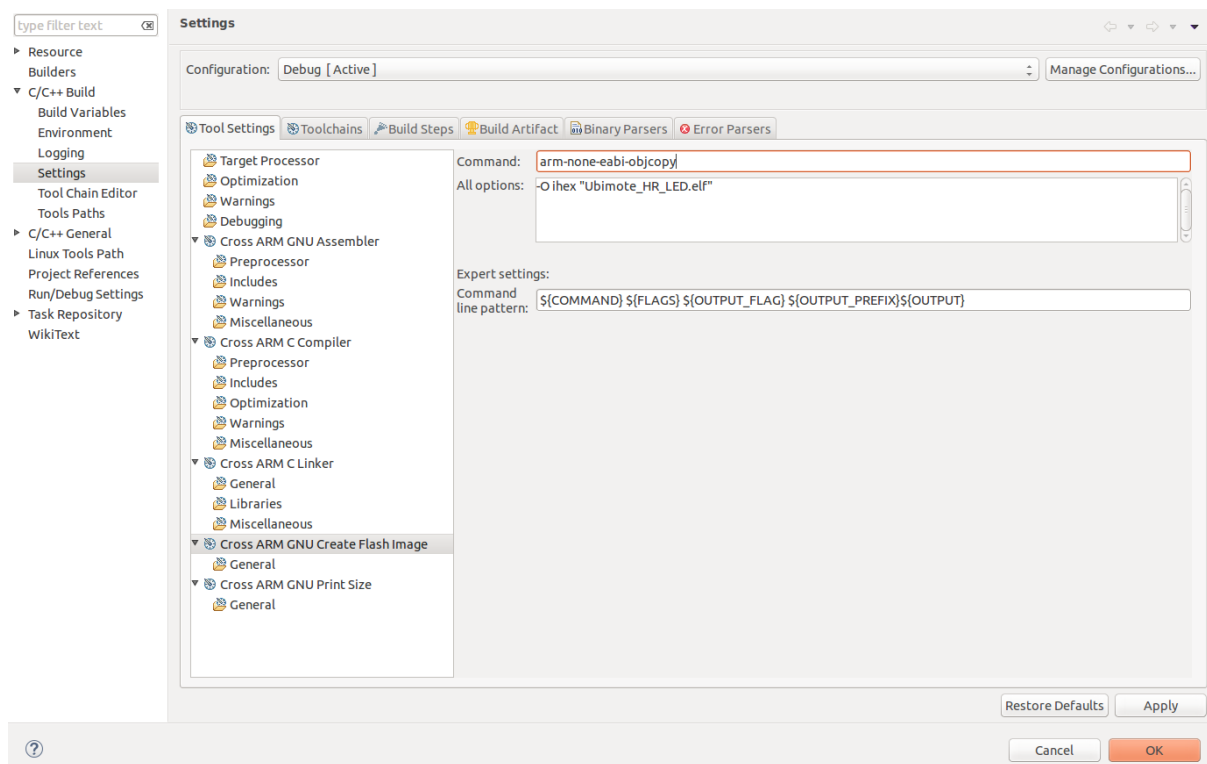
Note: If you have updated your computer PATH environment variable, you can just enter arm-none-eabi-gcc, without the full path.



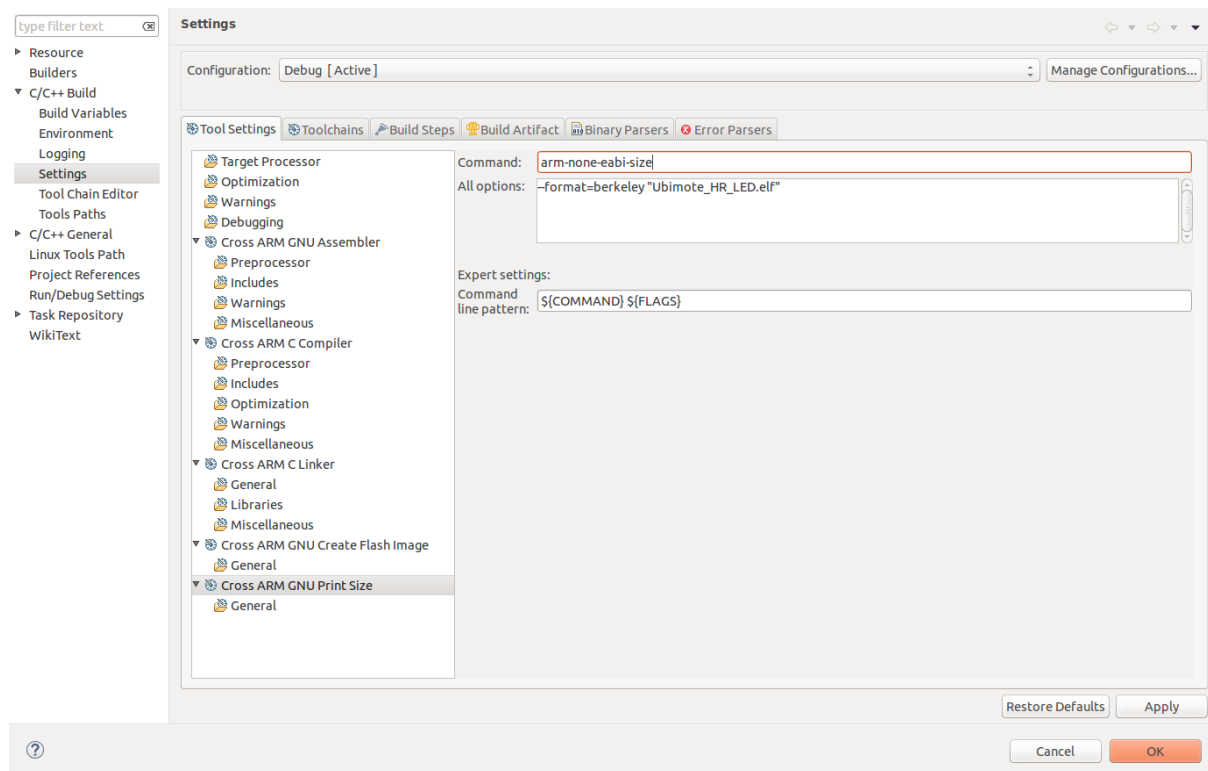
Repeat the same steps for the linker, which in our case is also `arm-none-eabi-gcc`.



Configure the "Flash image creator" as follows:



Configure the "Print Size" as follows:



At this point, configure any extra options you might want, such as optimizations and debug levels. Click OK to apply all changes.

Create a C file to test the compiler. You can toggle an LED according to your hardware specification or simply code a while loop and test the compiler.

```
int main(void)
{
    int i = 0;
    while (i != 10)
    {
        i++;
    }
    return i;
}
```

6. Building Project

Right-click on the project and select Build Project. Based on your configuration, Eclipse invokes the ARM GCC compiler, linker, flash image creator, and size printer tools.

```
13:58:57 **** Build of configuration Debug for project Ubimote_HR_LED ****
make all
Building file: ../main.c
Invoking: Cross ARM C Compiler
arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -O0 -fmessage-length=0 -fsigned-char -
ffunction-sections -fdata-sections -g3 -std=gnu11 -MMD -MP -MF"main.d" -MT"main.o"
-c -o "main.o" "../main.c"
Finished building: ../main.c

Building target: Ubimote_HR_LED.elf
Invoking: Cross ARM C Linker
arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -O0 -fmessage-length=0 -fsigned-char -
ffunction-sections -fdata-sections -g3 -nostartfiles -nostdlib -Wl,-
Map,"Ubimote_HR_LED.map" -o "Ubimote_HR_LED.elf" ./main.o
/usr/bin/../../lib/gcc/arm-none-eabi/4.9.3/../../../../arm-none-eabi/bin/ld: warning:
cannot find entry symbol _start; defaulting to 00000000000008000
Finished building target: Ubimote_HR_LED.elf

Invoking: Cross ARM GNU Create Flash Image
arm-none-eabi-objcopy -O ihex "Ubimote_HR_LED.elf" "Ubimote_HR_LED.hex"
Finished building: Ubimote_HR_LED.hex

Invoking: Cross ARM GNU Print Size
arm-none-eabi-size --format=berkeley "Ubimote_HR_LED.elf"
   text          data           bss             dec             hex             filename
   40             0             0             40             28
   Ubimote_HR_LED.elf
Finished building: Ubimote_HR_LED.siz

13:58:57 Build Finished (took 56ms)
```

Note: Error related to the Linker will be suppressed by enabling “Do not use standard start files” under File -> Properties -> Tool settings -> Cross ARM C Linker -> General

7. Setting up of Debugger

Download latest [Jlink](#) software from segger

Download the following files, will be used during the project configuration.
Or

You can use the linker and startup files as part of the Ubimote_HR SDK provided.

A linker file for CC2538SF53

wget

https://raw.githubusercontent.com/CDACBANG/Ubimote_HR/master/cc2538.lds

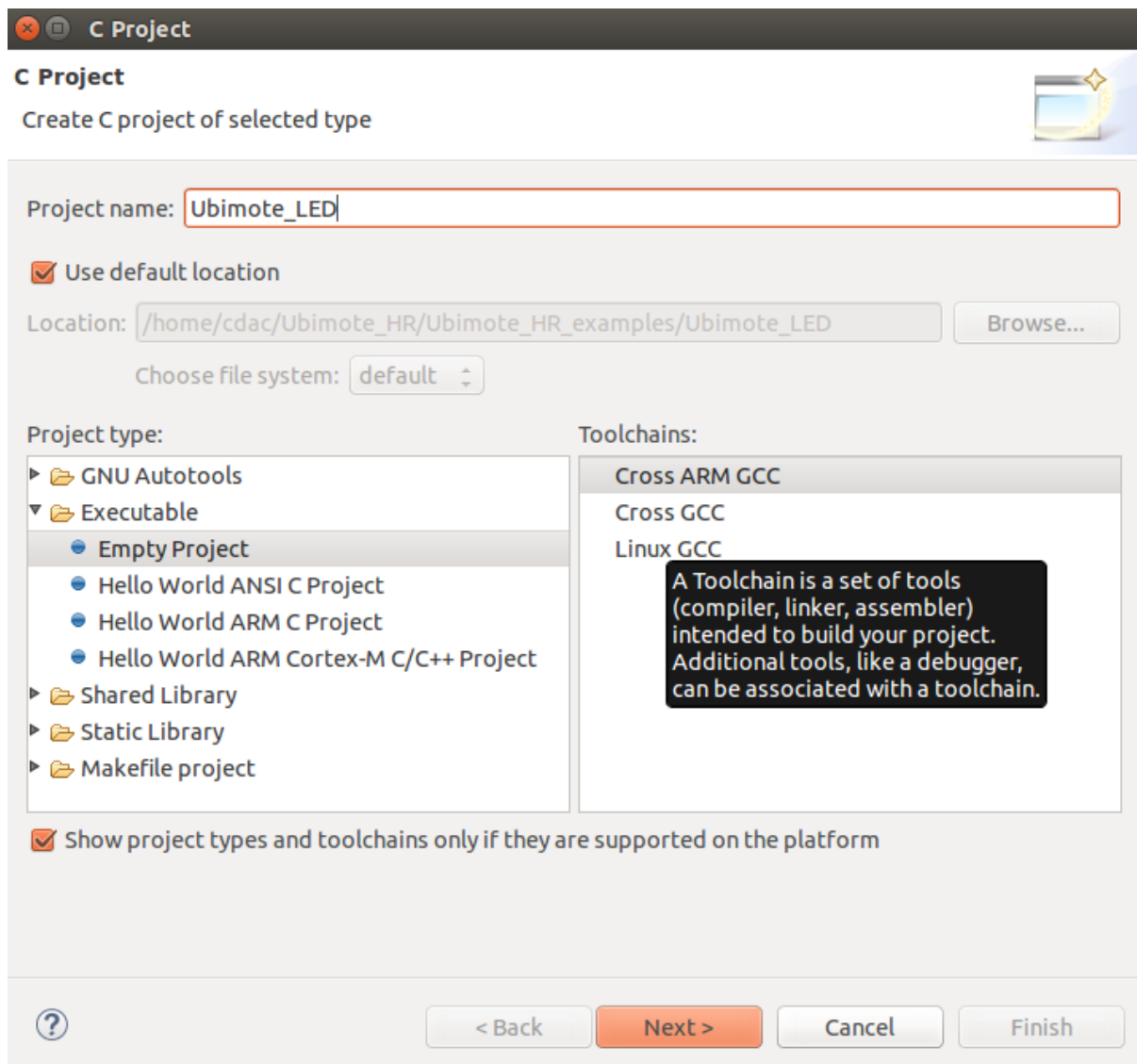
A startup file for the ARM GNU GCC toolchain

wget

https://raw.githubusercontent.com/CDACBANG/Ubimote_HR/master/startup_gcc.c

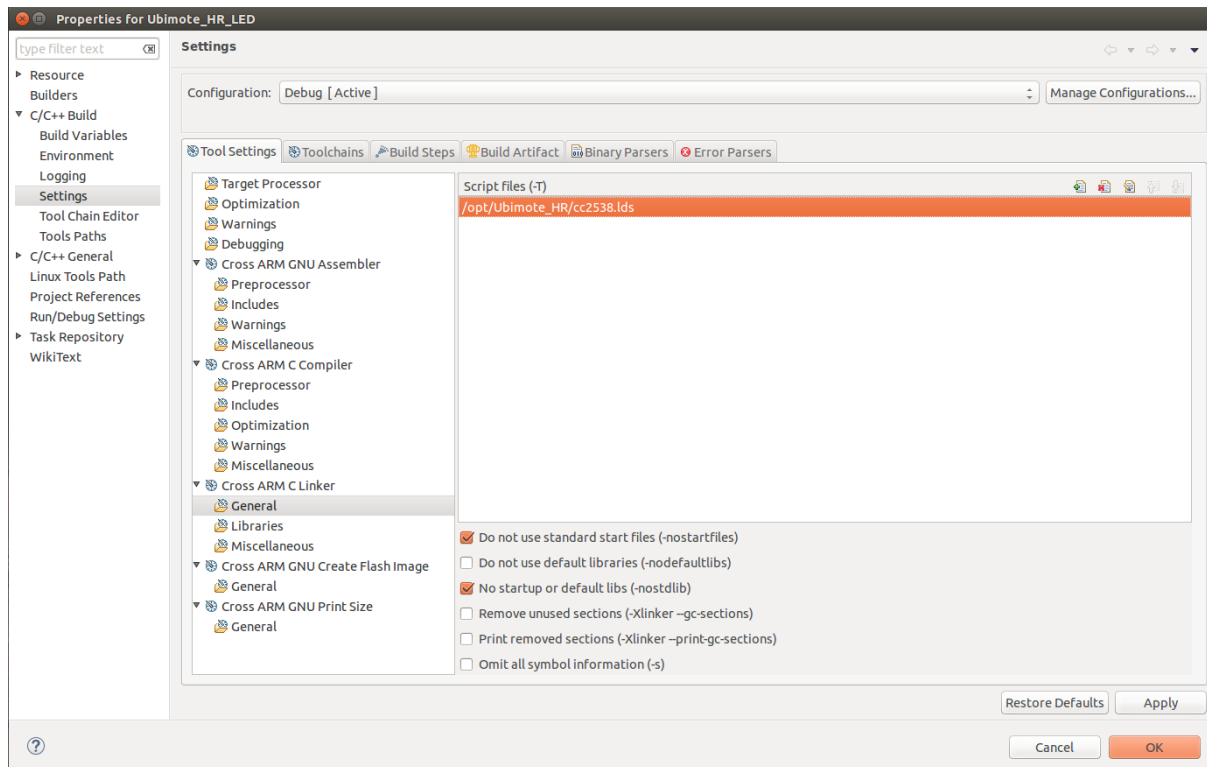
Configuration

- Create a new project from Eclipse: Select File -> New -> C Project
- Choose Executable -> Empty Project
- Choose the Cross ARM GCC
- Select a name for the project



In the project properties, configure the project to use the arm-gnu tool chain as above.

Go to Linker settings and In General select the path to the linker file (downloaded previously)



Click OK to save the changes.

In your project folder structure add the downloaded files (linker file and startup file) and create a main program containing the following code:

```
#include <stdint.h>
#define GPIO_C_DIR          0x400DB400
#define GPIO_C_DATA         0x400DB000
#define HWREG(x)  (*((volatile uint32_t *) (x)))

int main(void)
{
    volatile uint32_t ui32LoopCount;
    volatile uint32_t ui32LoopXV=0;
    HWREG(GPIO_C_DIR) |= 0xE0;
    HWREG(GPIO_C_DATA + (0xE0 << 2)) = 0;

    while(1)
    {
        HWREG(GPIO_C_DATA + (0xE0 << 2)) = 0x00;
        for(ui32LoopCount = 200000; ui32LoopCount > 0; ui32LoopCount--)
        {
            ui32LoopXV++;
        }
        HWREG(GPIO_C_DATA + (0xE0 << 2)) = 0xE0;
        for(ui32LoopCount = 200000; ui32LoopCount > 0; ui32LoopCount--)
        {
            ui32LoopXV++;
        }
    }
}
```

Make sure you can compile the code by using the "hammer". The console output should indicate something like that.

```
13:37:30 **** Incremental Build of configuration
Debug for project Ubimote_HR_LED****
make all
Invoking: ARM Windows GNU Print Size
arm-none-eabi-size  --format=berkeley
Ubimote_HR_LED.elf
   text      data        bss          dec
hex      filename
   824         0         512         1336
538      Ubimote_HR_LED.elf
Finished building: Ubimote_HR_LED.siz

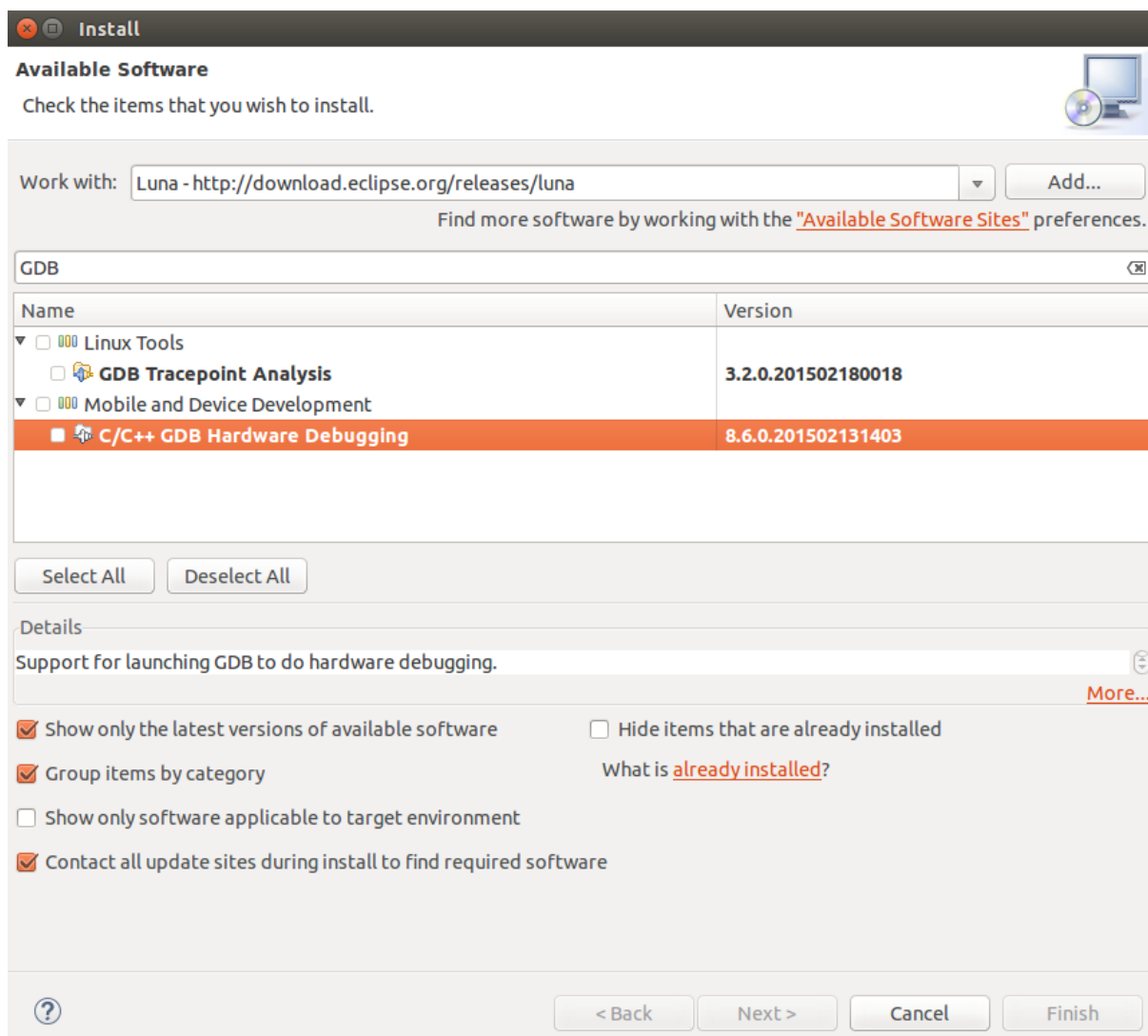
13:37:33 Build Finished (took 2s.612ms)
```


8. Create a Debug Configuration using GDB

Before starting a Debug configuration we have to make sure that the GDB hardware debug plugin in Eclipse is installed. To install it go to

Help -> Install New Software

Use the filtering option to find the GDB plug-in and select the GDB Hardware Debugging option.



This will require you to restart eclipse.

Before starting a debug configuration GDB server needs to be started. This is a tool provided by Segger and can be started by executing Segger J-link GDB server from the segger application menu.

The following command will allow you to configure the target chip and the connection features.

```
$ JLinkGDBServer -device CC2538SF53
```

```
root@IoT-Bnglr:~# JLinkGDBServer -device CC2538SF53
SEGGER J-Link GDB Server V4.98e Command Line Version

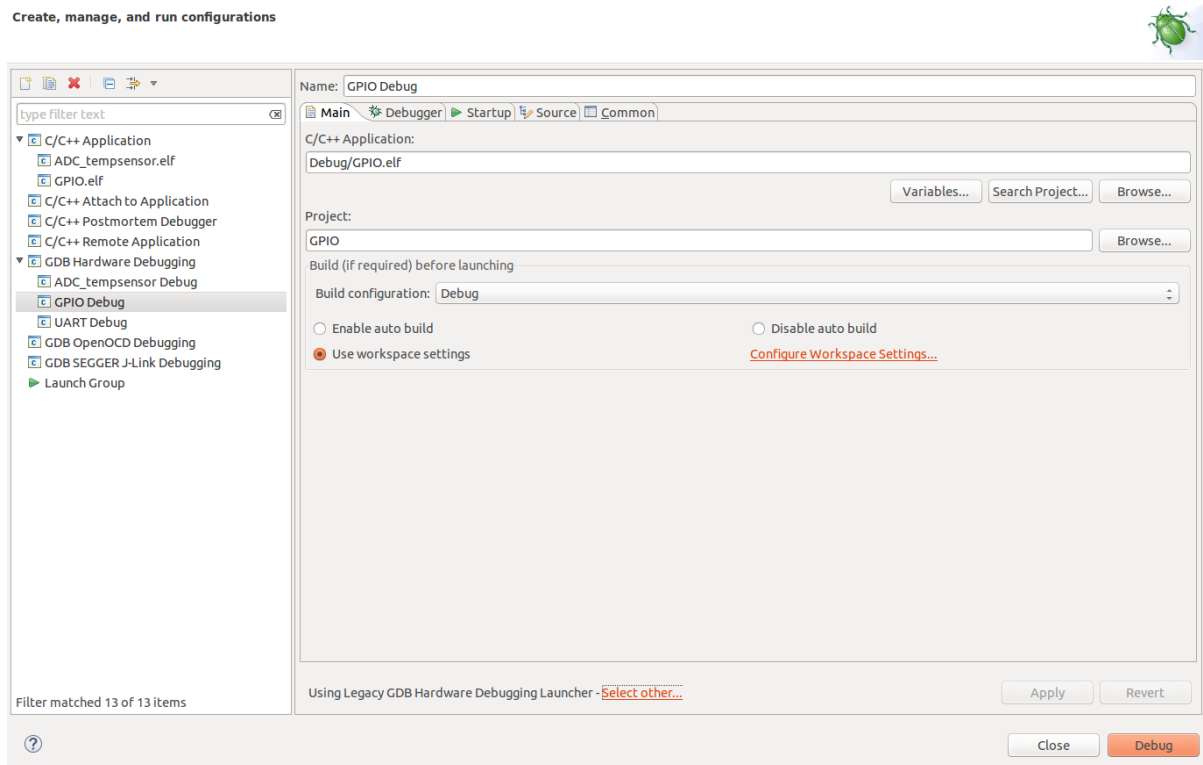
JLinkARM.dll V4.98e (DLL compiled May  5 2015 11:49:35)

-----GDB Server start settings-----
GDBInit file:          none
GDB Server Listening port: 2331
SWO raw output listening port: 2332
Terminal I/O port:     2333
Accept remote connection: yes
Generate logfile:       off
Verify download:        off
Init regs on start:     off
Silent mode:            off
Single run mode:        off
Target connection timeout: 0 ms
-----J-Link related settings-----
J-Link Host interface:  USB
J-Link script:          none
J-Link settings file:   none
-----Target related settings-----
Target device:          CC2538SF53
Target interface:       JTAG
Target interface speed: 1000kHz
Target endian:          little

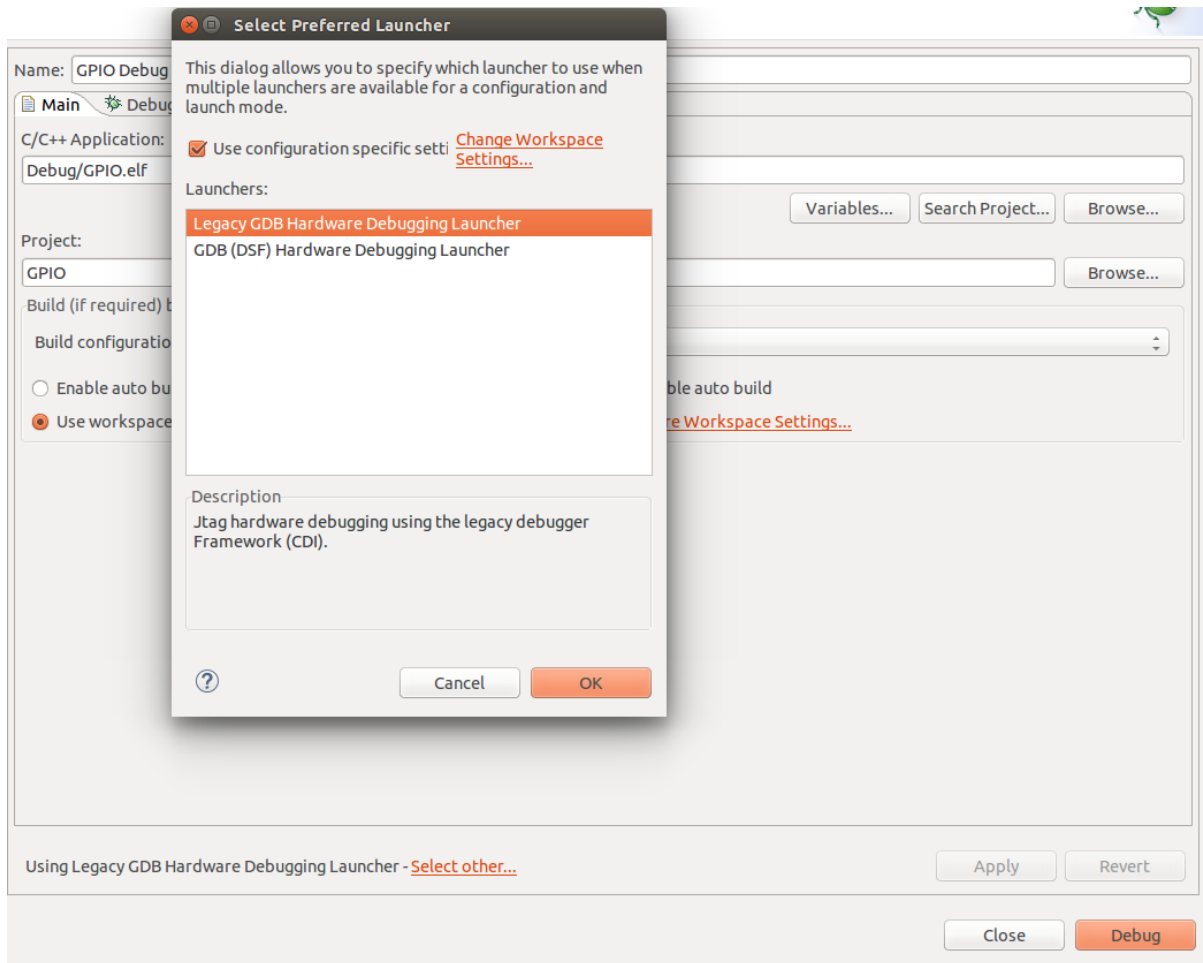
Connecting to J-Link...
J-Link is connected.
Firmware: J-Link ARM V8 compiled Nov 28 2014 13:44:46
Hardware: V8.00
S/N: 268005275
OEM: SEGGER-EDU
Feature(s): FlashBP, GDB
Checking target voltage...
Target voltage: 3.29 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 2 JTAG devices, Total IRLen = 10
JTAG ID: 0x4BA00477 (Cortex-M3)
Connected to target
Waiting for GDB connection...█
```

Now that GDB is running and we have the GDB server plug-in installed in Eclipse we can create a configuration to debug our CC2538 platform.

To do so, create a Debug Configuration as follows (click on the little arrow in the side of the little bug on the menu panel in eclipse):



Make sure to select the Legacy GDB Hardware Debugging Launcher (click select if this is not the case)



In the Debugger tab configure the debugger as follow (choose the location of the arm-none-eabi-gdb script)

Create, manage, and run configurations



Name: GPIO Debug

Main | **Debugger** | Startup | Source | Common

GDB Setup

GDB Command: Browse... Variables...

Command Set: Standard (Linux) ▾

Protocol Version: mi ▾

☐ Verbose console mode

Remote Target

☐ Use remote target

JTAG Device: GNU ARM J-Link ▾

Host name or IP address:

Port number:

Using Legacy GDB Hardware Debugging Launcher - [Select other...](#)

Apply Revert

Close Debug

Configure the start-up tab as follows:

Create, manage, and run configurations



Name: GPIO Debug

Main | Debugger | **Startup** | Source | Common

Initialization Commands

☒ Reset and Delay (seconds):

☒ Halt

target remote localhost:2331
monitor interface jtag
monitor speed 5000
monitor cadisable

Load Image and Symbols

☒ Load image

☒ Use project binary: GPIO.elf

☐ Use file: Workspace... File System...

Image offset (hex):

☒ Load symbols

☒ Use project binary: GPIO.elf

☐ Use file: Workspace... File System...

Symbols offset (hex):

Runtime Options

☐ Set program counter at (hex):

☒ Set breakpoint at:

☒ Resume

Run Commands

Using Legacy GDB Hardware Debugging Launcher - [Select other...](#)

Apply Revert

Close Debug

The code to put in the box is the following one.

```
target remote localhost:2331
monitor interface jtag
monitor speed 5000
monitor endian little
monitor flash download = 1
monitor flash breakpoints = 1
monitor reset
```

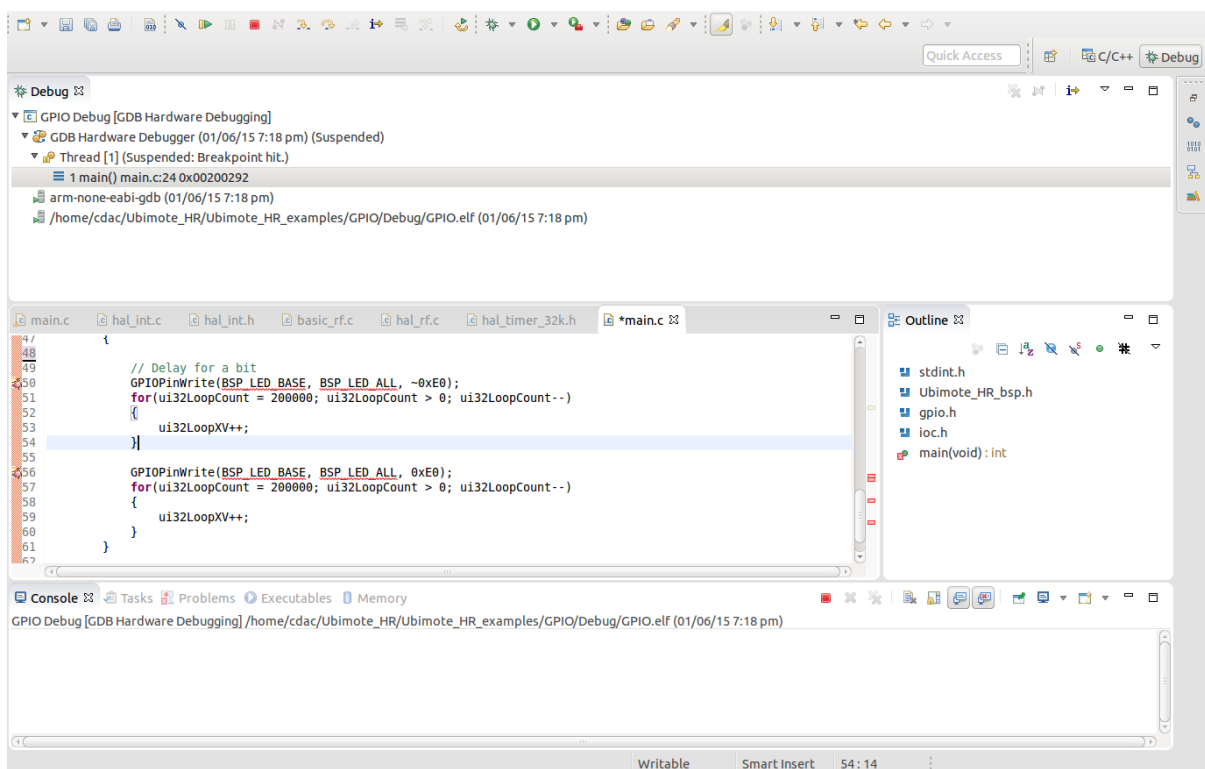
Click Apply to save the settings and Debug to start the debugging session.

9. Debugging using GDB

Finally, that everything is configured we can just launch our debug session. Make sure first that:

JTAG is connected to the board

Segger GDB Server app is running



10. Cloning Ubimote HR SDK

Clone Ubimote HR SDK into /opt folder,

```
$ cd /opt/
```

```
$ git clone
```

```
https://github.com/CDACBANG/Ubimote\_HR.git
```

11. Loading Projects into Eclipse

Open eclipse from the terminal as super user

```
$ sudo ./eclipse
```

Initially open the default workspace

Now, go to File -> Switch Workspace -> other

Choose /opt/Ubimote_HR/Ubimote_HR_examples

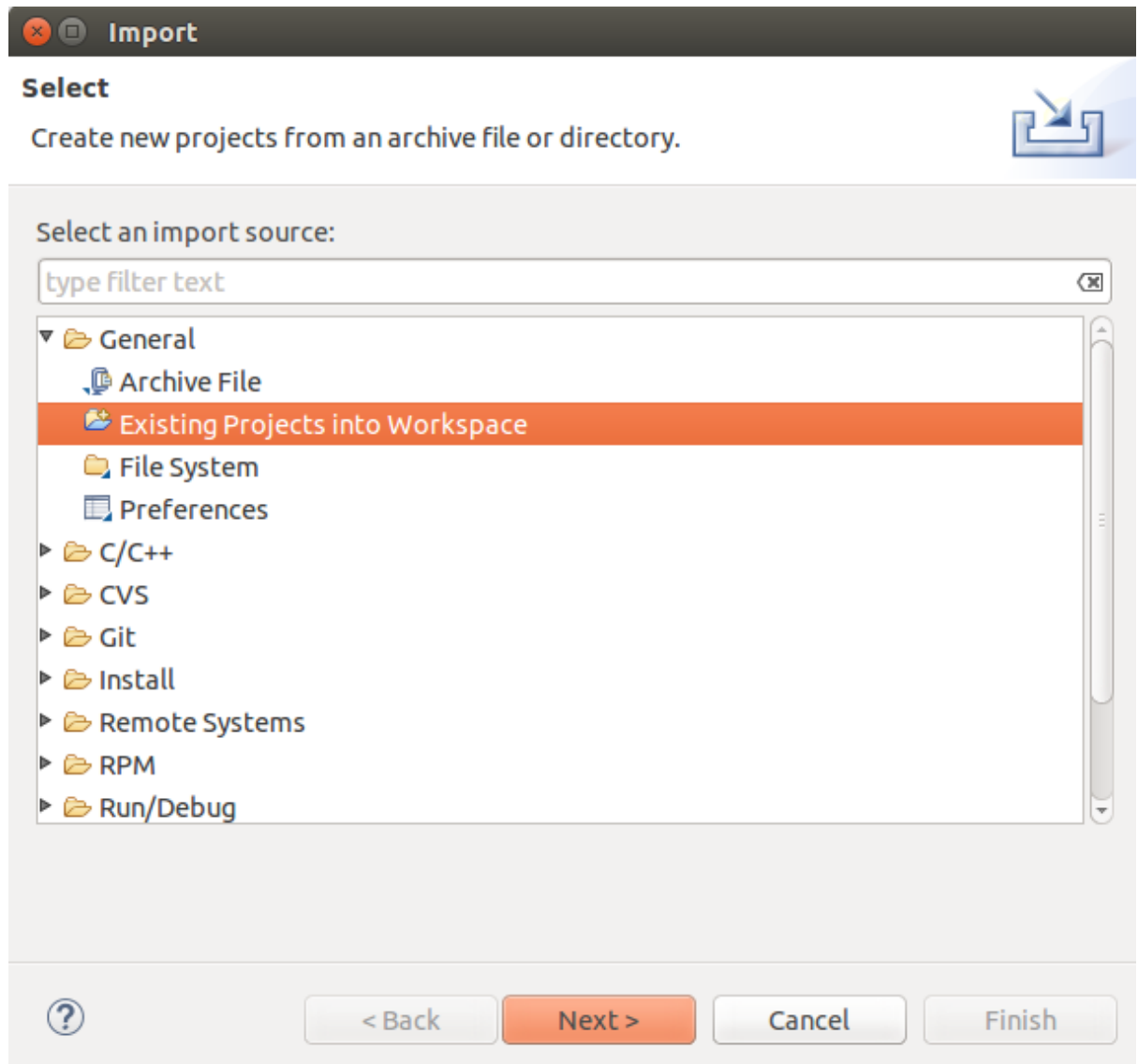
By default example projects will be loaded into the workspace, if the Projects are not imported into the workspace, import them into the eclipse workspace.

go to File -> import browse for the /opt/Ubimote_HR/

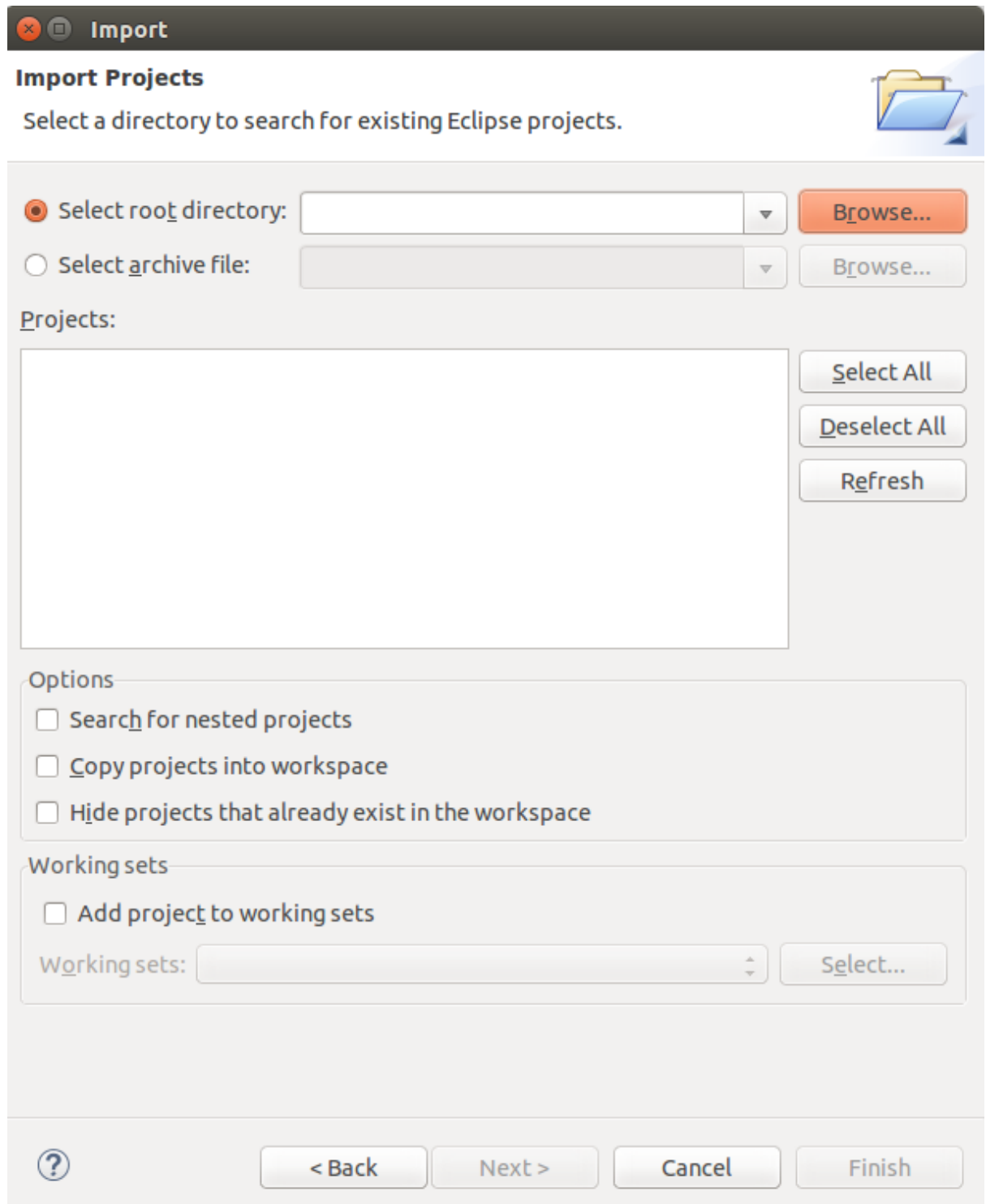
and select the Ubimote_HR_examples for the general examples

or

select the Ubimote_HR_UbiSense for the UbiSense examples



Note: If projects are already there in the workspace don't do anything



The screenshot shows the 'Import' dialog box in Eclipse. The title bar says 'Import'. Below the title bar, the text 'Import Projects' is displayed. A folder icon is shown to the right of the text. Below this, the instruction 'Select a directory to search for existing Eclipse projects.' is shown. There are two radio buttons: 'Select root directory:' (selected) and 'Select archive file:'. Each has a text field and a 'Browse...' button. Below these is a section labeled 'Projects:' with a large empty list box. To the right of the list box are three buttons: 'Select All', 'Deselect All', and 'Refresh'. Below the 'Projects:' section is a section labeled 'Options' with three checkboxes: 'Search for nested projects', 'Copy projects into workspace', and 'Hide projects that already exist in the workspace'. Below the 'Options' section is a section labeled 'Working sets' with a checkbox 'Add project to working sets'. Below this is a 'Working sets:' label, a text field, and a 'Select...' button. At the bottom of the dialog are four buttons: a help button (question mark), '< Back', 'Next >', 'Cancel', and 'Finish'.

Import

Import Projects

Select a directory to search for existing Eclipse projects.

☒ Select root directory: **Browse...**

☐ Select archive file: **Browse...**

Projects:

Options

☐ Search for nested projects

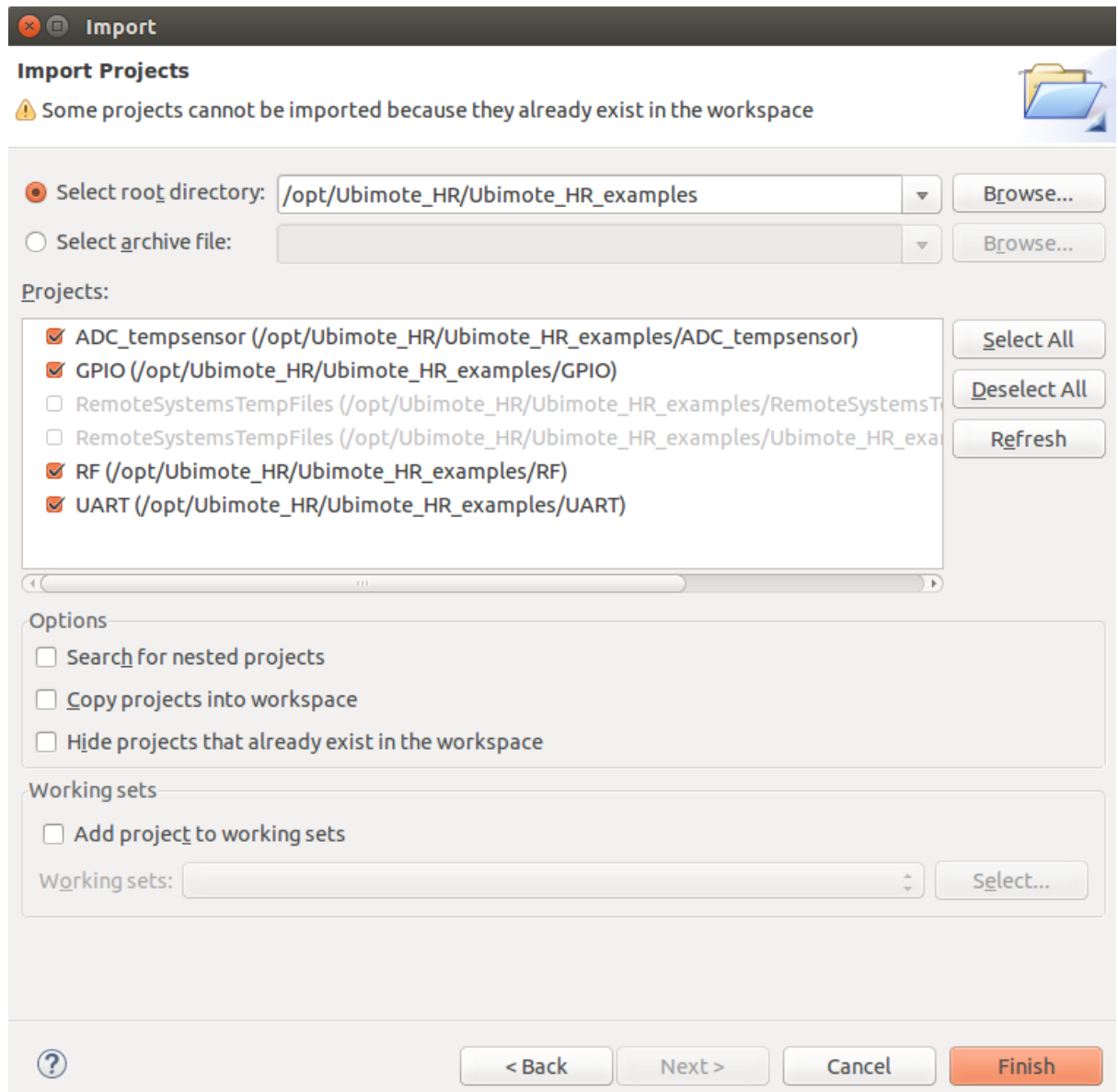
☐ Copy projects into workspace

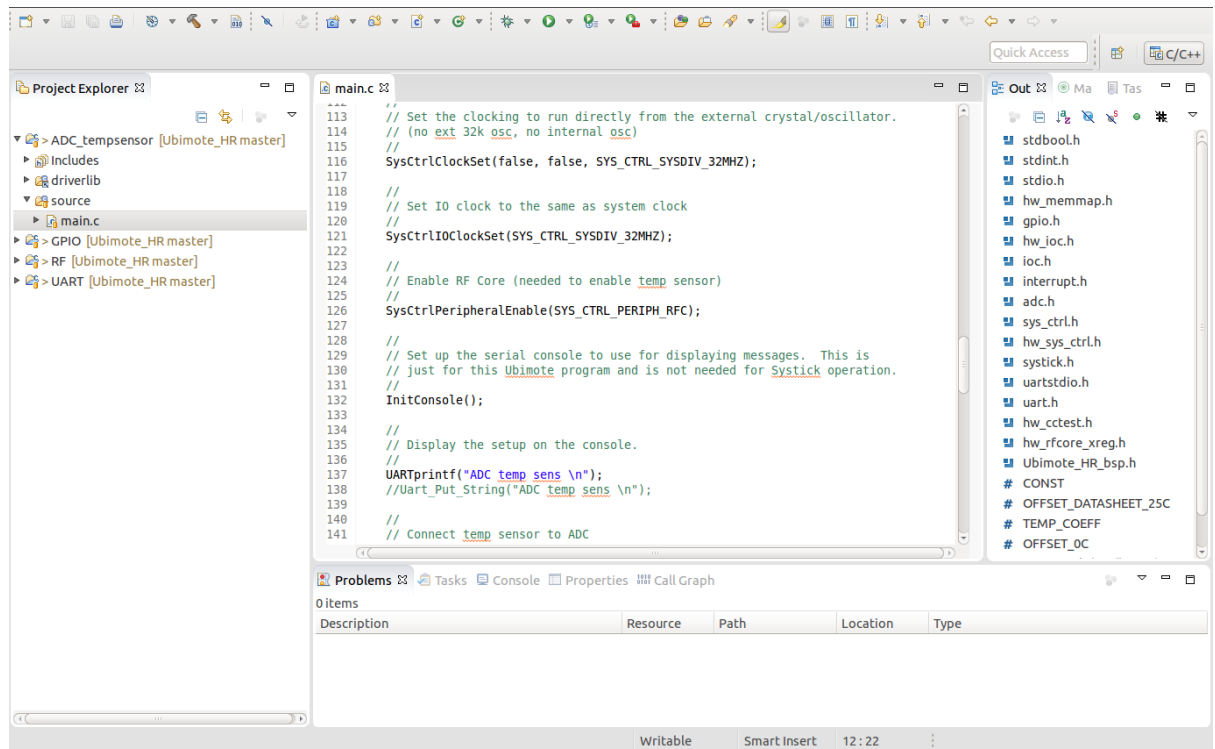
☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets: **Select...**





Now the projects will be added to the current workspace

Default Project settings has been set to the /opt/Ubimote_HR

If you downloaded Ubimote_HR examples to some other folder, update project settings as per that.

Once all the settings has been done, set the debug configuration as explained in section 8 and 9

Note: By default Debug configuration will be added, if not done set the configuration as explained in section 8 and 9.

Note: Right click on the Project and do Refresh once to resolve project links