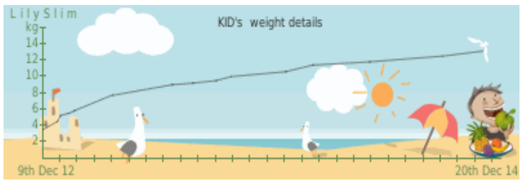


← Previous Post Next Post →

Cross-Compile and Remote Deploy from Windows for BeagleBone using Eclipse and a Linaro-gcc Toolchain.

ĐẶNG ANH TUẤN OCT 1, 2014 0

This article explains in detail how to set up a C/C++ Toolchain with a gcc Cross Compiler and the Eclipse IDE in Windows 7 and shows how both are used with a BeagleBone board.

There is a similar project from Michael Leonard available and also a video tutorial from Derek Molloy but instead of using a virtual machine with Linux this tutorial shows how to set up a toolchain on **Windows 7 "native"**. I assume Windows XP and Windows 8 could be used as well. (I just didn't check).

I couldn't find a good step-by-step introduction for this topic therefore I decided to contribute this page to the BeagleBone community. Besides that I'm pretty sure everything I show here is not related to the BeagleBone only and it would work on Raspberry Pi or other embedded Linux platforms in the same way.

Step 1: Set up the cross compiler

A good source for cross compiler toolchains is linaro.org . The one I'm using has to have the Linux support build in so I decided to download this file:

```
gcc-linaro-arm-linux-gnueabihf-4.8-2013.09_win32.zip
```

Just download and extract the zip file to your local harddrive. I recommend using a short path without spaces to ease up the build configuration later on. Therefore my choice is to put the cross compiler in my Harddrive's root directory like:

```
C:\gcc-linaro
```

Step 2: Set up Eclipse

In my opinon the strength of Eclipse is of course the flexible modular concept which enables the user to pick additions from a large variety of choices. The weakness is the complexity which is the price for the flexibility. For our project we're lucky since everything we need works "out of the box" if an actual version of Eclipse is used. So if you like to follow me go to www.eclipse.org/downloads and pick the most recent one under **"Eclipse IDE for C/C++ Developers"**. You might end up with the one I have:

```
eclipse-cpp-kepler-SR1-win32.zip
```

Similar to our gcc compiler we don't need an installation process for Eclipse. Again we just download and extract the zip file to your local harddrive. A good place for Eclipse is:

```
C:\Eclipse
```

Step 3: Set up GNU Make

With gnu make we face a little bit of a challenge. The GNU philosophy is to distribute everything as source code. Also the**make** tool (I'm not 100% sure if this **make**s sense or not). I had a binary set of binutils on

Search here..

Hỏi đáp nhanh

Trẻ dưới 5 tuổi có nên ăn trứng vịt lộn không?

Hỏi về bệnh lồng ruột ở trẻ

Cách giải quyết mâu thuẫn giữa mẹ và trẻ

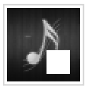
Mình muốn tham khảo các hoạt động Home School dành cho trẻ

Giáo dục sớm không đúng cách có tác hại gì không?

Có các mô hình giáo dục sớm nào?

Popular Comments Tags

Bộ cài ArcGIS Desktop 10.1 mới nhất, sạch nhất (Download từ trang chủ của esri)
Aug 5, 2013, 33 Comments on Bộ cài ArcGIS


Apologize – Timbaland Ft OneRepublic
Jun 6, 2013, 10 Comments on Apologize – Timbaland Ft OneRepublic

ArcGIS Desktop 10.2 Full crack
Oct 7, 2014, 7 Comments on ArcGIS Desktop 10.2 Full crack

Tuấn Phong trên Facebook

my harddrive from a while ago, but unfortunately the yagarto project is closed. Therefore I decided to make

```
yagarto-tools-20070303-setup.exe
```

available here for download.

A lazy dog like me who hates to tweak the **PATH** settings all the time just extracts everything from the .exe file and put the content of the **\bin** folder into

```
C:\Windows
```

In my opinion there is so much stuff which shouldn't be in the Windows folder that is doesn't hurt to have some useful stuff there as well. Besides **make** the yagarto-tools contain also **touch** and **rm** and other useful things which help to keep Makefiles very compatible between Windows and Linux.

Note*: I personally like to work with* ***make*** *and a manually edited* ***Makefile****. This gives me the good feeling that I am "in control of things". Eclipse has a build-in make which I avoid because I switch editors very often. If you want to use the Eclipse make process please stop reading here and go back to Google.*

Step 4: Our HelloBone sample project

The hopefully last download for today is the HelloBone.zip file. Download and extract the content anywhere you want. I stored my project here:

```
C:\work\hellobone
```

Step 5: Check if you can compile without Eclipse

At first you shoud check if **make** is really visible from everywhere. Open a command shell and go to your working directory where **Makefile** resides and type **make -v**

```
c:\work\hellobone>make -v
GNU Make 3.81 Copyright (C) 2006  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.

This program built for i686-pc-mingw32
```

If your result is similar to what you see above we're good to go. Try **make** without argument and you shoud see:

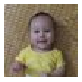
```
c:\work\hellobone>make
.
---COMPILE---  c:/work/hellobone/source/hellobone.c
"C:\gcc-linaro\bin\arm-linux-gnueabihf-gcc.exe" -c -o c:/work/hellobone/object/hellobone.o c:/
ellobone/source/hellobone.c -marm -O0  -g  -Ic:/work/hellobone/include
.
---COMPILE---  c:/work/hellobone/source/tools.c
"C:\gcc-linaro\bin\arm-linux-gnueabihf-gcc.exe" -c -o c:/work/hellobone/object/tools.o c:/work
bone/source/tools.c -marm -O0  -g  -Ic:/work/hellobone/include
.
---LINK---
"C:\gcc-linaro\bin\arm-linux-gnueabihf-gcc.exe" -o hellobone c:/work/hellobone/object/hellobon
/work/hellobone/object/tools.o -marm -O0  -g  -Ic:/work/hellobone/include
.
---SUCCESS---  hellobone

c:\work\hellobone>
```

In case you see the **—SUCCESS—** message you're done and we can continue with Eclipse. In case you don't please feel free to jump back to Step 1 and try again.

Note: *If you check the* ***Makefile*** *you'll find that I did it a little more complicated compared to what's really necessary. First we compile two source files with two header files (like in every good example they do almost nothing). Secondly they are in different directories. The object files are placed in a seperate directory as well. But if you are a beginner and you use this structure as a seed for your future projects you will love me for that later on. Handling only a few more source files and having them together with object and include in just one directory starts hurting quickly if the project is growing.*

Step 6: Start Eclipse

My recommendation is to double click on



Tuấn Phong

Like

67 people like Tuấn Phong.

Facebook social plugin

Lịch

June 2015

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					
« May						

Meta

Log in

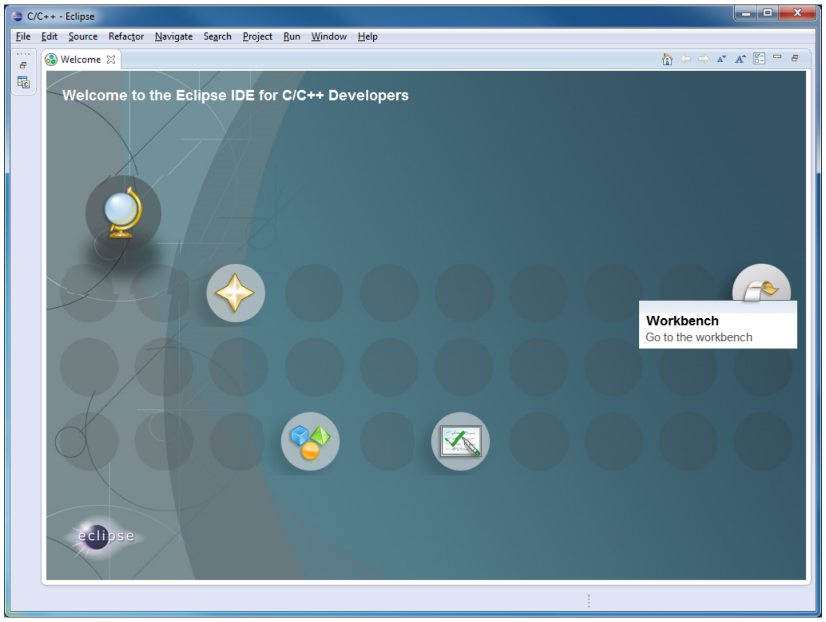
Entries RSS

Comments RSS

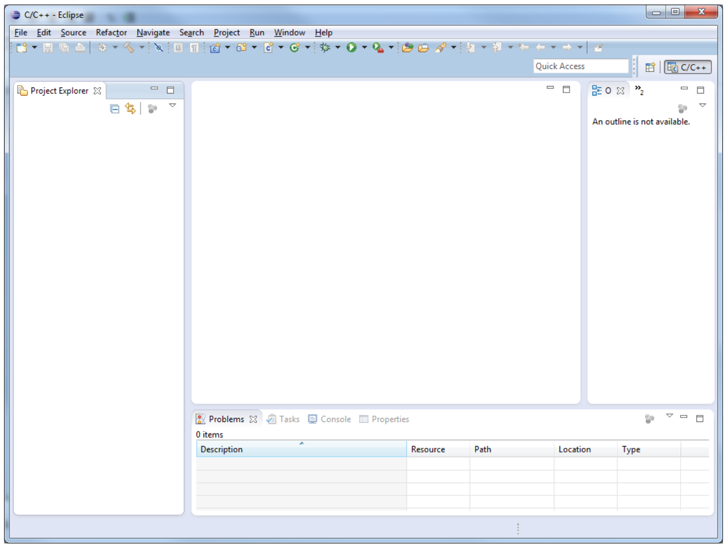
WordPress.org

```
C:\Eclipse\eclipse.exe
```

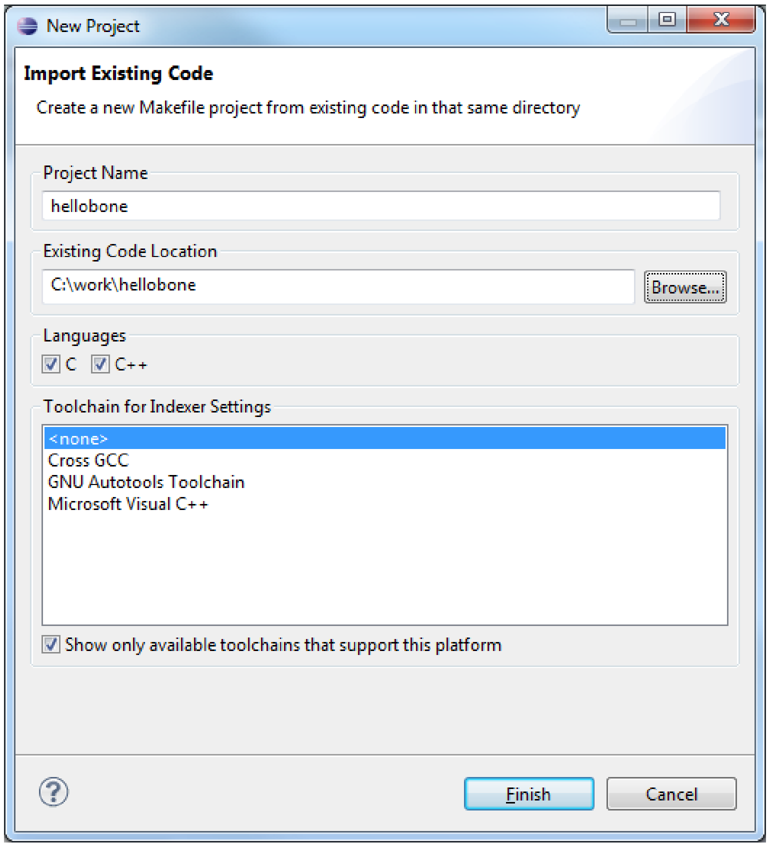
The result should look like the screenshot below. If not it might be neccessary to install or update JAVA at this point in case your machine isn't ready for Eclipse yet.


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Welcome to the Eclipse IDE for C/C++ Developers
Workbench
Go to the workbench
eclipse

Click on the Workbench symbol and a virgin IDE should welcome you:



Since we don't want to use the internal make functions of Eclipse the next step is to tell Eclipse where our project resides. Open **File – New – Makefile Project with existing code** and navigate to our project folder. Use as the toolchain setting for now. Feel free to go deeper into the Eclipses possibilities later when our little lesson here is over.



New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: hellobone

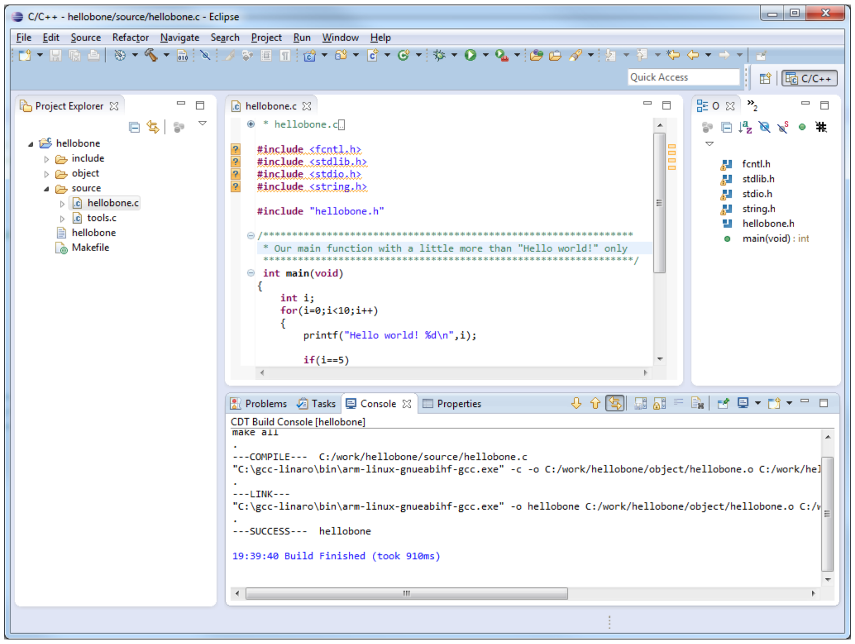
Existing Code Location: C:\work\hellobone

Languages: [✓ C] [✓ C++]

Toolchain for Indexer Settings:
- [✓ <none>]
- Cross GCC
- GNU Autotools Toolchain
- Microsoft Visual C++

[✓] Show only available toolchains that support this platform

Afterwards the Eclipse IDE should (almost) look like this if you did what I did and opened the **hellobone.c** file by navigating through the **Project Explorer.** You will already notice why I put certain files in different directories: Eclipse uses the directories for project navigation automatically.



```
 * hellobone.c

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

#include "hellobone.h"

/*****************************************************************
 * Our main function with a little more than "Hello world!" only
 *****************************************************************/
 int main(void)
{
    int i;
    for(i=0;i<10;i++)
    {
        printf("Hello world! %d\n",i);

        if(i==5)
```

CDT Build Console [hellobone]

```
make all
.
---COMPILE---  C:/work/hellobone/source/hellobone.c
"C:\gcc-linaro\bin\arm-linux-gnueabihf-gcc.exe" -c -o C:/work/hellobone/object/hellobone.o C:/work/hel
.
---LINK---
"C:\gcc-linaro\bin\arm-linux-gnueabihf-gcc.exe" -o hellobone C:/work/hellobone/object/hellobone.o C:/w
.
---SUCCESS---  hellobone

19:39:40 Build Finished (took 910ms)
```

Now if you hit **Project – Build all** Eclipse should be able to execute the same make process we used in Step 5. You should see the output of our make process in the **Console** window. With **Project – Clean** you should also be able to clean our project as described in the **Makefile**.

Note: If you follow **Project – Properties – C/C++ General – Preprocessor Include path** and add

```
C:\gcc-linaro\arm-linux-gnueabihf\libc\usr\include
```

as a **File System Path** under **CDT User Setting Entries** the Eclipse Code parser knows where our **#include** files are and the nasty question mark symbols will disappear. Better for the eyes. But compilation works also without it if you don’t care about your eyes too much.

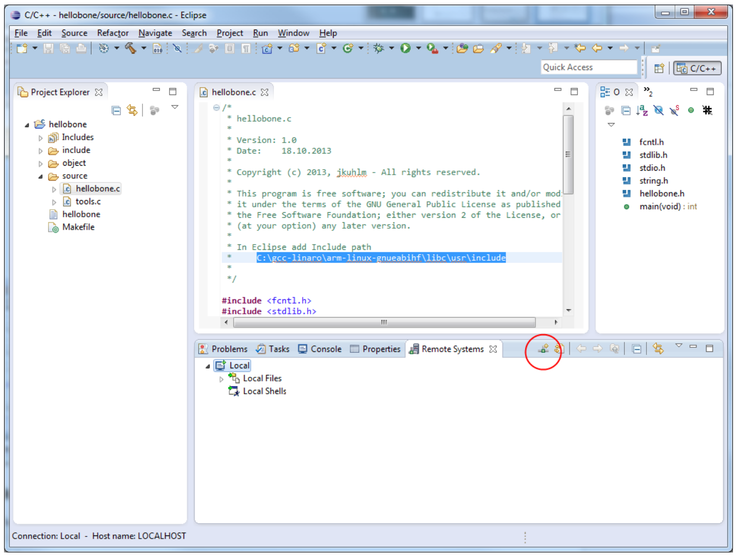
So we’re done cross compiling our project. That means we’re ready for the next step:

Step 7: Remote deploying to the BeagleBone

I use two different BeagleBones: The **whitebone** one runs Angstrom and the **blackbone** one Debian. Our project here is running on both. Some minor changes need to be made in the Makefile for Angstrom (see the CFLAGS section in the **Makefile**). For session here I’m using the **blackbone** and my setup is:

```
host: blackbone
ip: 192.168.1.15
sshd: running
user: root
password: ********
```

Eclipse already knows that people like us want to put stuff on toys like the BeagleBone. Therefore Eclipse is equipped with modules like **Window – Show View – Other – Remote Systems – Remote Systems**. Follow me on this one and you should be able to see what I see:



```
/*
 * hellobone.c
 *
 * Version: 1.0
 * Date:    18.10.2013
 *
 * Copyright (c) 2013, jkuhlm - All rights reserved.
 *
 * This program is free software; you can redistribute it and/or mod
 * it under the terms of the GNU General Public License as published
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * In Eclipse add Include path
 *      C:\gcc-linaro\arm-linux-gnueabihf\libc\usr\include
 *
 */

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

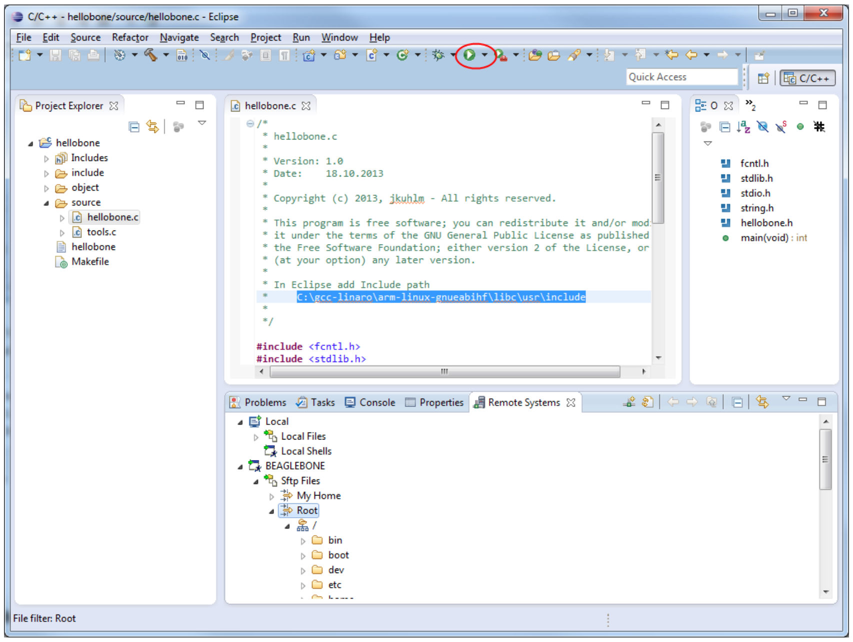
If you click where the little **Define a connection to remote system** button is (red circle) a dialog appears where you can select **SSH only** first and hack in your BeagleBone personality afterwards. In my case:

```
Hostname: 192.168.1.15
Connection name: BEAGLEBONE
Description: BEAGLEBONE
```

If you double click on **Sftp Files** a dialog for user name and password appears like expected. I recommend to check the**Save User Id** and **Save Password** checkbox since we might need to login more frequently in the near future.

Note: *BE CAREFUL NOT TO SCREW UP HERE. I did it once and it took me a while to find out where Eclipse stores this kind of information.*

Of course you might also see some **SSH key acceptance dialog** which I recommend to accept if you trust your BeagleBone.

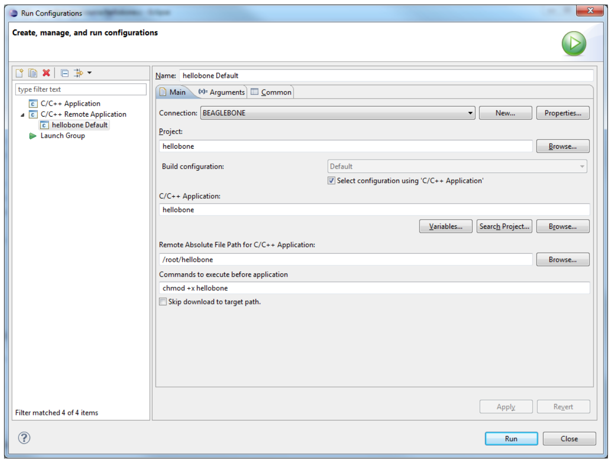


```
/*
 * hellobone.c
 *
 * Version: 1.0
 * Date:    18.10.2013
 *
 * Copyright (c) 2013, jkuhlm - All rights reserved.
 *
 * This program is free software; you can redistribute it and/or mod
 * it under the terms of the GNU General Public License as published
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * In Eclipse add Include path
 *     C:\gcc-linaro\arm-linux-gnueabihf\libc\usr\include
 *
 */

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

Now we can browse through the BeagleBone file system. Isn’t that nice? (A feature which a regular Windows Explorer user might never see all his live).

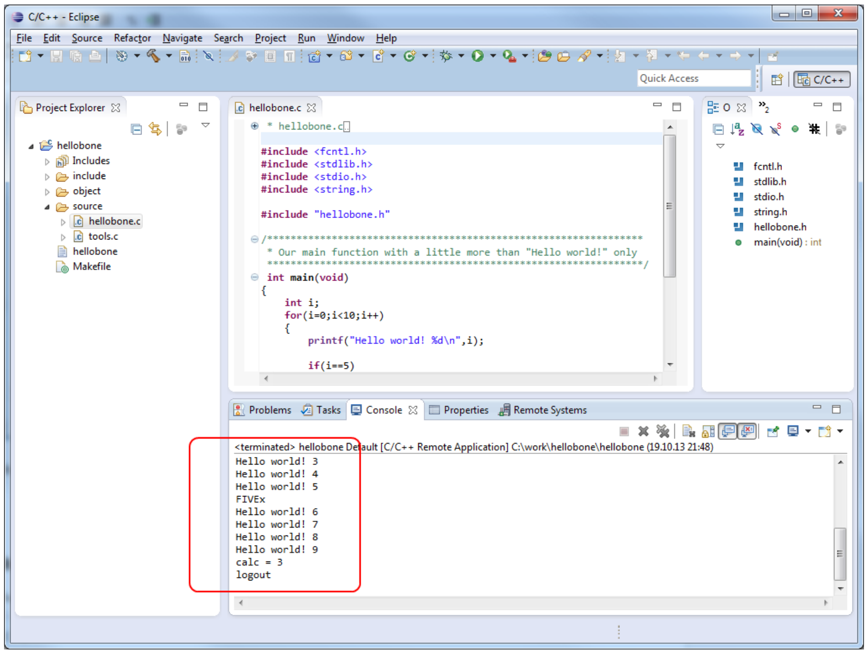
So our connection is there which means we are ready for deployment. Open **Run configurations…** in the pull down menu next to the run button (this time the red circle in the screenshot above became an oval by accident) and add a new**Run configuration** under **C/C++ Remote Application**. Eclipse names it **hellobone Default** automatically.

Select your Connection **BEAGLEBONE** (or however you named it) and make sure you fill everything else out like I did:



Name: hellobone Default
Connection: BEAGLEBONE
Project: hellobone
Build configuration: Default
[✓] Select configuration using 'C/C++ Application'
C/C++ Application: hellobone
Remote Absolute File Path for C/C++ Application: /root/hellobone
Commands to execute before application: chmod +x hellobone
[ ] Skip download to target path.

After saving our settings with **Apply** we can **Close** this dialog and hit **hellobone Default** in the Pull Down menu of the**Run Configurations** button (red oval, remember?). If everything goes well our binary gets sent to the BeagleBone and is executed afterwards. In the screenshot below you might notice the printf output of our little software in the**Console** window which tells us that we’re running.



```
 * hellobone.c

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

#include "hellobone.h"

/***************************************************************
 * Our main function with a little more than "Hello world!" only
 ***************************************************************/
int main(void)
{
    int i;
    for(i=0;i<10;i++)
    {
        printf("Hello world! %d\n",i);

        if(i==5)
```

```
<terminated> hellobone Default [C/C++ Remote Application] C:\work\hellobone\hellobone (19.10.13 21:48)
Hello world! 3
Hello world! 4
Hello world! 5
FIVEx
Hello world! 6
Hello world! 7
Hello world! 8
Hello world! 9
calc = 3
logout
```

Now is this something or what? But hold on: It's getting even better in

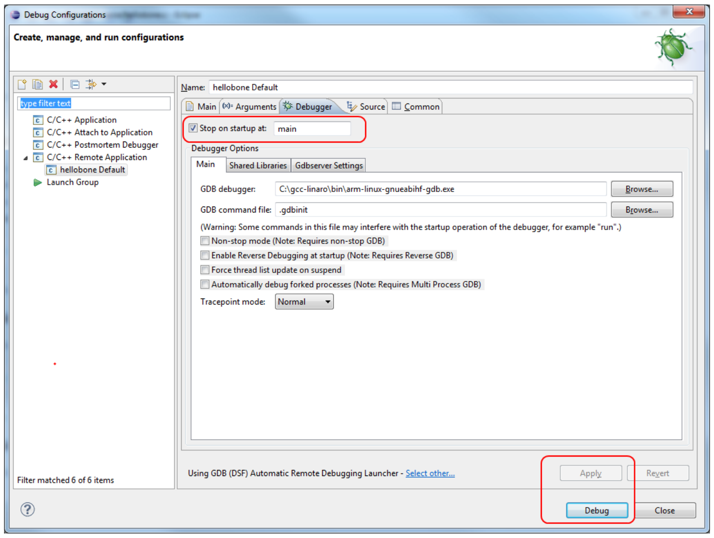
Step 8: Remote Debugging on the BeagleBone

In the same way we opened our **hellobone Default** in the **Run Configurations…** we open now the **Debug Configurations…** which is in the pull down menue next to the sweet little green bug symbol. You will notice that the settings from **hellobone Default** are already prepopulated.

Change to the tab **Debugger** and browse to the **gdb** binary of our linaro toolchain to select the **GDB debugger** you want to use. (We are very lucky that we installed one in Step 1 by accident)

```
C:\gcc-linaro\bin\arm-linux-gnueabihf-gdb.exe
```

The **GDB command file** you can leave untouched or empty it. This doesn't make a difference for now. Your settings should look like the screen below. Believe it or not that's all.



Name: hellobone Default

[✓] Stop on startup at: main

GDB debugger: C:\gcc-linaro\bin\arm-linux-gnueabihf-gdb.exe

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

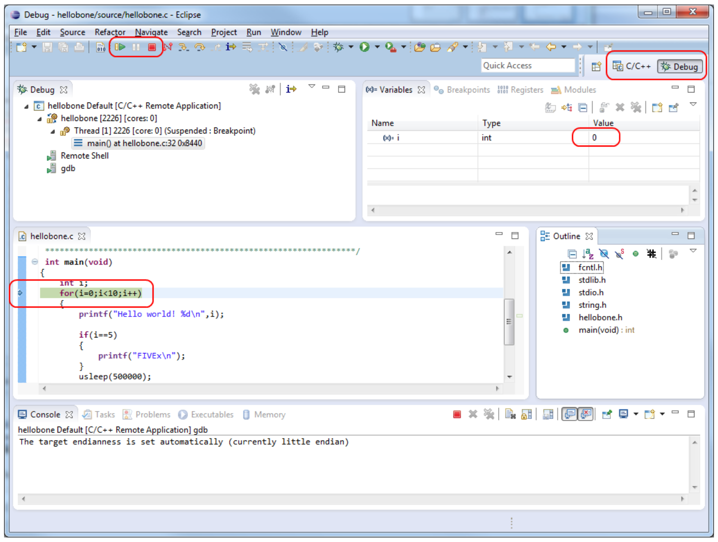
[ ] Automatically debug forked processes (Note: Requires Multi Process GDB)

Tracepoint mode: Normal

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

Filter matched 6 of 6 items

You might have noticed that we tell the debugger to stop at main if we leave the check box as is. After using the buttons**Apply** and **Debug** (or **Close** and pull down **bonehello Default** next to the sweet little green animal) we should be ready to rumble:



```
********************************************************************/
int main(void)
{
    int i;
    for(i=0;i<10;i++)
    {
        printf("Hello world! %d\n",i);

        if(i==5)
        {
            printf("FIVEx\n");
        }
        usleep(500000);
```

```
hellobone Default [C/C++ Remote Application] gdb
The target endianness is set automatically (currently little endian)
```

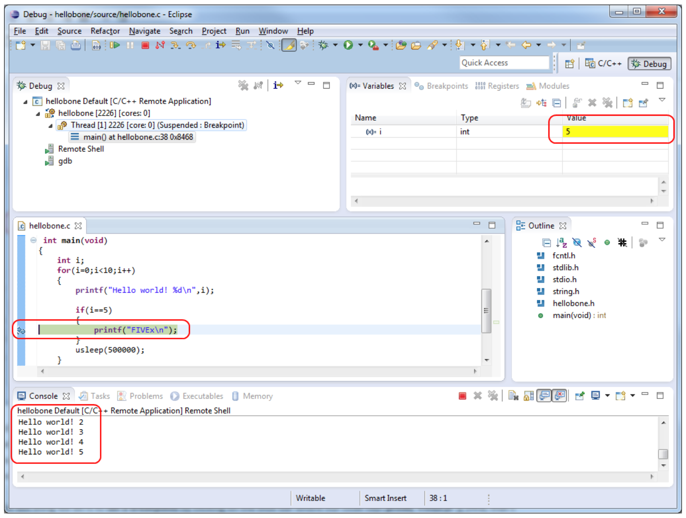
Eclipse changes the perspective from **C/C++** to **Debug** and gives you notice about that. You can always jump forth and back between these perspectives by using the buttons in the upper right corner.

You can **Start (Resume)** and **Stop (Terminate)** your program with the symbols in the toolbar. You also might have noticed that the debugger stoped as promised at **main()** and you see that a variable from the actual context **"i"** is shown as well.

The last thing we do is to **set a breakpoint** by clicking on the blue bar on the left hand side where our code says**printf("FIVEx\n");** (Aha, that's the reason why his sample programm doesn't just print "Hello world!" once).

By the way breakpoints can be set in each perspective, which means also in the C/C++ coding view.

Now hit **Start (Resume)** and see what happens:



```
int main(void)
{
    int i;
    for(i=0;i<10;i++)
    {
        printf("Hello world! %d\n",i);

        if(i==5)
        {
            printf("FIVEx\n");
        }
        usleep(500000);
    }
```

```
hellobone Default [C/C++ Remote Application] Remote Shell
Hello world! 2
Hello world! 3
Hello world! 4
Hello world! 5
```

The execution stops at our **breakpoint**. The console output also. The variable **"i"** has changed to the value 5. Eclipse helps us by highlighting this in yellow.

Is this cool or what? We're done! Enjoy the moment! Go and grab a beer!

Here ends our little journey. I hope you enjoyed it. Feel free to leave a comment if something is unclear and should be improved or if you have any other remarks after reading this.

Source: http://jkuhlm.bplaced.net/hellobone/

TAGS » BeagleBone, IT, Tutorial, Windows **POSTED IN »** IT

About the author: Đặng Anh Tuấn

View all posts by Đặng Anh Tuấn

Twitter - Facebook

Related »

Cài đặt song song GCC 4.4. và GCC 4.8 cho Ubuntu

Hướng dẫn cài đặt Entrust Package cho Laravel 4

Hướng dẫn cài đặt Confide Package cho Laravel 4

Khắc phục lỗi Port 80 in use by "Unable to open process" with PID 4 trên XAMPP

Leave A Response »

Name (required)

Email (required)

Website

Comment

Post Comment

more...

Thống kê

1 0 5 9 3 8

Visit Today : 19
Total Visit : 105938
Hits Today : 36
Total Hits : 363954
Who's Online : 3

Your IP Address: 75.138.19.213

Archives

Archives Select Month ▼

Polls

Đánh giá Website

- Rất đẹp
- Đẹp
- Bình thường
- Xấu

Vote

View Results

Polls Archive