

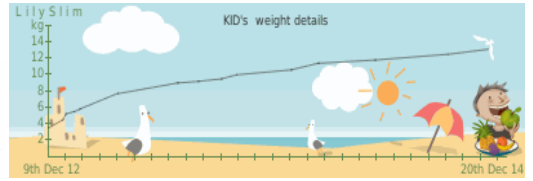
KID

T P W T P  
B KID

Lilypie



KID is 2 years old. Happy Birthday!



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 settings all the time just extracts everything from the .exe file and put the content of the 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 the yagarto-tools contain also and 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.

## S 4:0 H B

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\helloworld

## S 5:0 E

At first you should check if is really visible from everywhere. Open a command shell and go to your working directory where resides and type

```
c:\work\helloworld>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 without argument and you should see:


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

In case you see the 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.


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 separate 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.

## S 6:0 E

My recommendation is to double click on



67 people like Tuấn Phong.



Facebook social plugin

## L

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						

## M

Log in

-----

Entries RSS

-----

Comments RSS

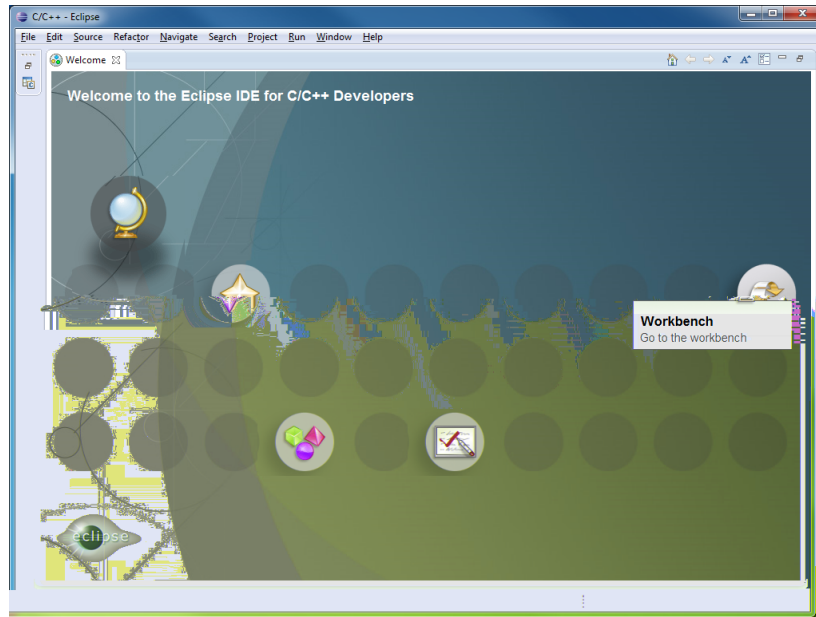
-----

WordPress.org

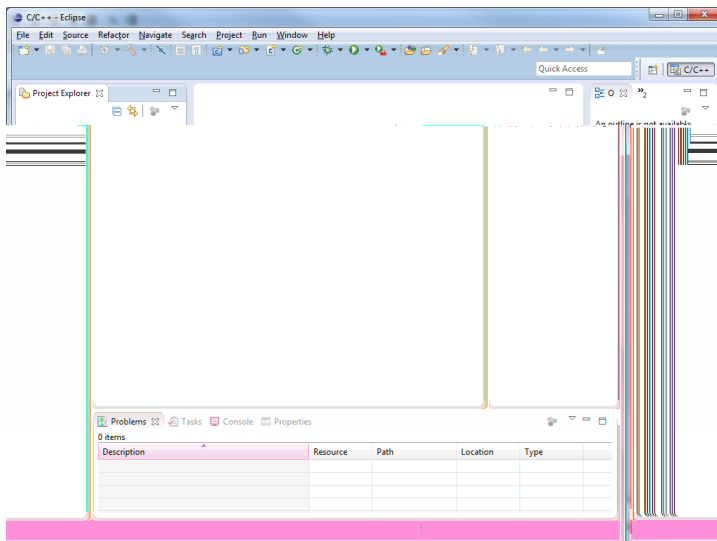
-----

C:\Eclipse\eclipse.exe

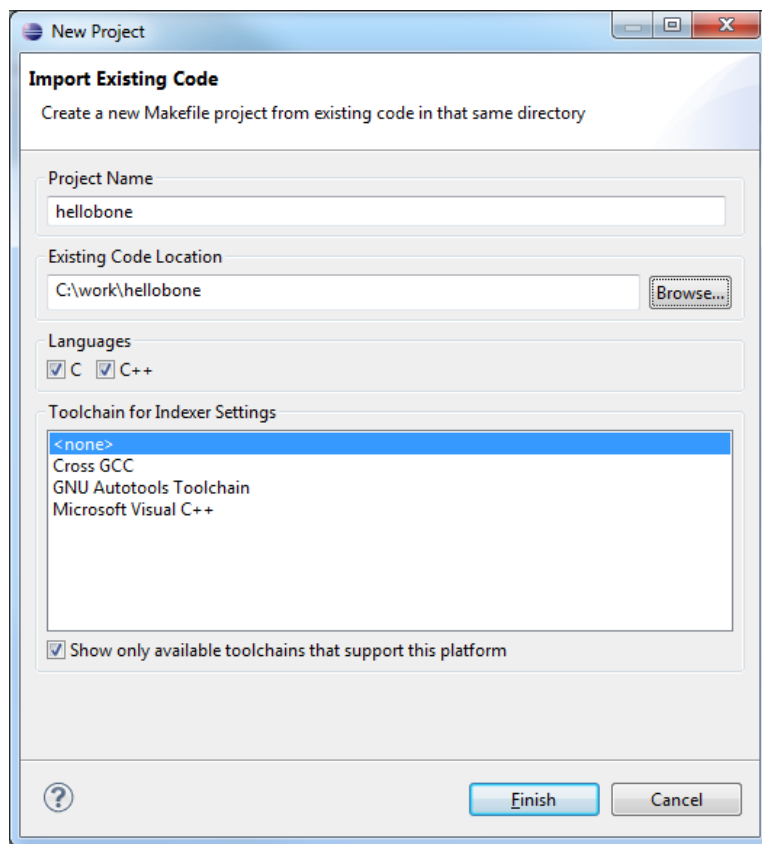
The result should look like the screenshot below. If not it might be necessary to install or update JAVA at this point in case your machine isn't ready for Eclipse yet.



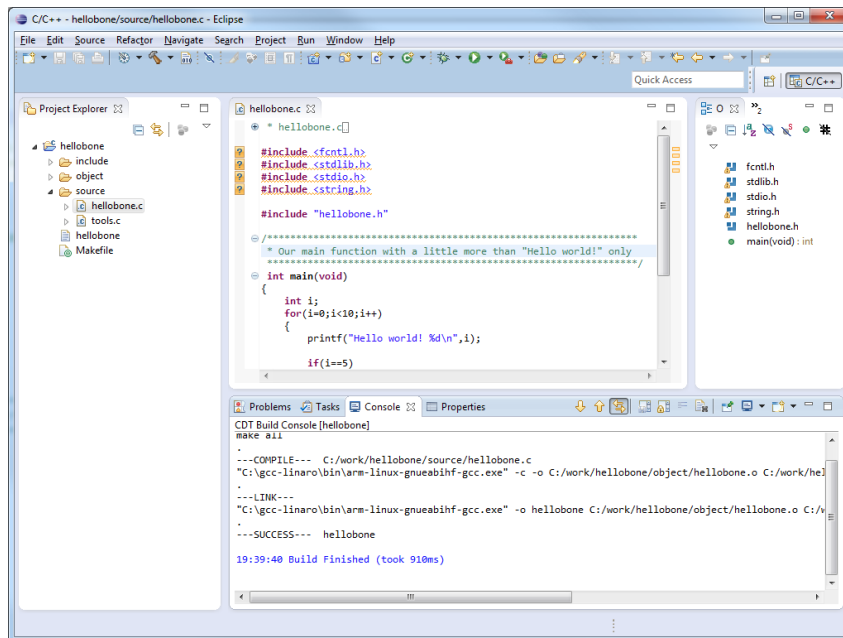
Click on the Workbench symbol and a virgin IDE should welcome you:



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 > Project...` and navigate to our project folder. Use `g++` as the toolchain setting for now. Feel free to go deeper into the Eclipse's possibilities later when our little lesson here is over.



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



Now if you hit the 'Run' button, 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 the 'Run' button you should also be able to clean our project as described in the 'Run' dialog.

Note: If you follow

compilation works also without it if you don't care about your eyes too much.

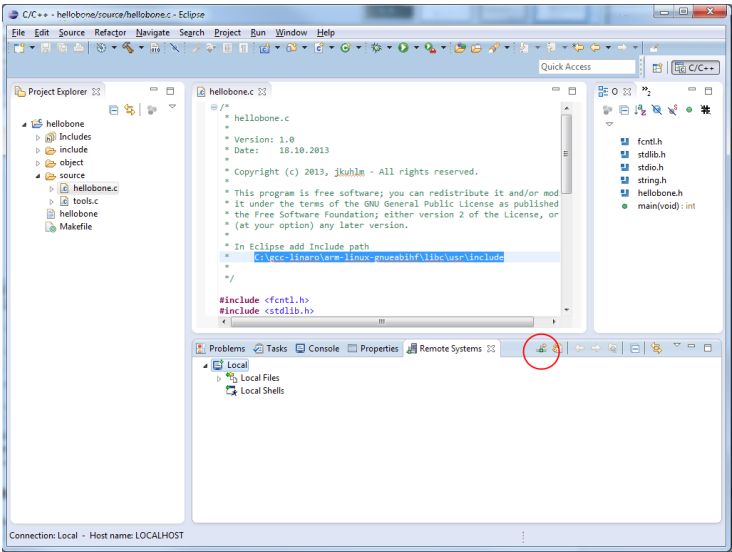
So we're done cross compiling our project. That means we're ready for the next step:

S 7:R B B

I use two different BeagleBones: The s 5 one runs Angstrom and the 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 ). For session here I'm using the and my setup is:

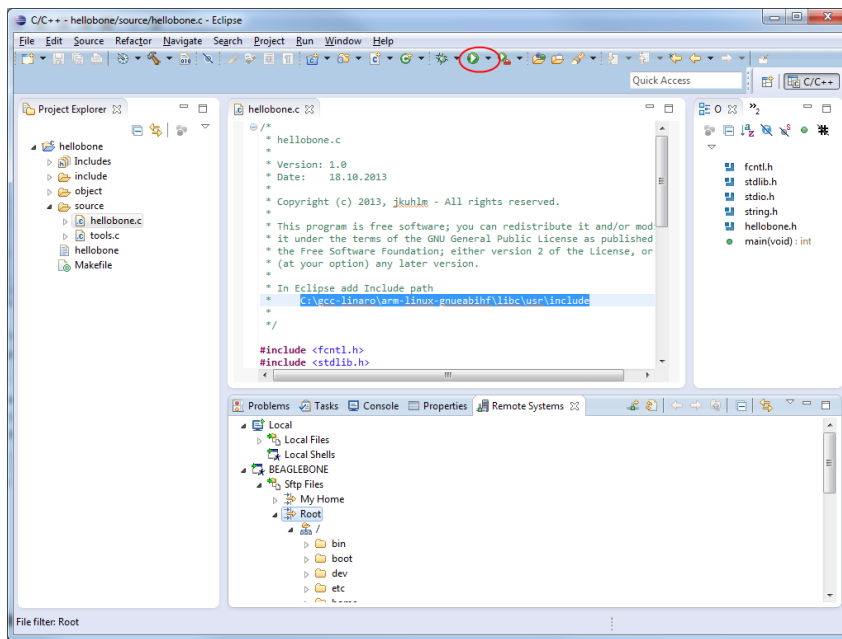
```
host: 192.168.1.15
ip: 192.168.1.15
sshd: running
user: root
password: *****
```

Eclipse already knows that people like us want to put stuff on toys like the BeagleBone. Therefore Eclipse is equipped with modules like R S . Follow me on this one and you should be able to see what I see:



If you click where the little button is (red circle) a dialog appears where you can select first and hack in your BeagleBone personality afterwards. In my case:

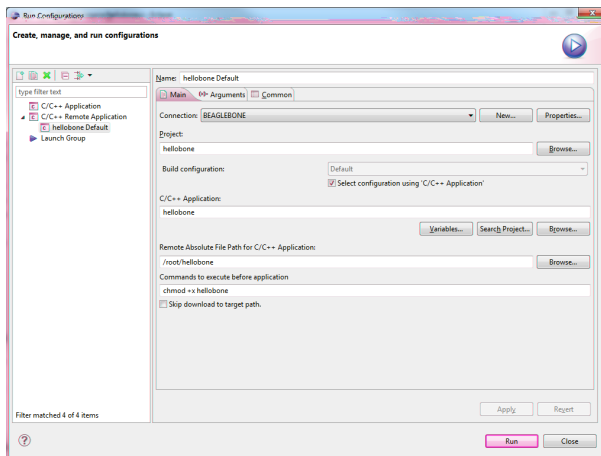
Hostname: 192



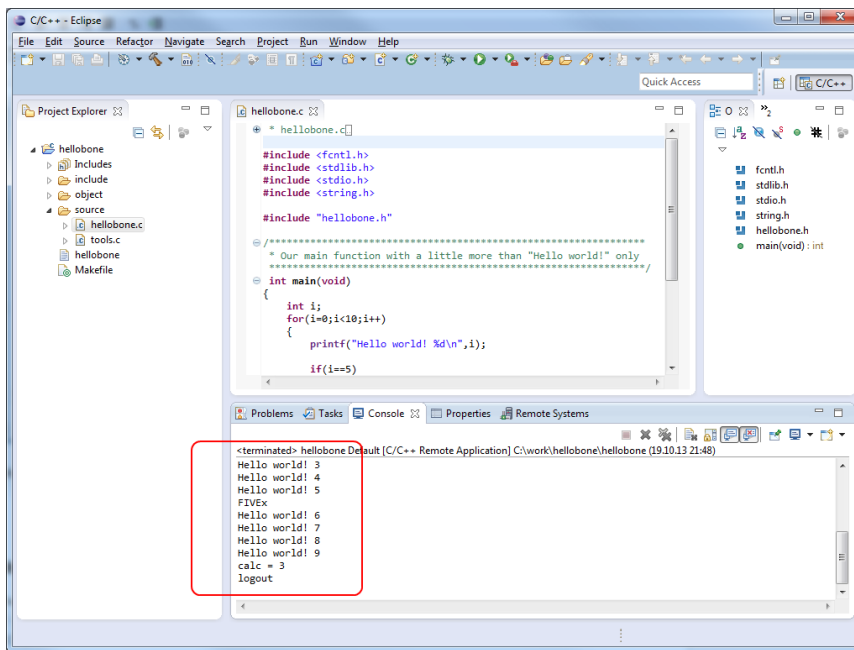
Now we can browse through the BeagleBone file system. Isn't that nice? (A feature which a regular Windows Explorer user might never see all his live).

So our connection is there which means we are ready for deployment. Open  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  under . Eclipse names it  automatically.

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



After saving our settings with  we can  this dialog and hit  in the Pull Down menu of the  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  window which tells us that we're running.



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

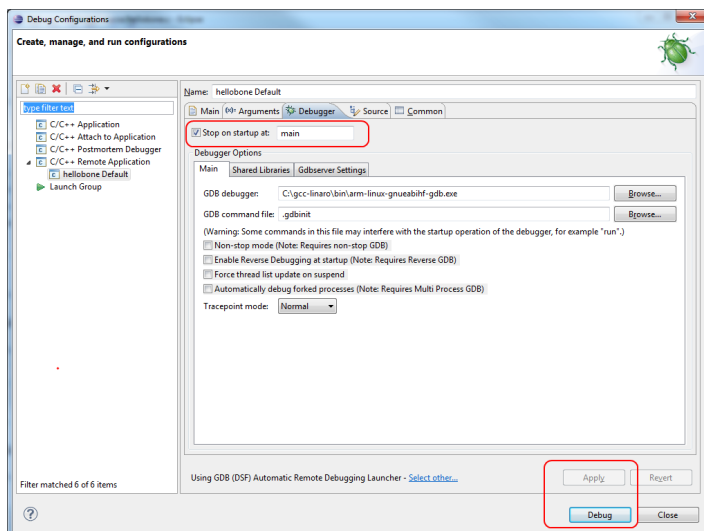
## S 8: R D B B

In the same way we opened our  in the  we open now the  which is in the pull down menu next to the sweet little green bug symbol. You will notice that the settings from  are already prepopulated.

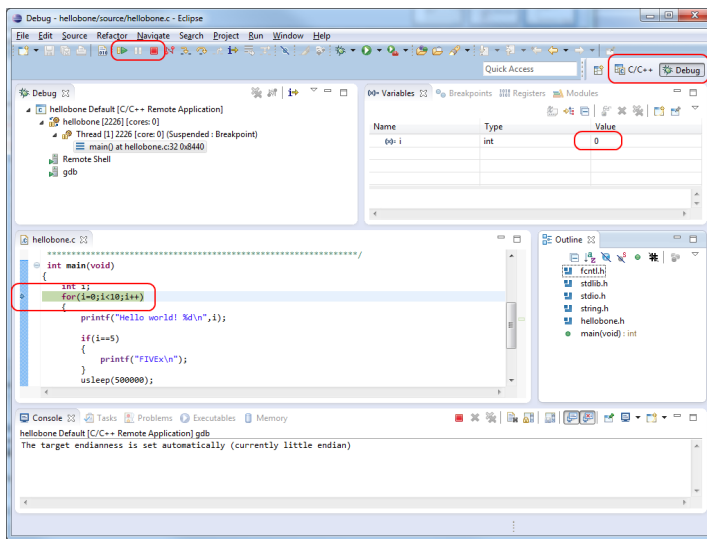
Change to the tab  and browse to the  binary of our linaro toolchain to select the  you want to use. (We are very lucky that we installed one in Step 1 by accident)

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

The  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.



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  and  (or  and pull down  next to the sweet little green animal) we should be ready to rumble:



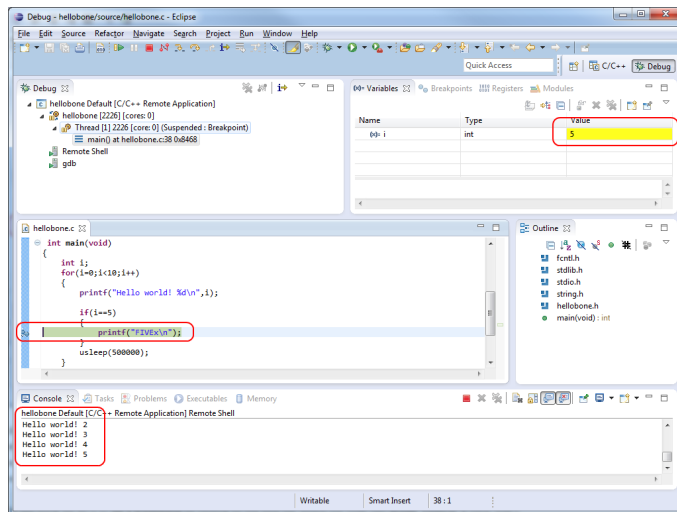
Eclipse changes the perspective from `Code` to `Debug` and gives you notice about that. You can always jump forth and back between these perspectives by using the buttons in the upper right corner.

You can `step over` and `step into` your program with the symbols in the toolbar. You also might have noticed that the debugger stopped as promised at `breakpoint` and you see that a variable from the actual context `main` is shown as well.

The last thing we do is to `step over` by clicking on the blue bar on the left hand side where our code says `if(i==5)` (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.

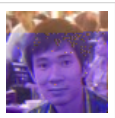
Now hit `F5` and see what happens:



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.

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.





About the author: Đặng Anh Tuấn

[View all posts by Đặng Anh Tuấn](#)

[Twitter](#) - [Facebook](#)

R

»



L

AR

»

Name (required)

Comment

Email (required)

Website



more...

T

A

P

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

☐ Rất đẹp

☐ Đẹp

☐ Bình thường

☐ Xấu

[View Results](#)

[Polls Archive](#)