CSE 160 – Computer Networks
University of California, Merced

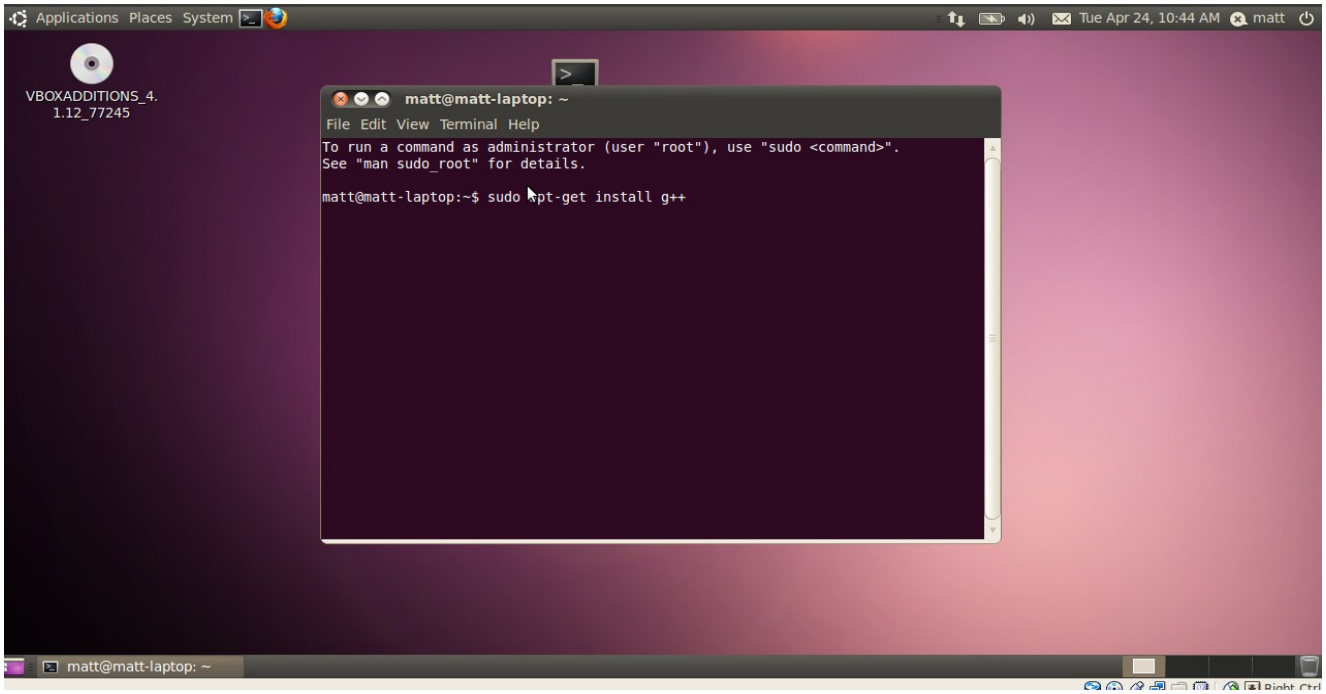
Project 0: TinyOS

Introduction:

This semester you will be implementing various algorithms and protocols used on the internet today. Your playground for this pseudo-network that you will create will be TinyOS, a very lightweight operating system that runs applications written in NesC, an extension of C. This project is meant to help you setup your environment for developing your network and demonstrate how to run and compile a TinyOS simulator called TOSSIM. The steps below are comprehensive and should take you through setting up TinyOS without issue. Note that you **WILL NEED A LINUX SYSTEM** available to you for the projects this semester. Using virtual box will work for the purposes of running TOSSIM however you may not be able to transfer your application over to a mote. This guide is demonstrated using a fresh install of Ubuntu 10.04.

Pre-Setting up TinyOS/Eclipse/Python:

Make sure you having the following software installed:
G++: sudo apt-get install g++

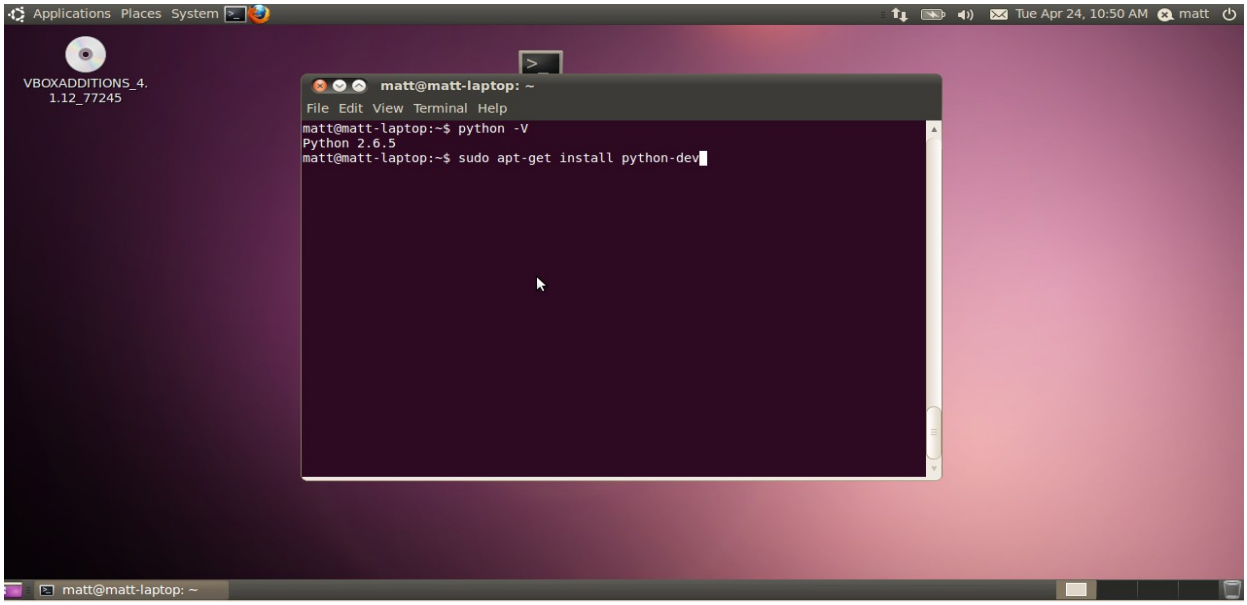


Python and Python-dev:
Ubuntu 10.04 comes pre-installed with python however check that you are running version 2.6.5 or later with the command:

python -V

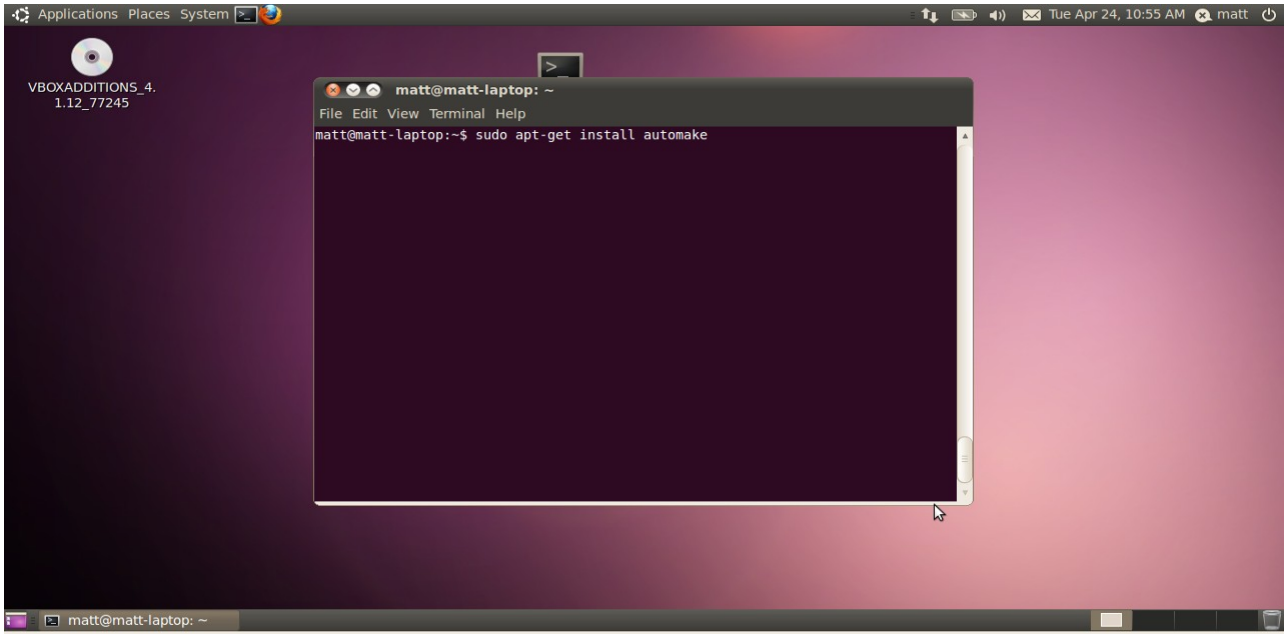
You will need to install python-dev using the following command:

sudo apt-get install python-dev



Automake:

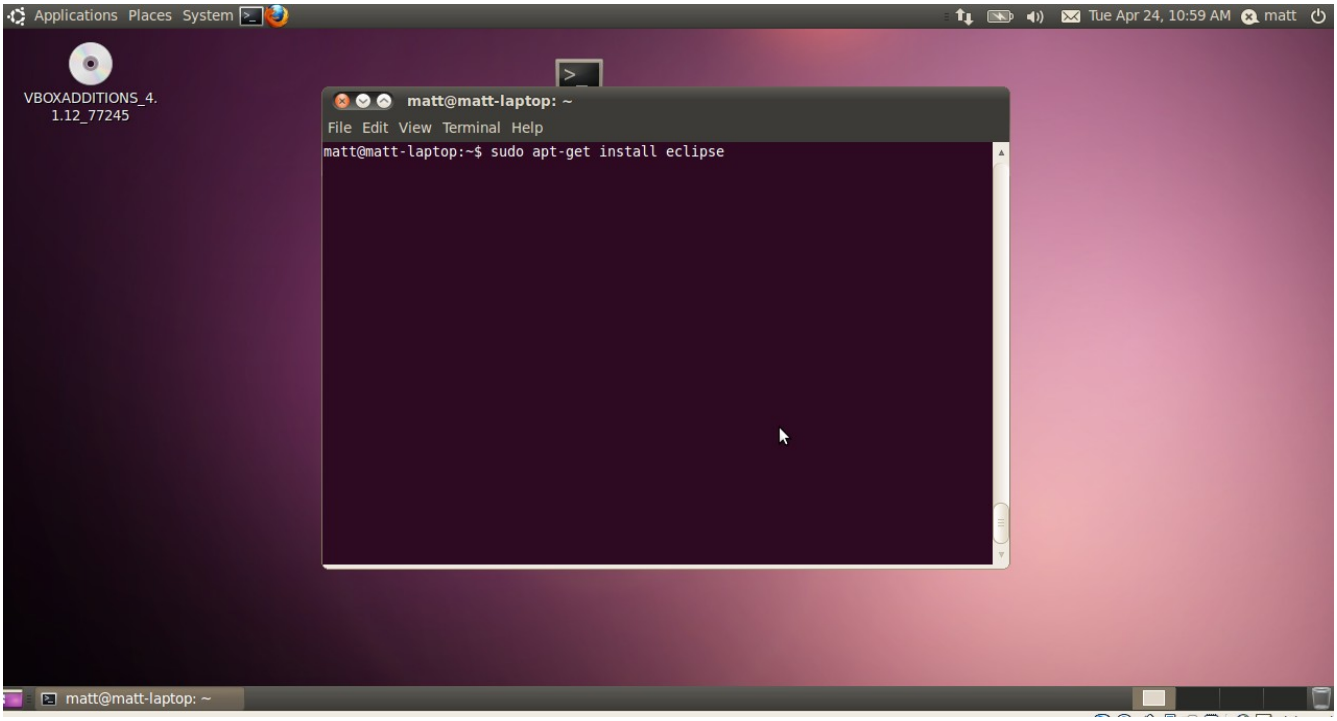
sudo apt-get install automake



Eclipse:

It is highly recommended that you use eclipse as your coding environment. It is certainly possible to use vim or gedit however this guide will not cover how to set up those environments. To install eclipse used the command:

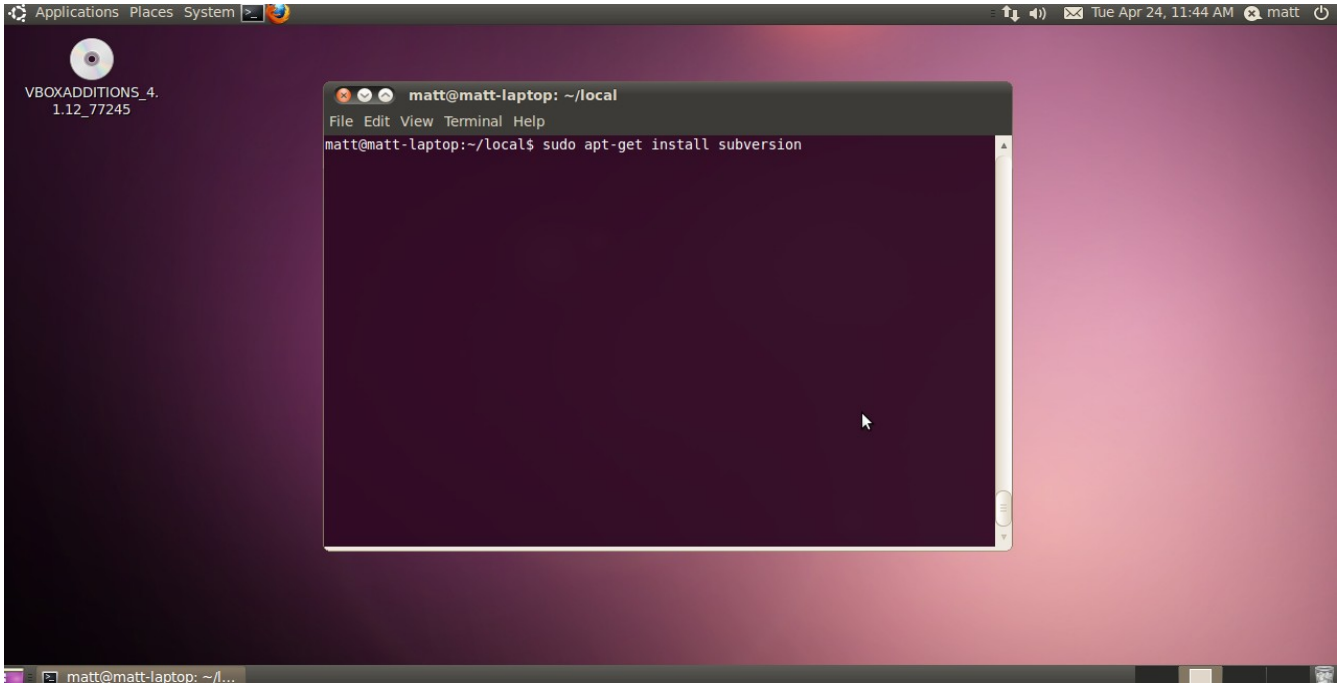
sudo apt-get install eclipse



Project 0 - TinyOS

Subversion:

sudo apt-get install subversion

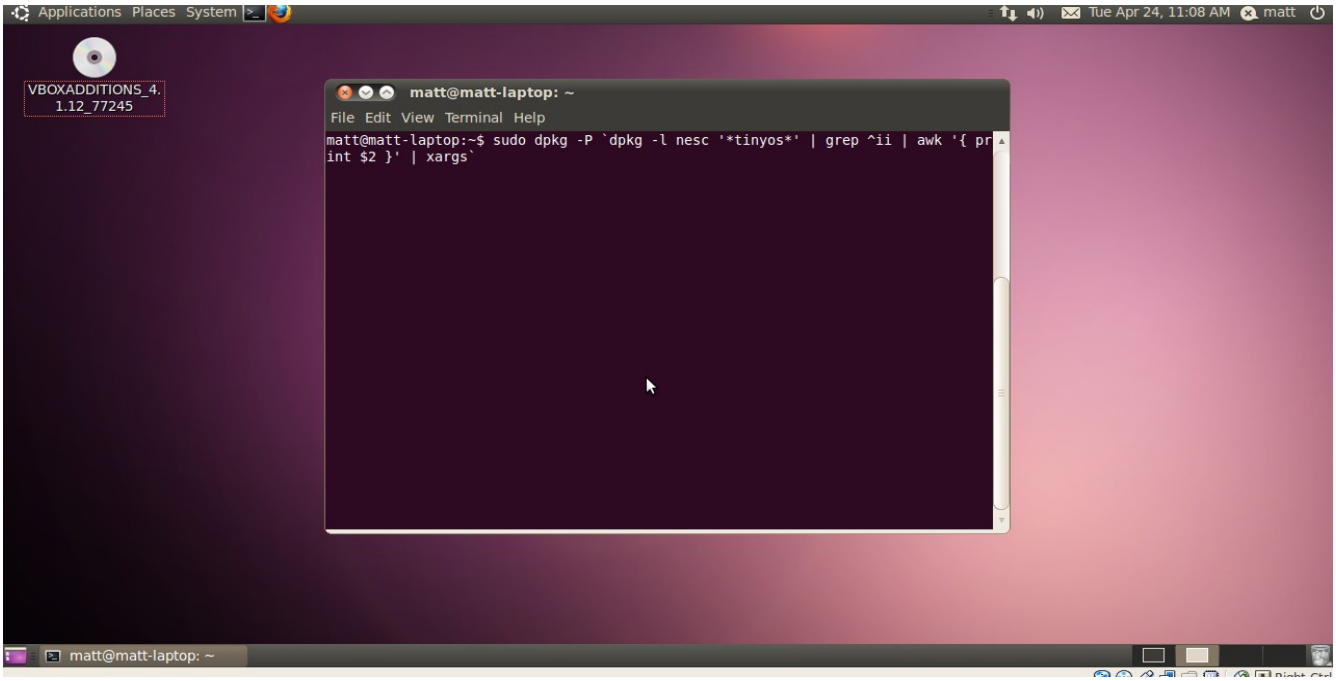


Setting up TinyOS/Eclipse/Python:

(SKIP THIS STEP IF YOU NEVER HAVE INSTALLED TinyOS!)

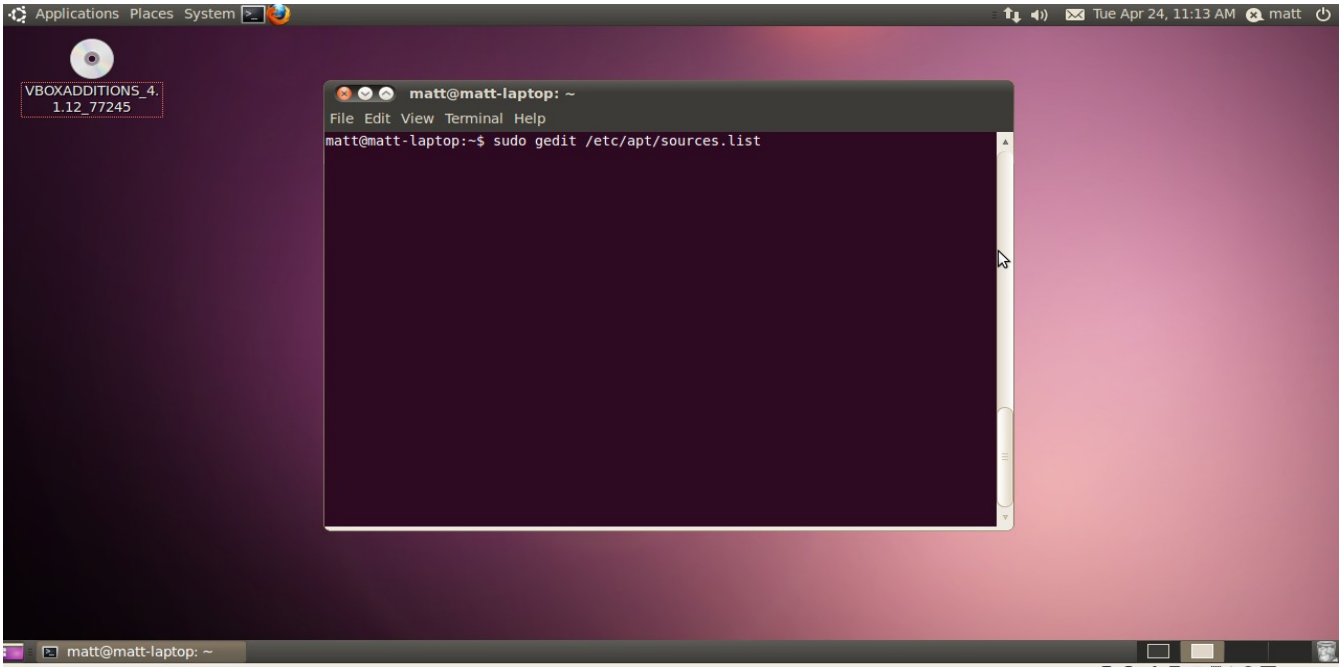
If you have tinyos currently installed and wish to re-install open a terminal and type in:

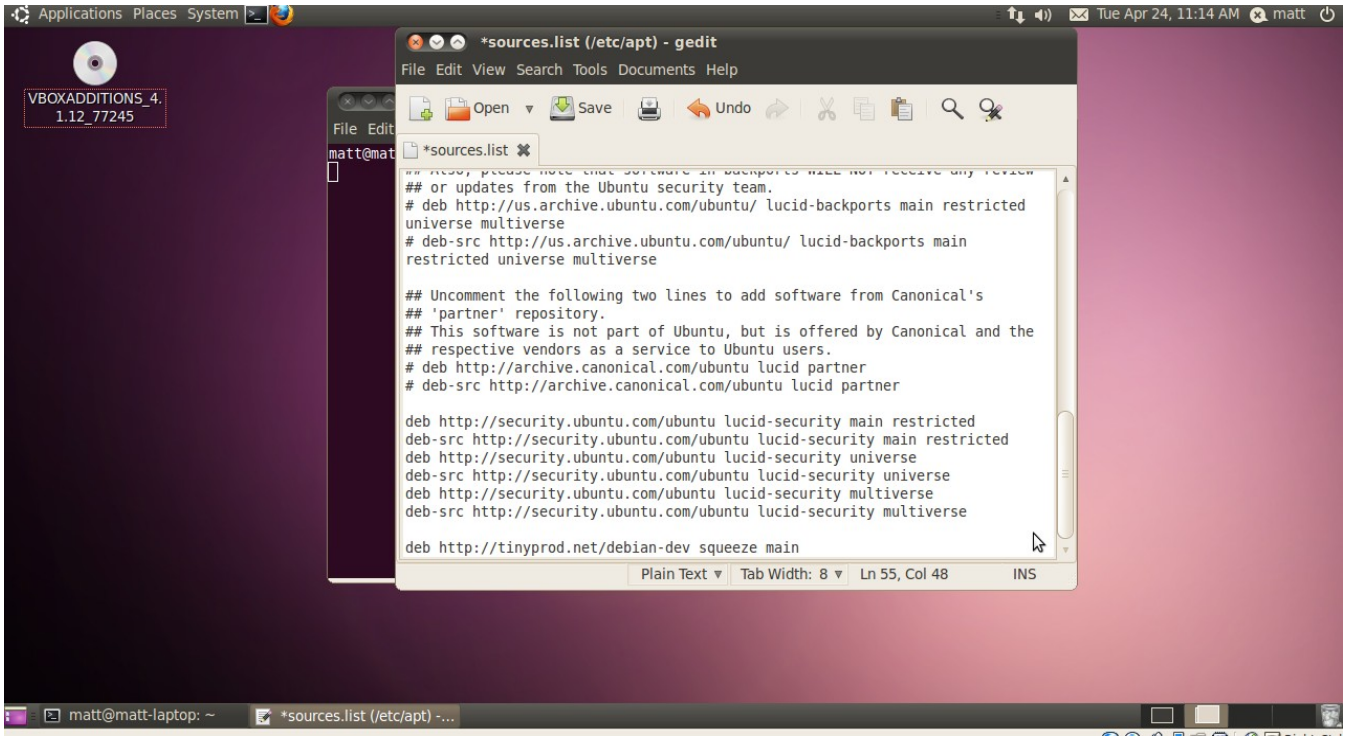
sudo dpkg -P `dpkg -l nesc '*tinyos*' | grep ^ii | awk '{ print $2 }' | xargs`



Open /etc/apt/sources.list with gedit/vim using sudo and add these lines at the bottom of the file:

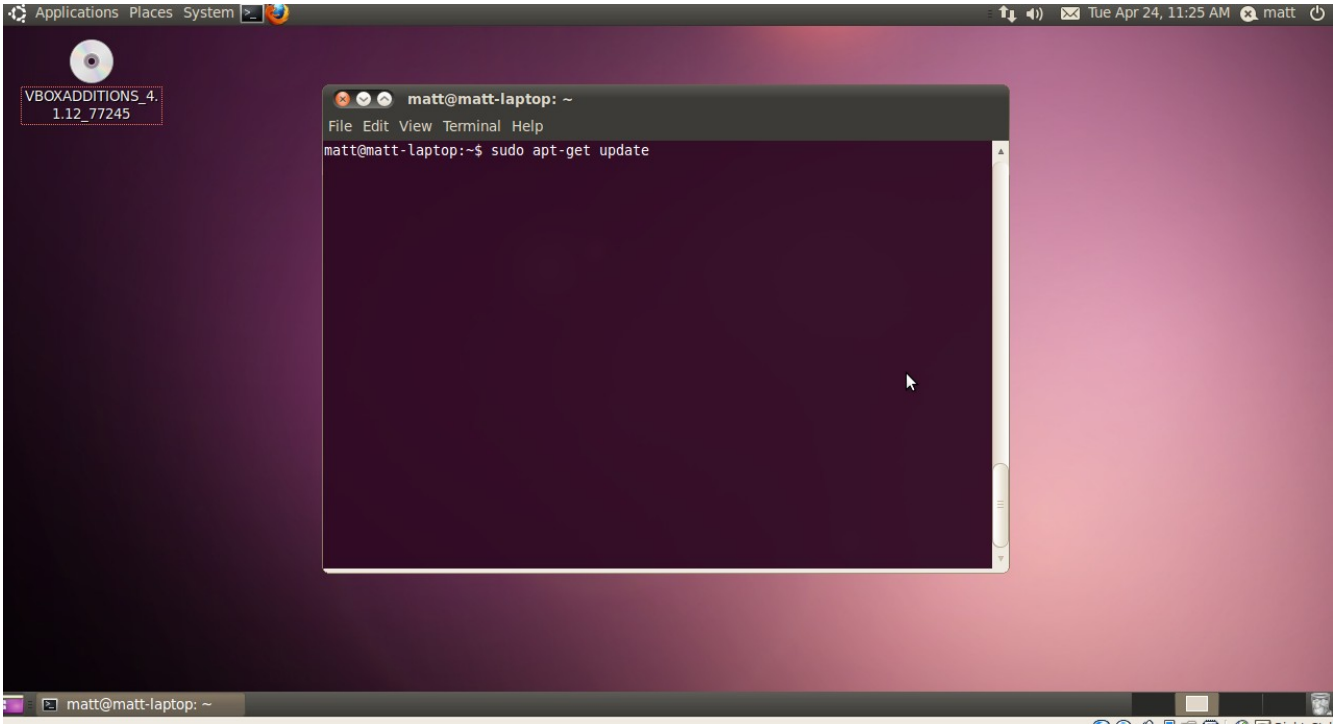
deb http://tinyprod.net/debian-dev squeeze main
deb http://tinyprod.net/repos/debian msp430-46 main

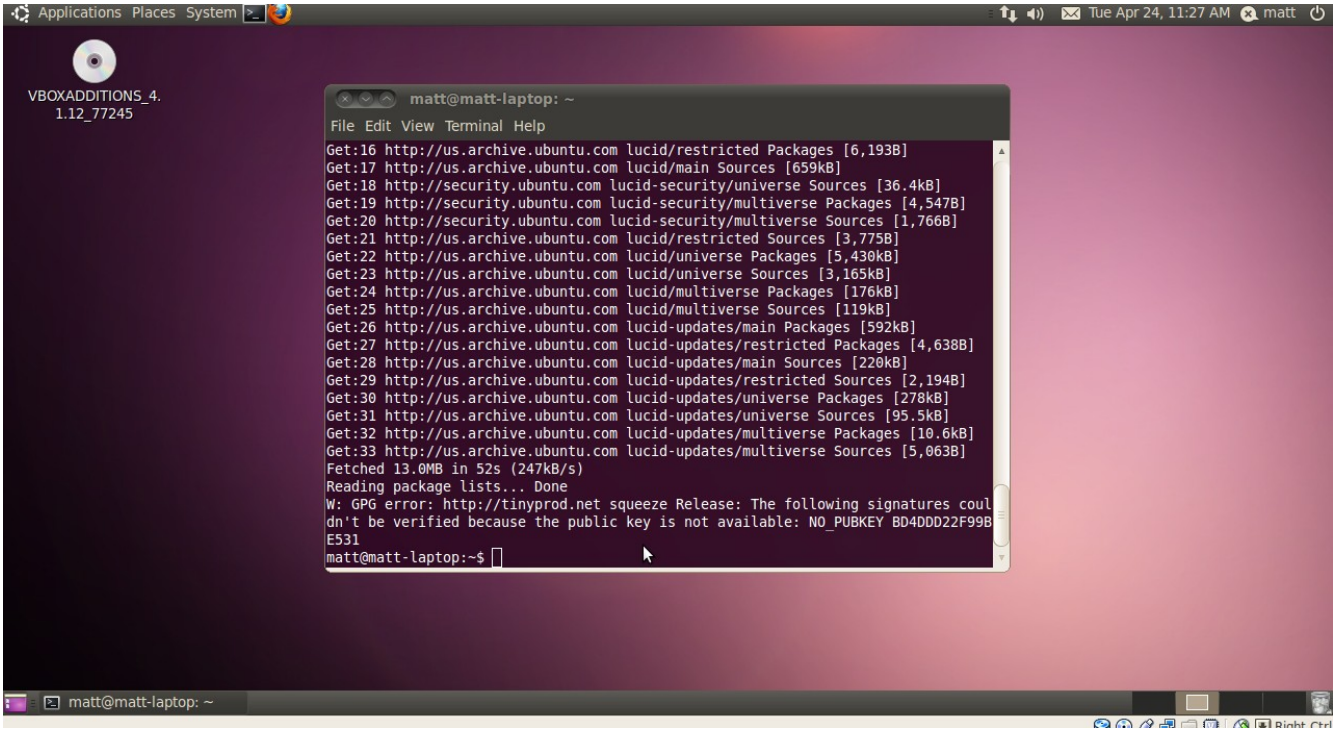




Save and close this file.
In the terminal enter (Note: There will be an "error" that apt will complain about, ignore this):

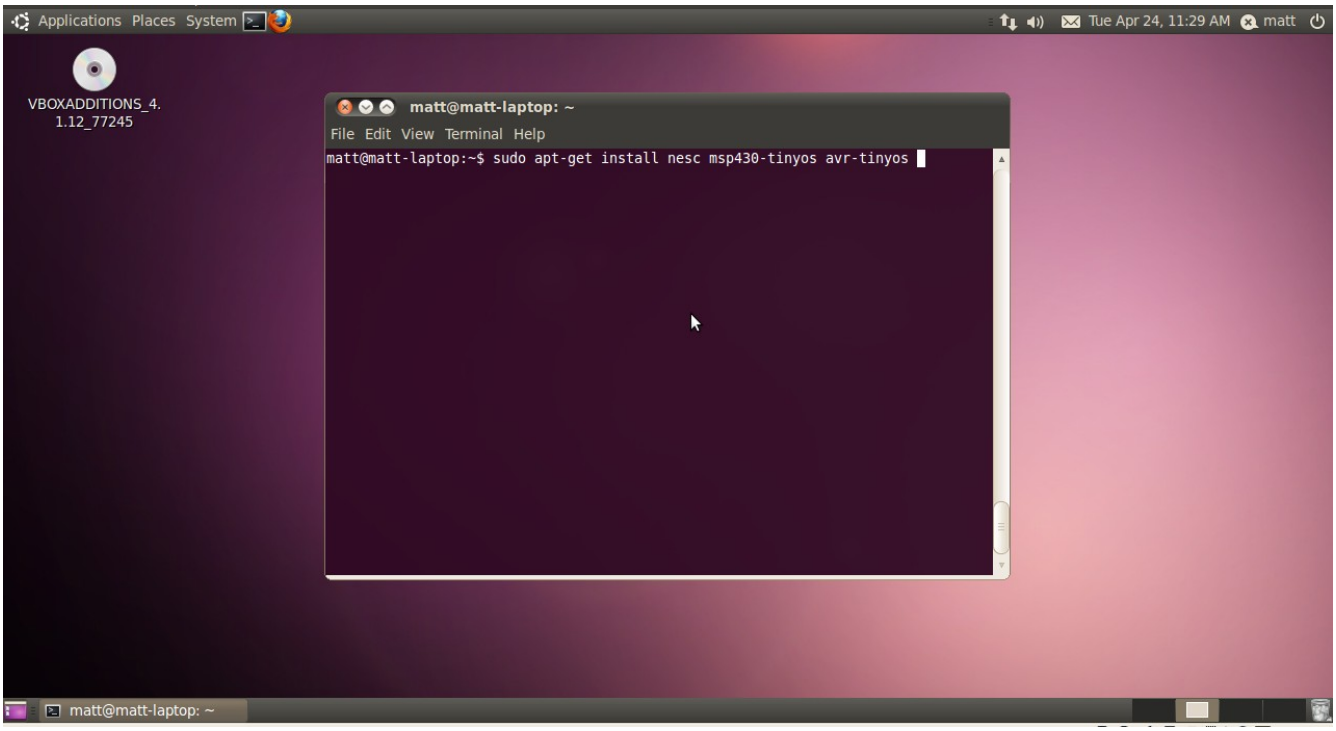
sudo apt-get update





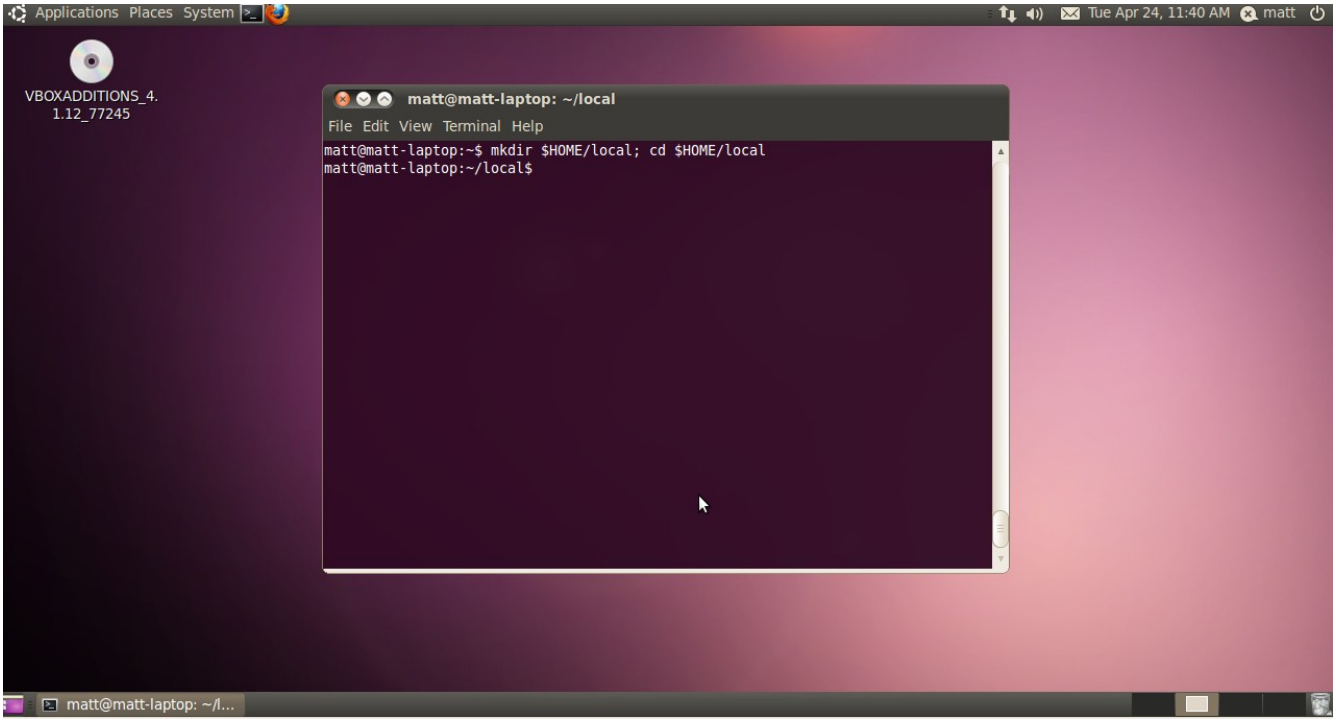
Next install NesC with the following command:

sudo apt-get install nesc msp430-46 avr-tinyos



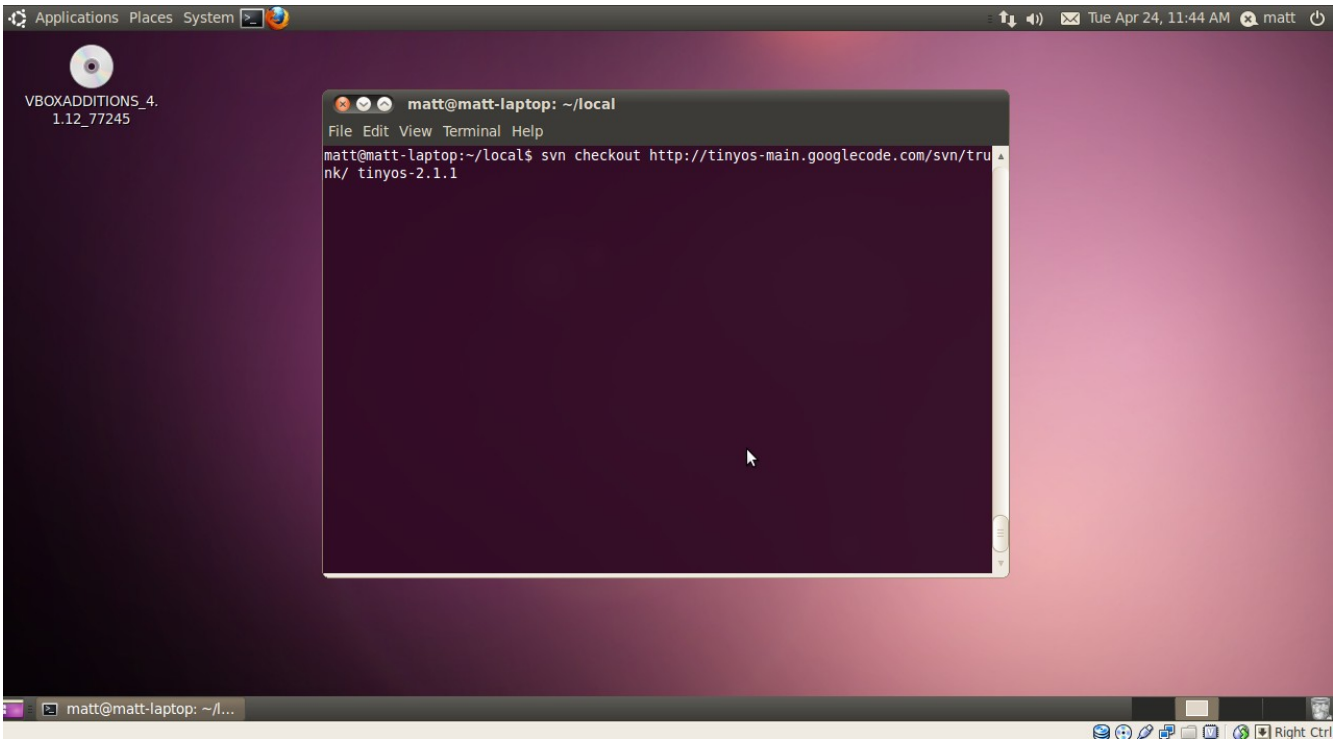
Create and move to the following folder:

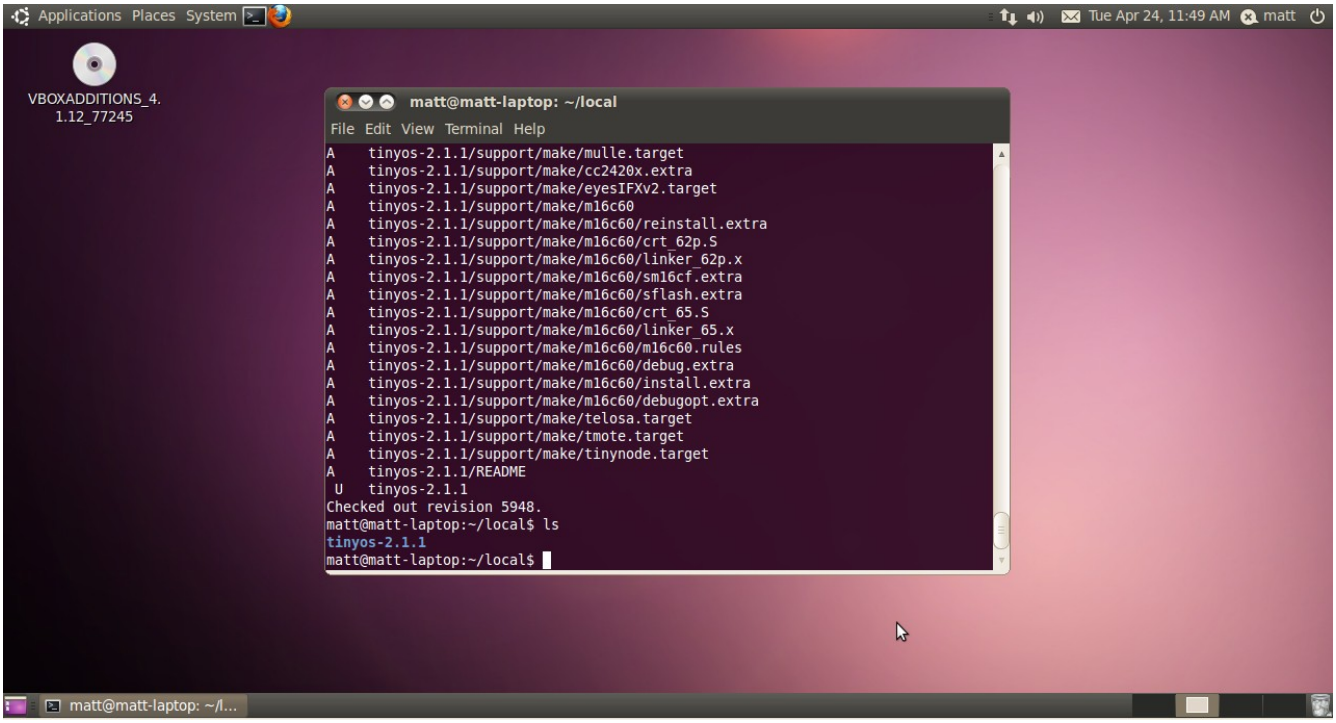
mkdir $HOME/local; cd $HOME/local



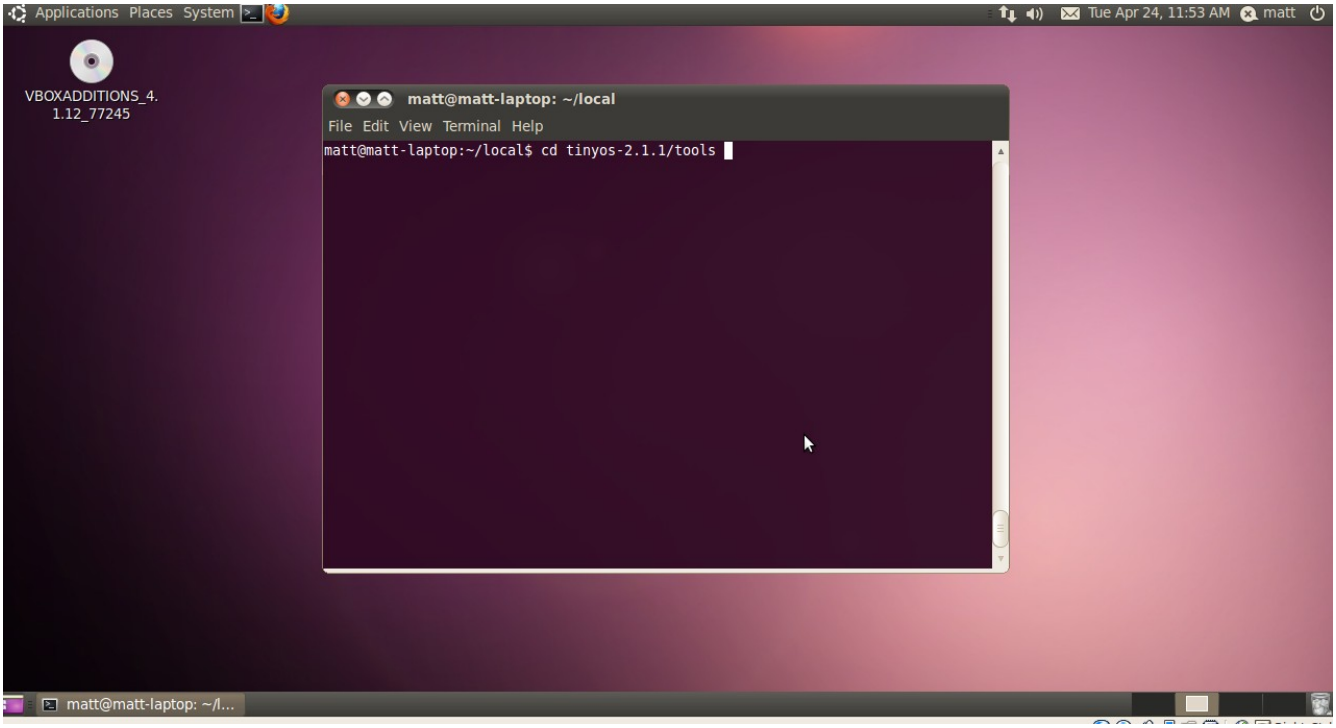
Next check out TinyOS:

svn checkout http://tinyos-main.googlecode.com/svn/trunk/ tinyos-2.1.1



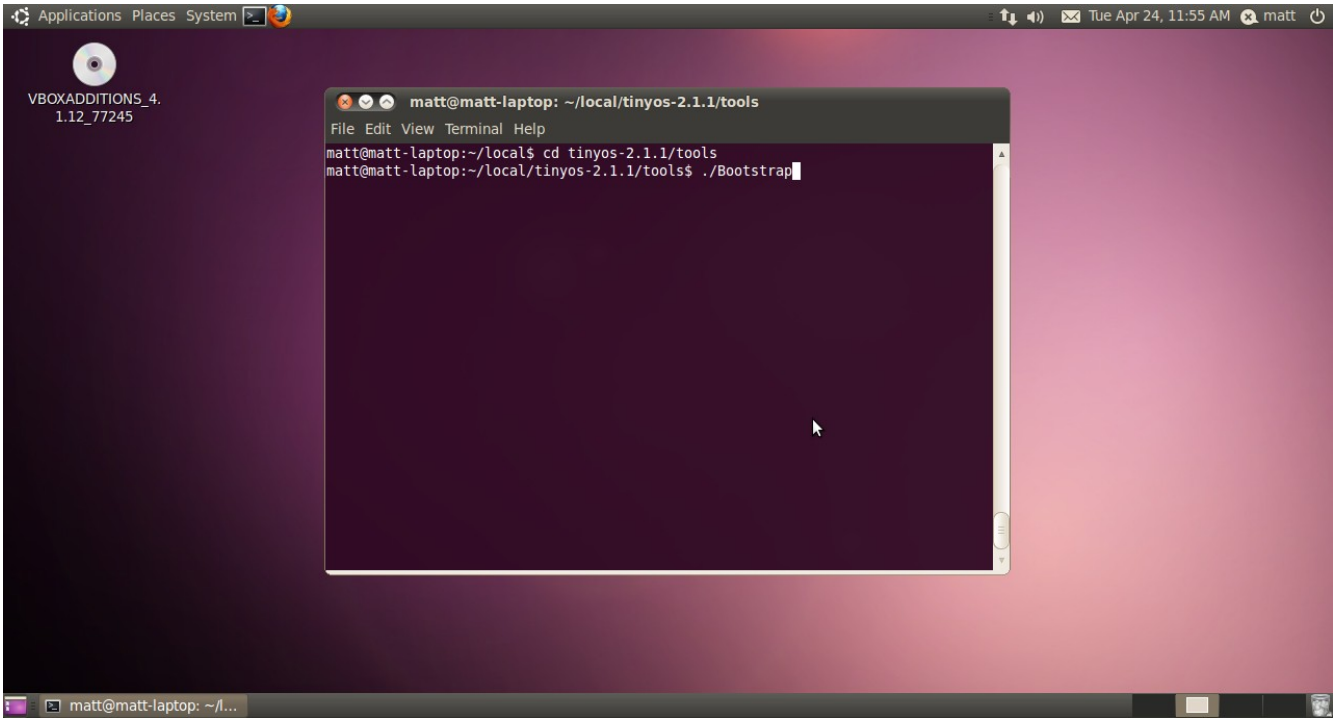


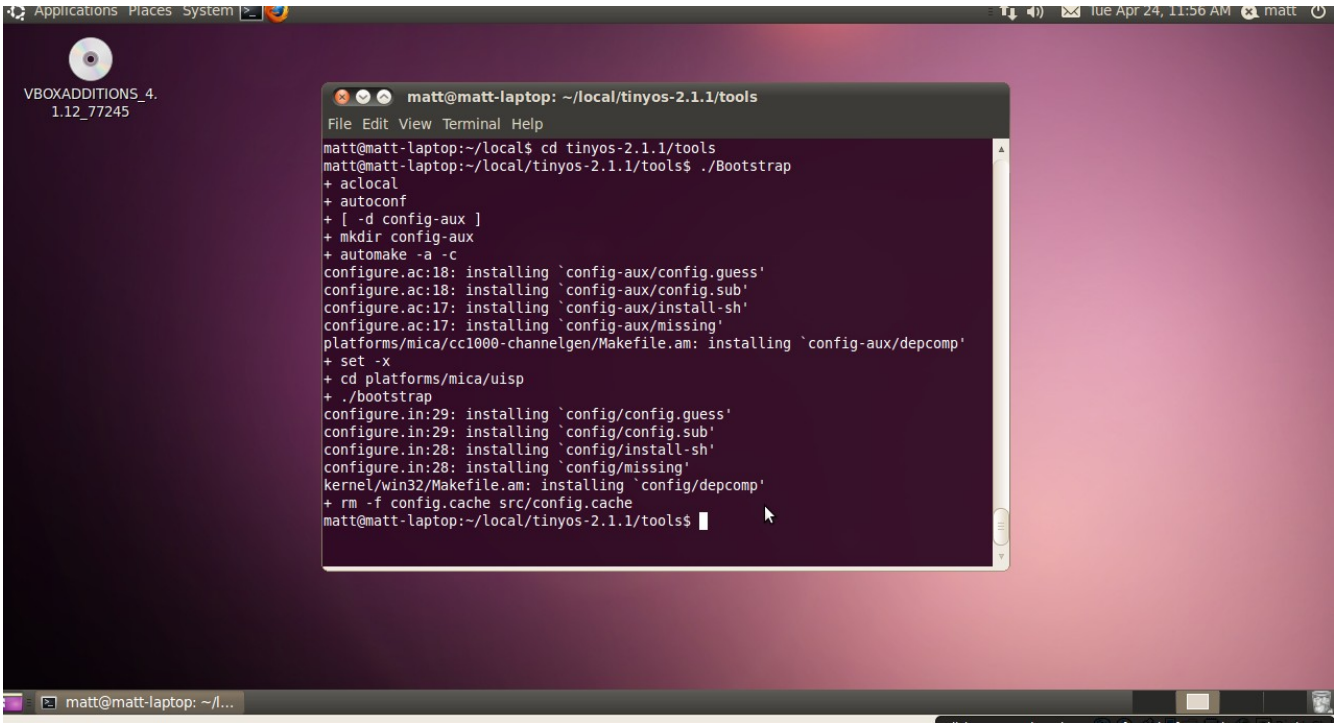
Move to the following folder:
cd tinyos-2.1.1/tools



Run the bash script within this folder:

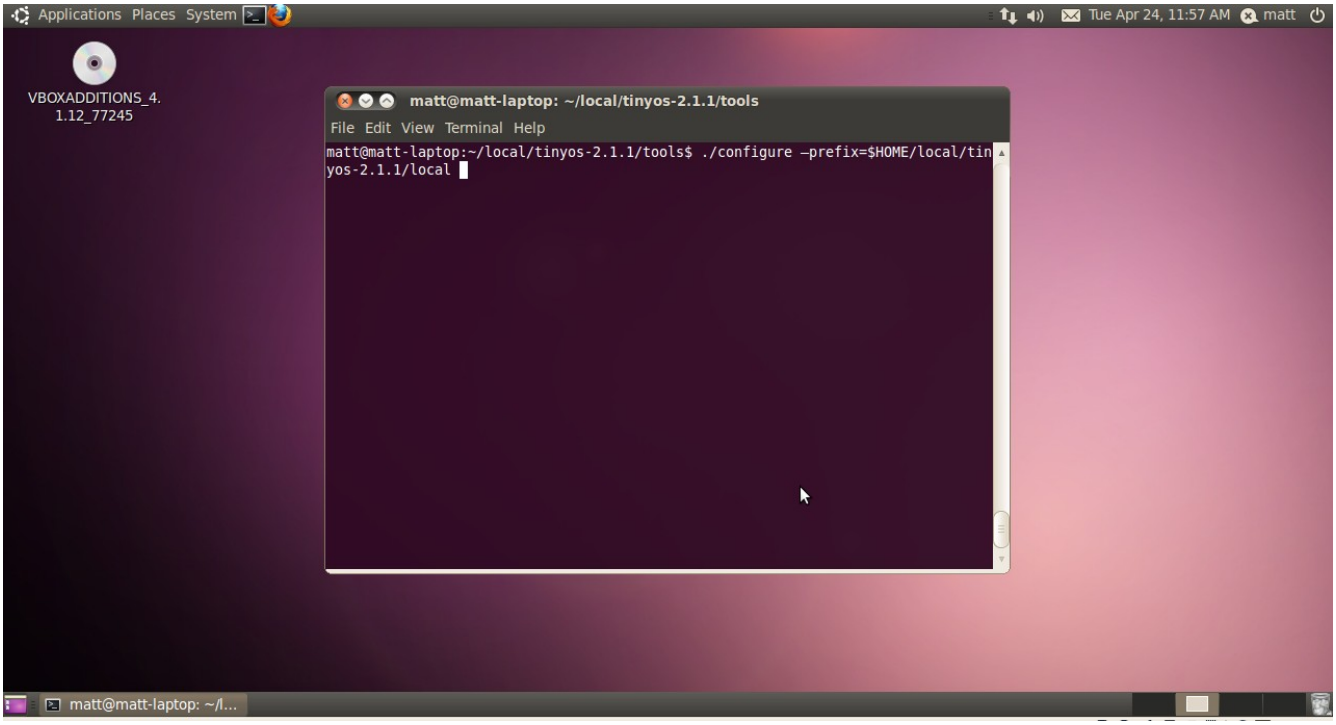
./Bootstrap





Next run the following command:

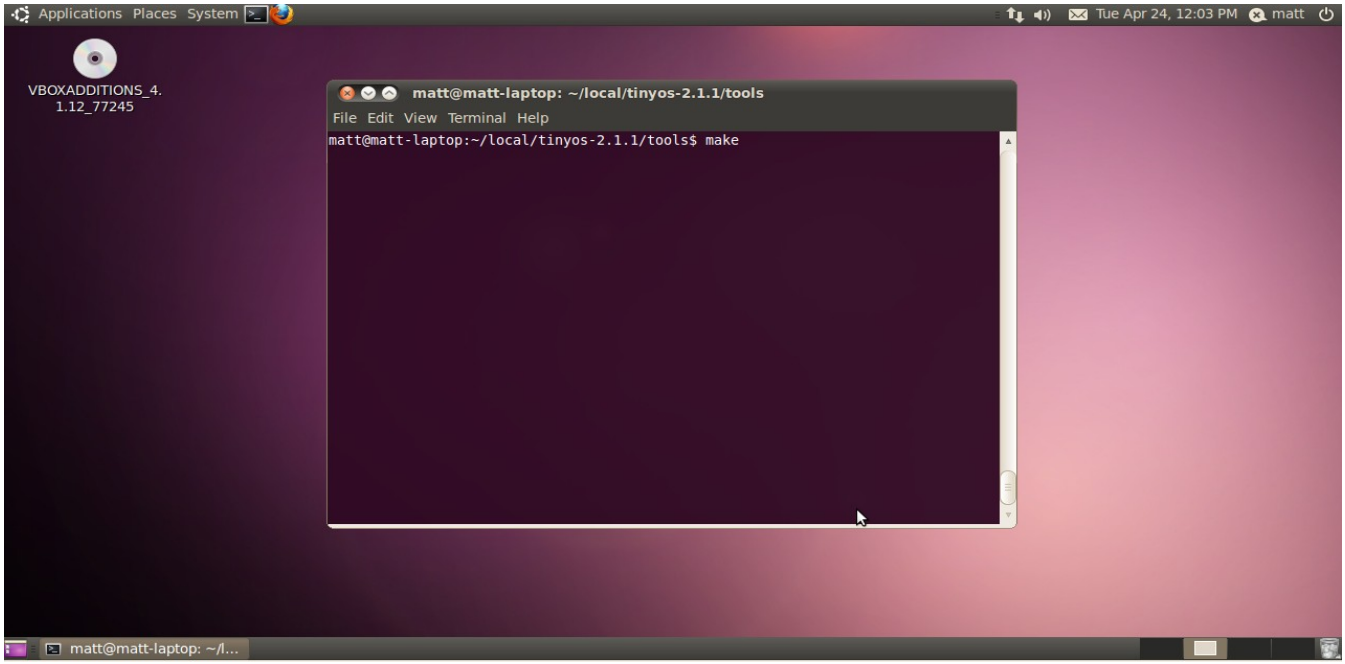
./configure –prefix=$HOME/local/tinyos-2.1.1/local



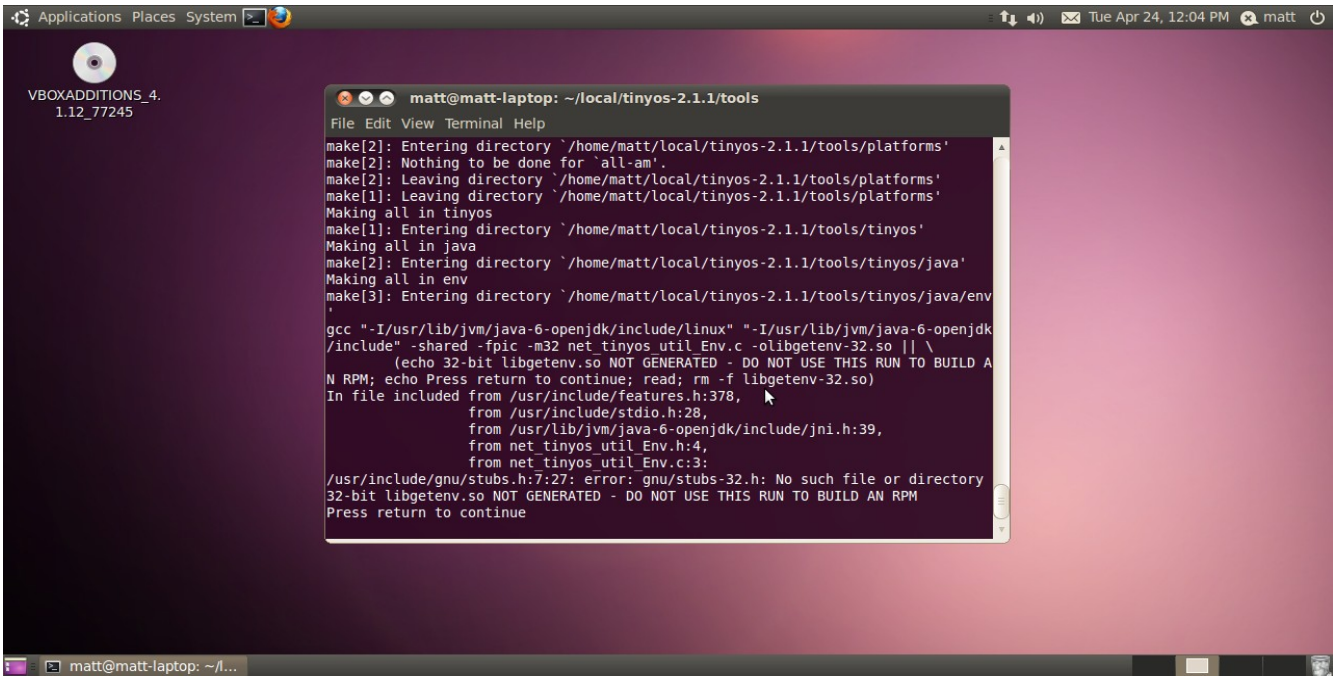
NOTE: If you get a message saying "configure: error: invalid variable name: `–prefix' ", then it is a copy/paste error and you need to retype the "-" in "-prefix".

Next run the following command:

make

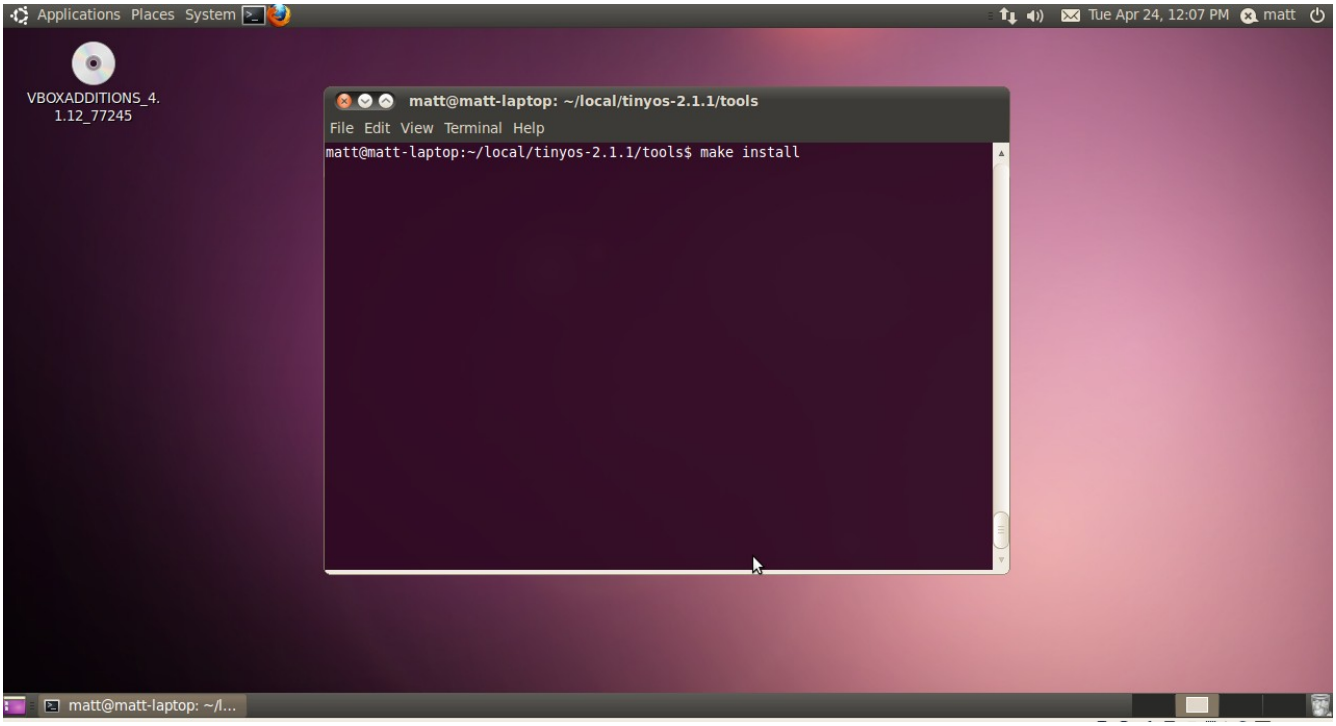


You will be prompted to "not use this run to build an rpm" several times, this is normal so simply press enter each time.



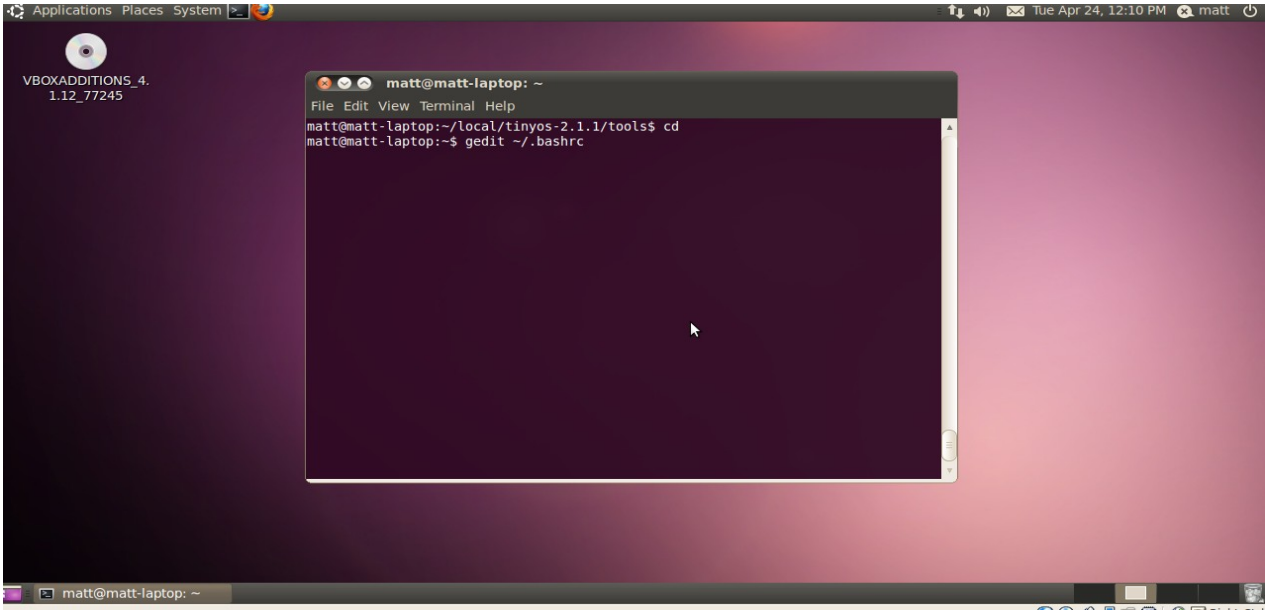
Next run the following command (You will see the same message as shown above, continue to hit enter each time it pops up):

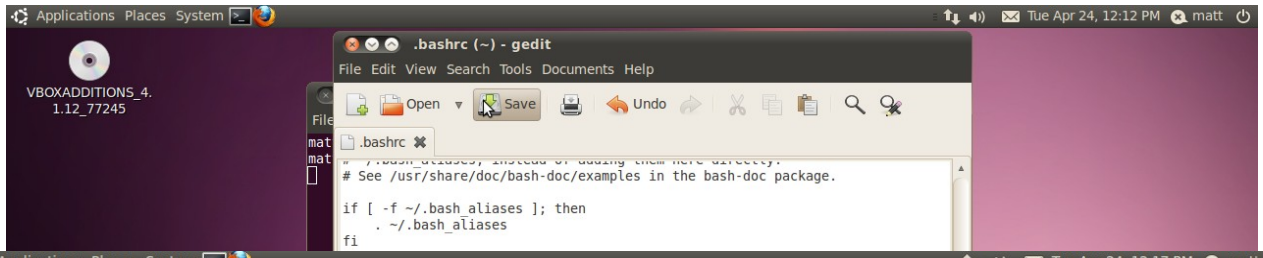
make install

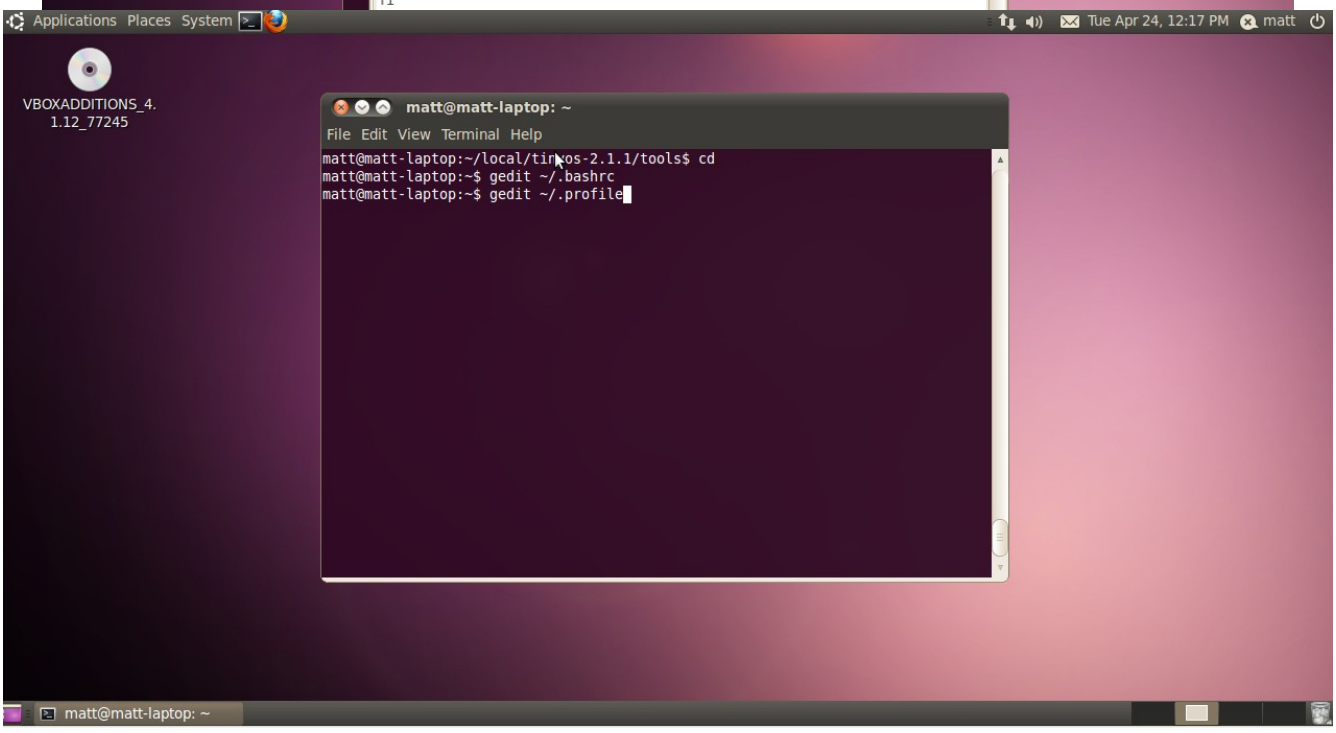


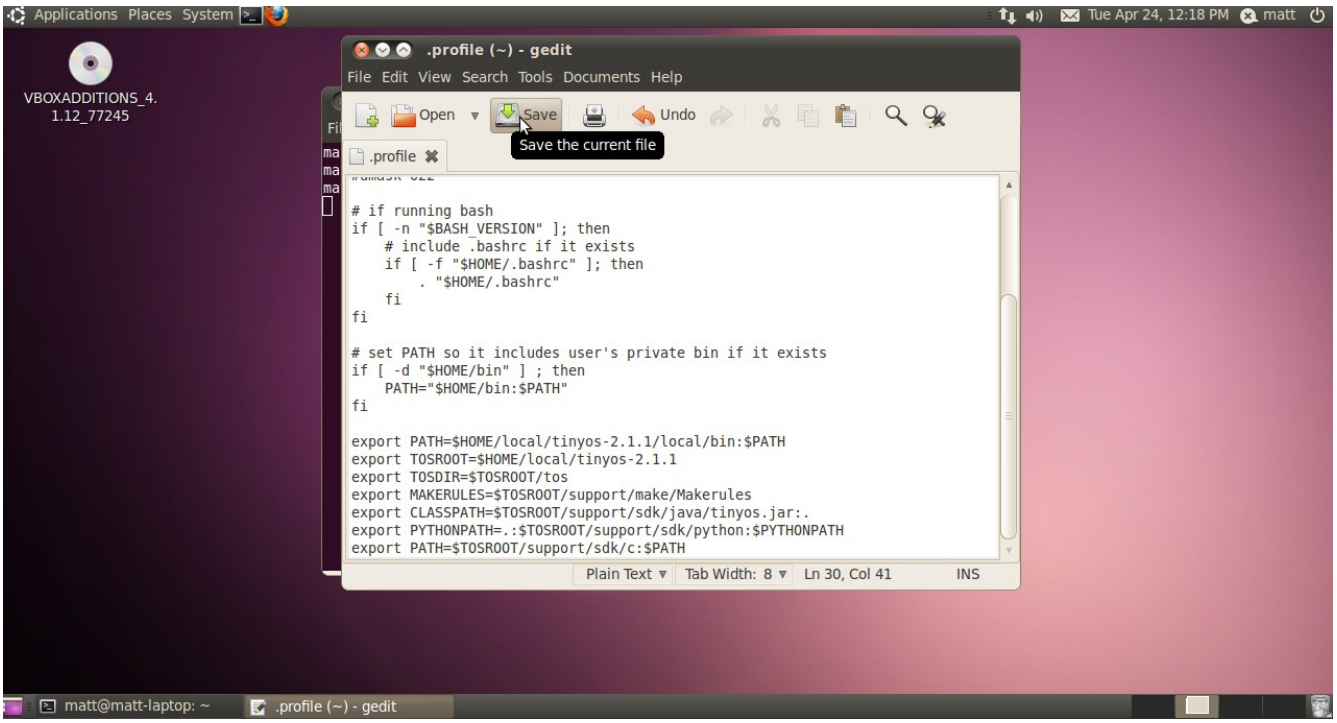
Next you need to setup the environment variables in **BOTH** your .bashrc and .profile files. Append the following to the end of each file:

```
export PATH=$HOME/local/tinyos-2.1.1/local/bin:$PATH
export TOSROOT=$HOME/local/tinyos-2.1.1
export TOSDIR=$TOSROOT/tos
export MAKERULES=$TOSROOT/support/make/Makerules
export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.
export PYTHONPATH=.:$TOSROOT/support/sdk/python:$PYTHONPATH
export PATH=$TOSROOT/support/sdk/c:$PATH
```

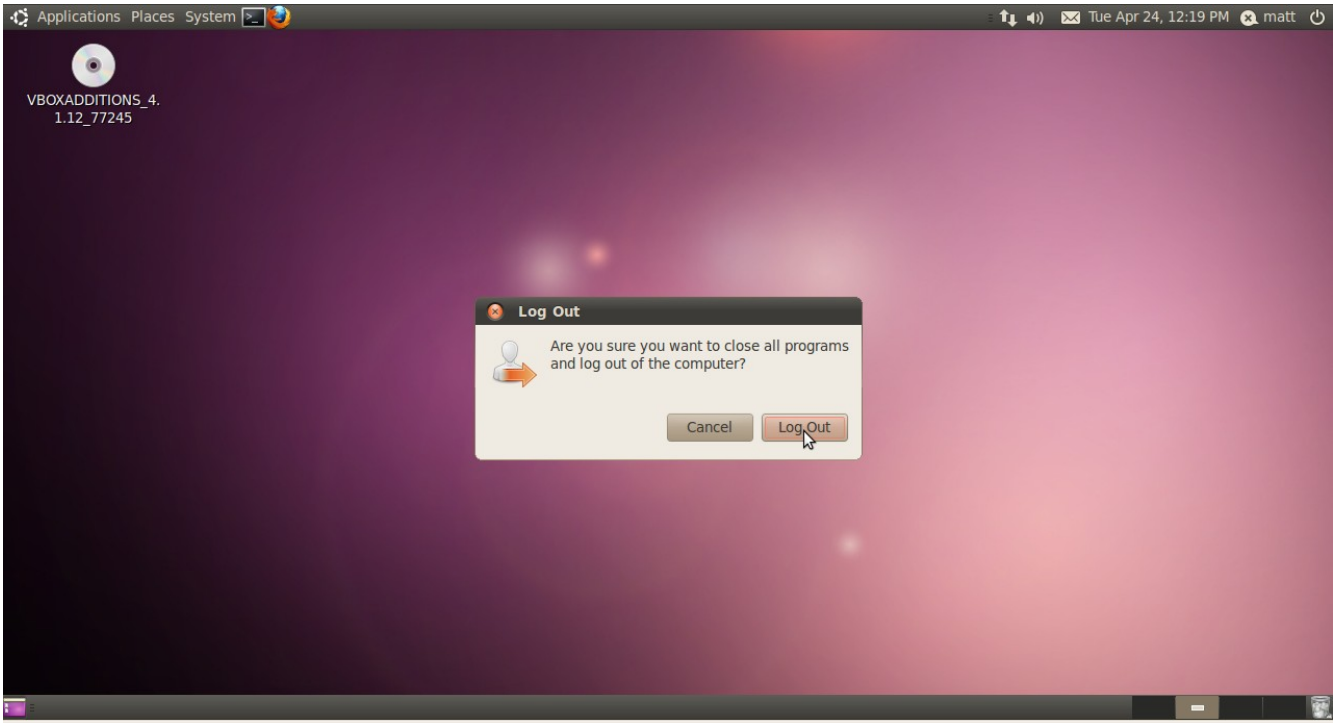

Applications Places System
Tue Apr 24, 12:10 PM matt
VBOXADDITIONS_4.
1.12_77245
matt@matt-laptop: ~
File Edit View Terminal Help
matt@matt-laptop:~/local/tinyos-2.1.1/tools$ cd
matt@matt-laptop:~$ gedit ~/.bashrc
matt@matt-laptop: ~


Applications Places System
Tue Apr 24, 12:12 PM matt
VBOXADDITIONS_4.
1.12_77245
.bashrc (~) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
.bashrc
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
if [-f ~/.bash_aliases]; then
. ~/.bash_aliases
fi

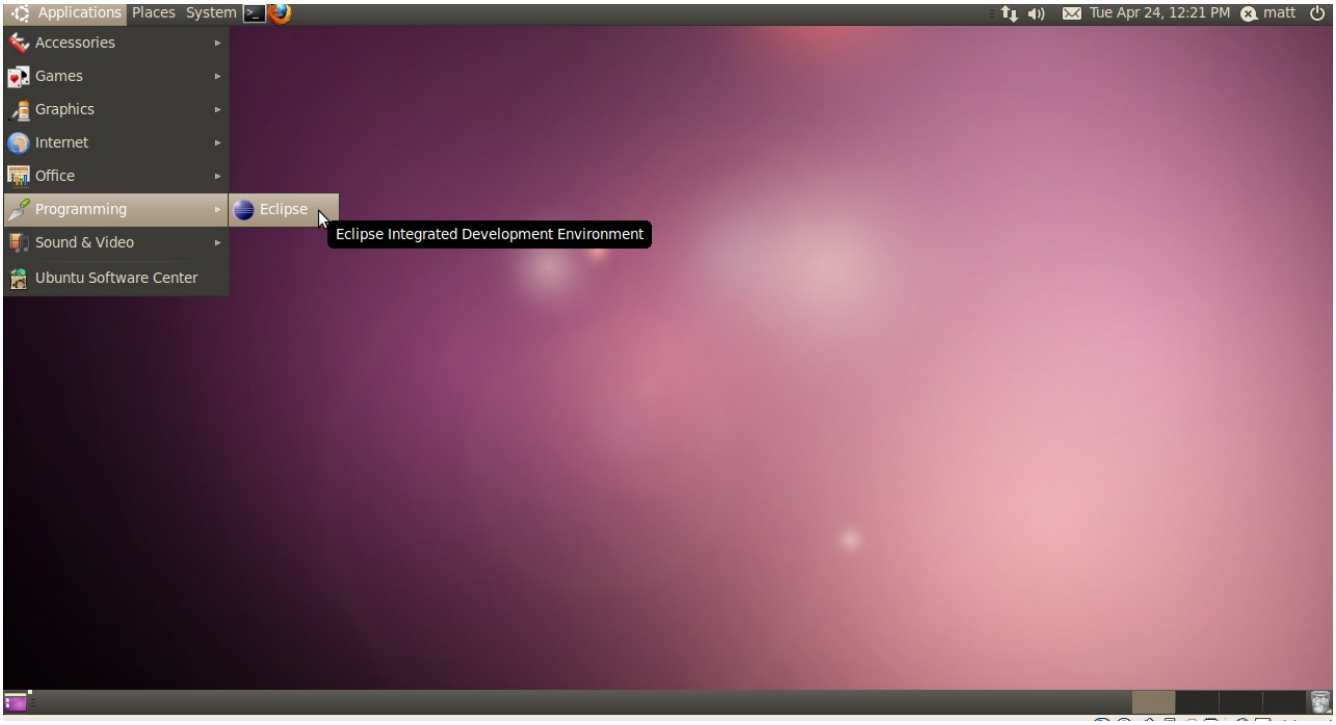

Applications Places System
Tue Apr 24, 12:17 PM matt
VBOXADDITIONS_4.
1.12_77245
matt@matt-laptop: ~
File Edit View Terminal Help
matt@matt-laptop:~/local/tinyos-2.1.1/tools$ cd
matt@matt-laptop:~$ gedit ~/.bashrc
matt@matt-laptop:~$ gedit ~/.profile
matt@matt-laptop: ~



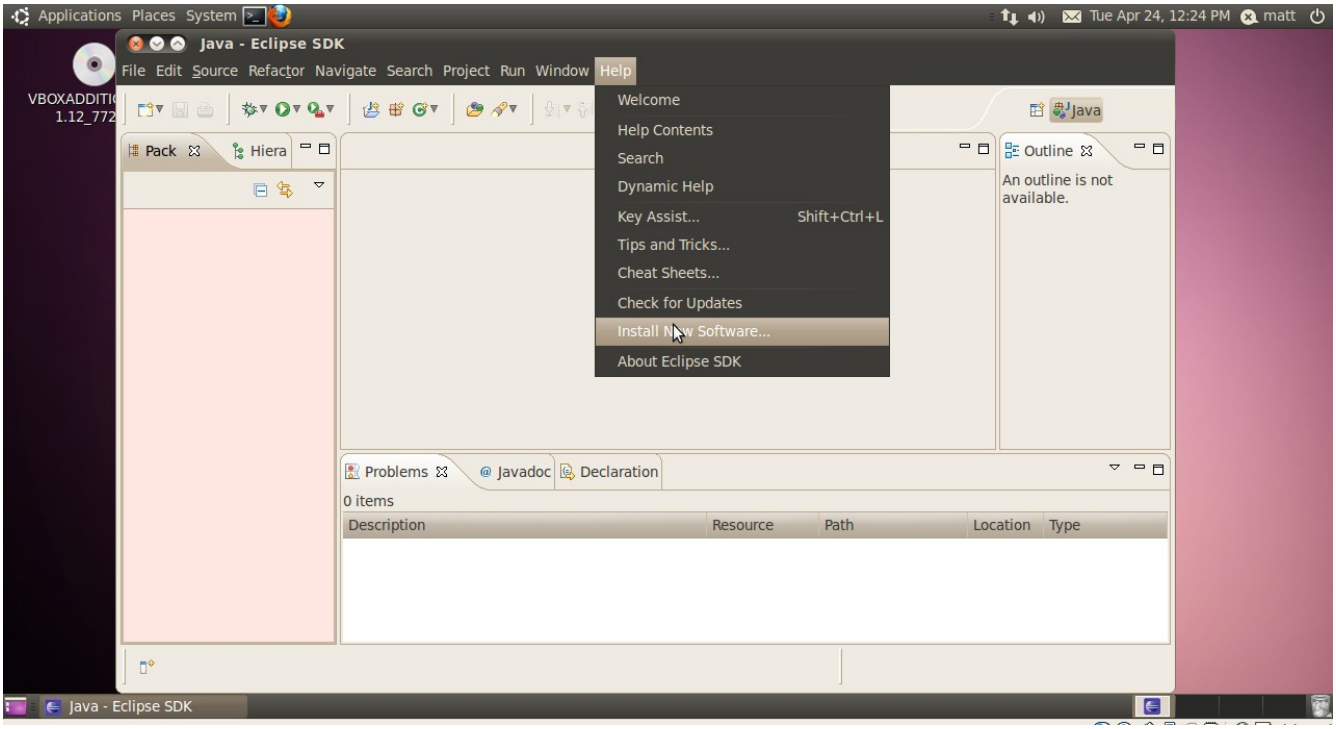
You will need to log out and back in for the changes to take effect.



After logging back you will want to install 2 addons within eclipse, Yeti2 and Pydev. Start by opening Eclipse:

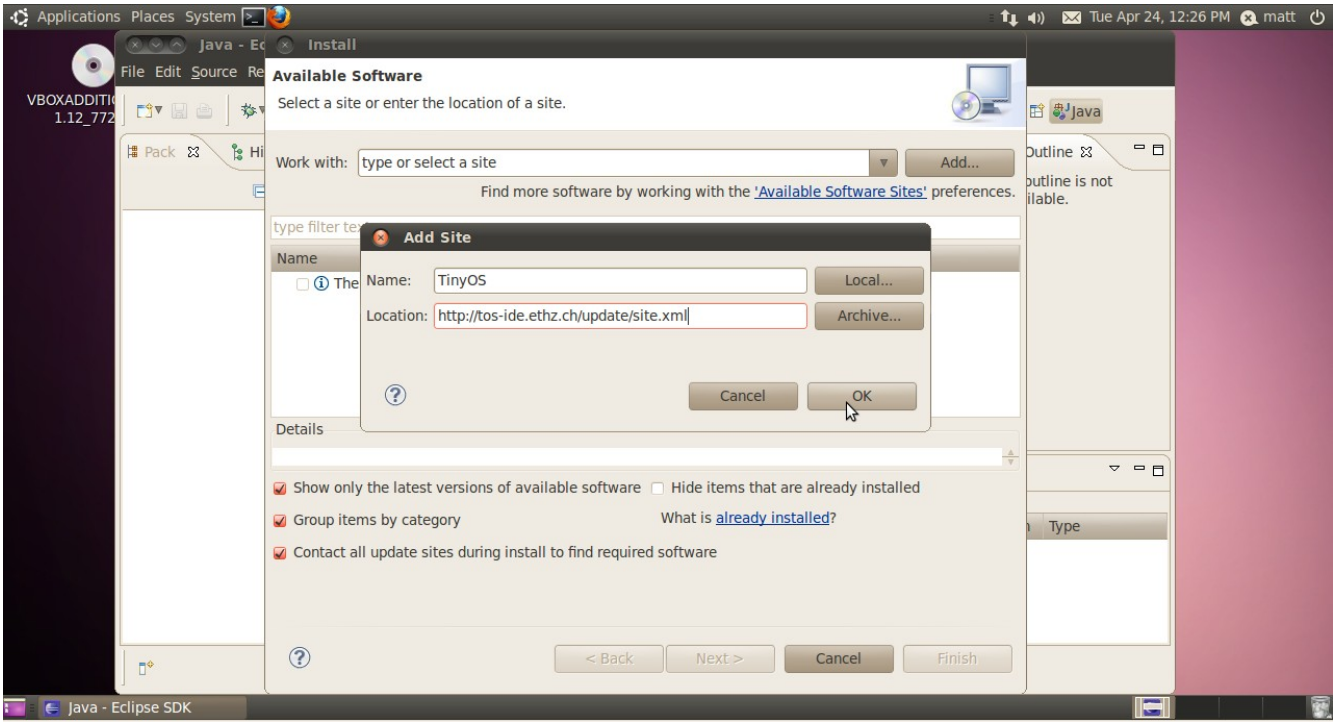


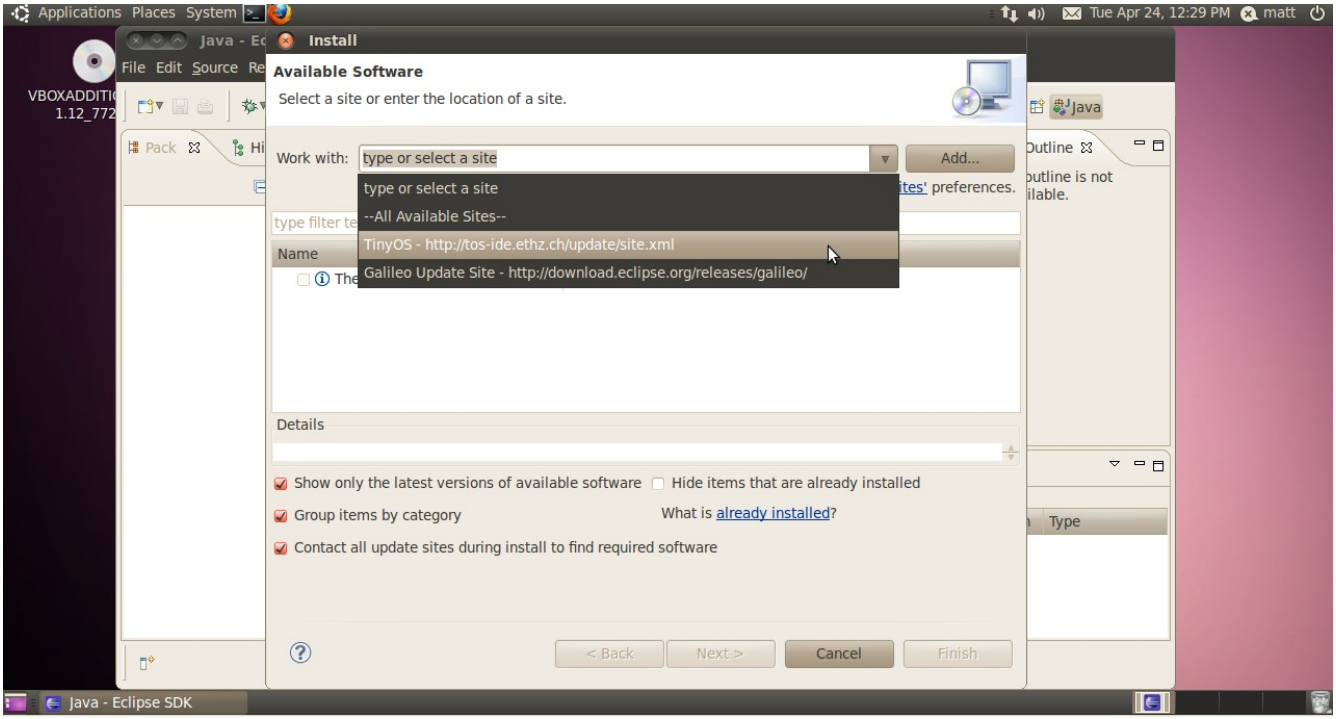
<u>Installing Yeti2</u>:
Under the Help tab click “install new software”.



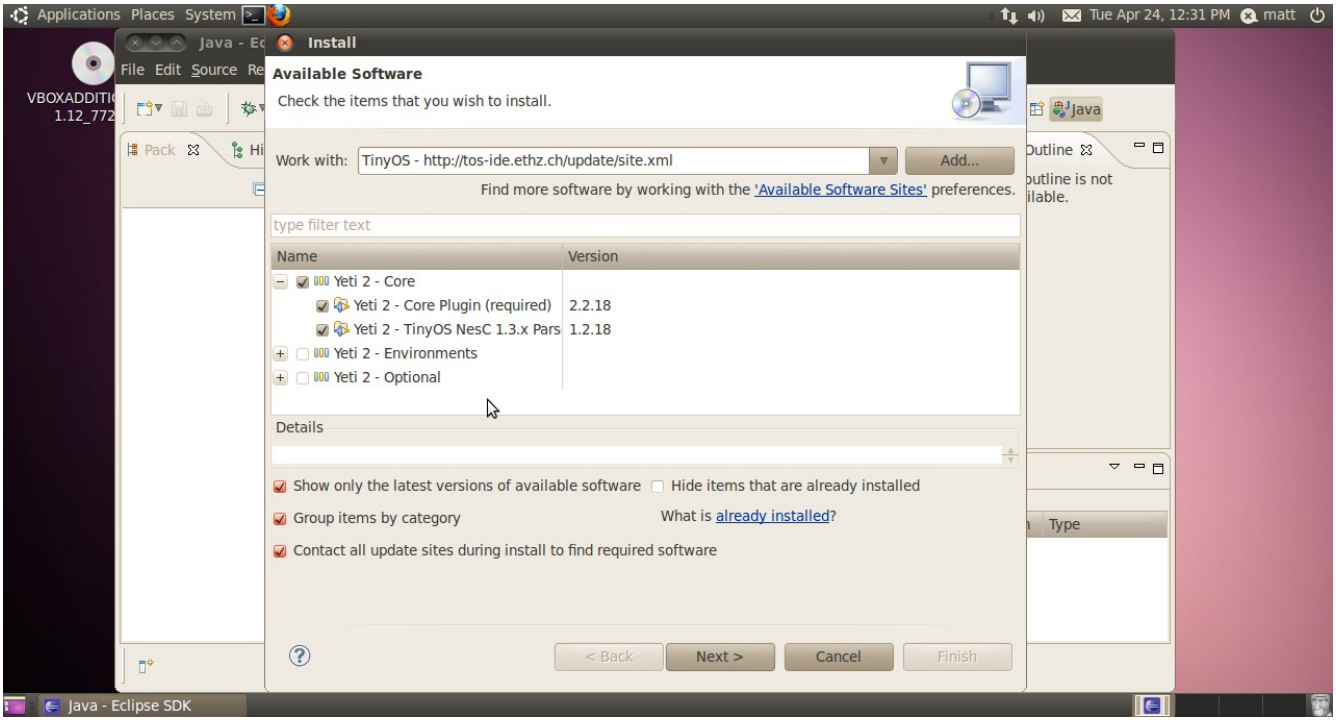
Click “ADD” type the address below and then select it from the drop down list labeled "Work with":

http://tos-ide.ethz.ch/update/site.xml

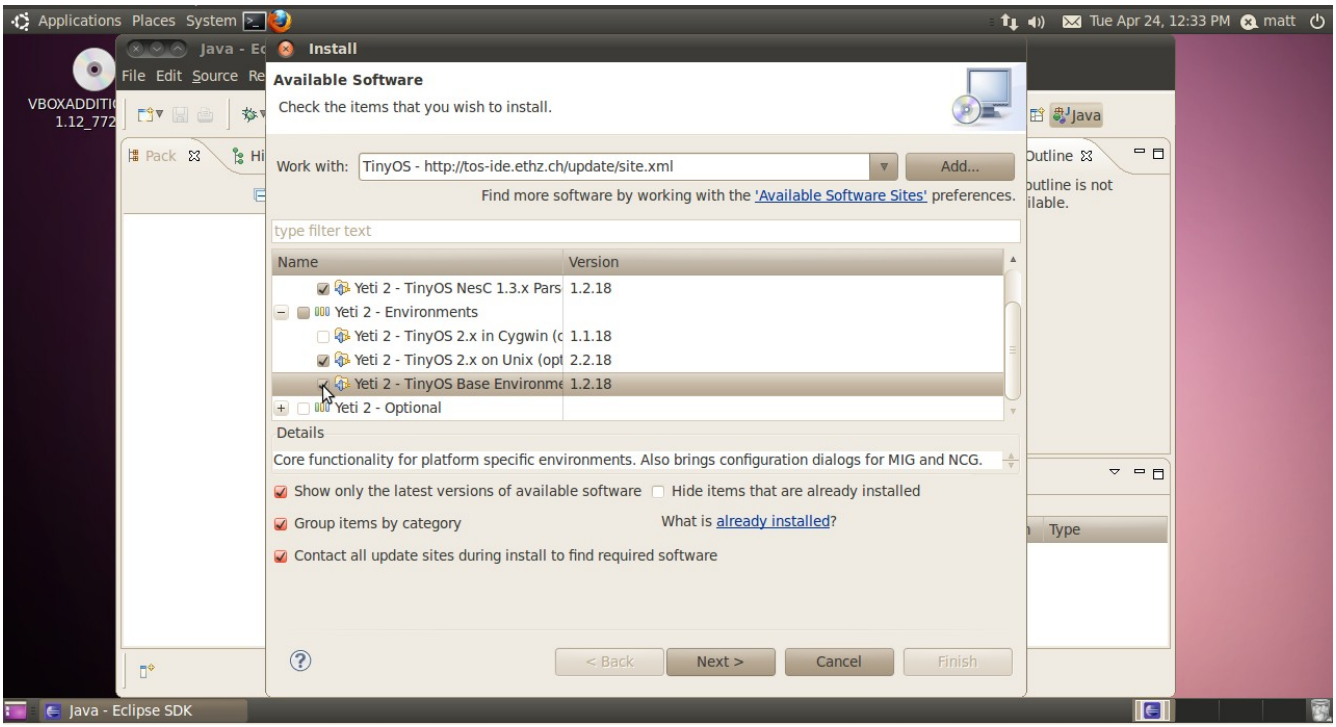




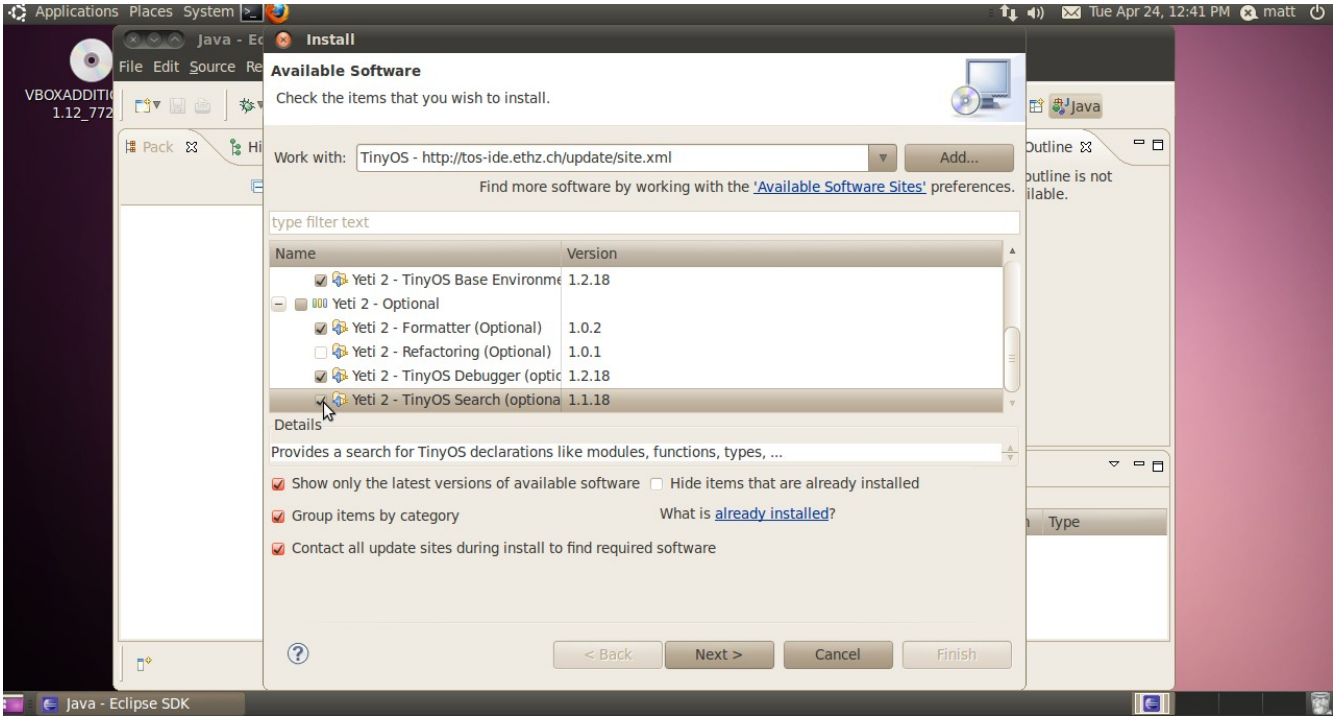
After a couple seconds a few options will pop-up.
Install everything under Yeti 2 – Core.



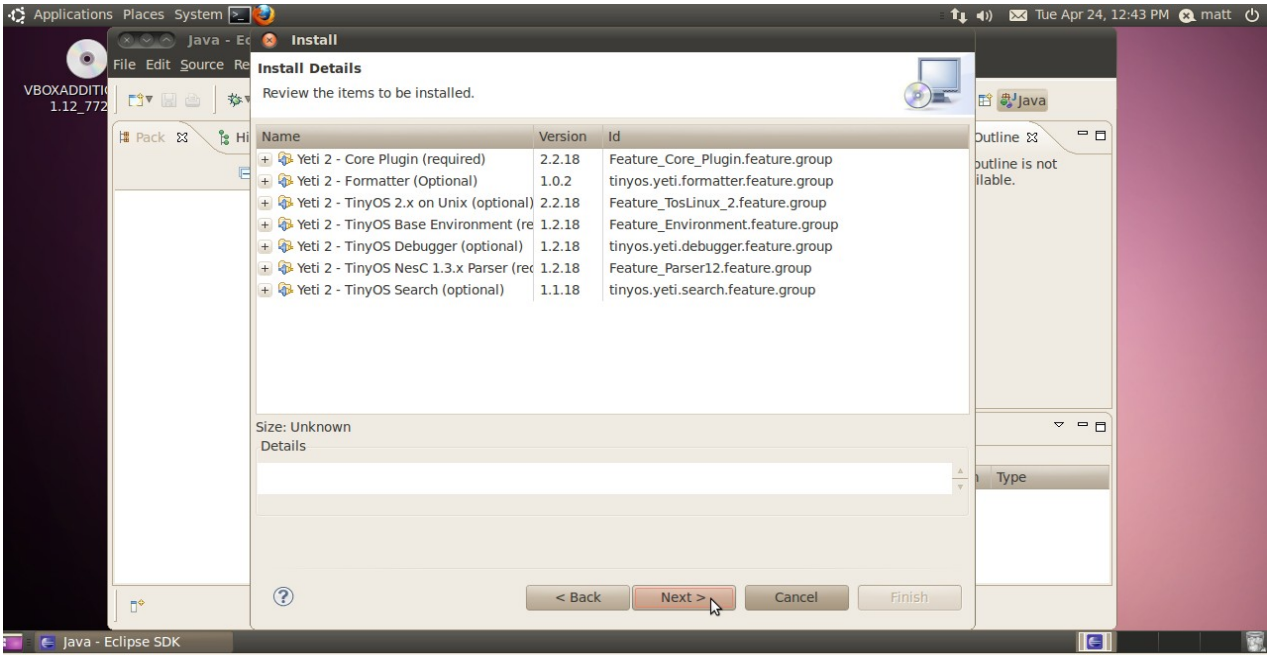
Under "Yeti 2 – Enviroments" install evreything except "Yeti 2 - TinyOS 2.x in Cygwin(optional)1.1.18".

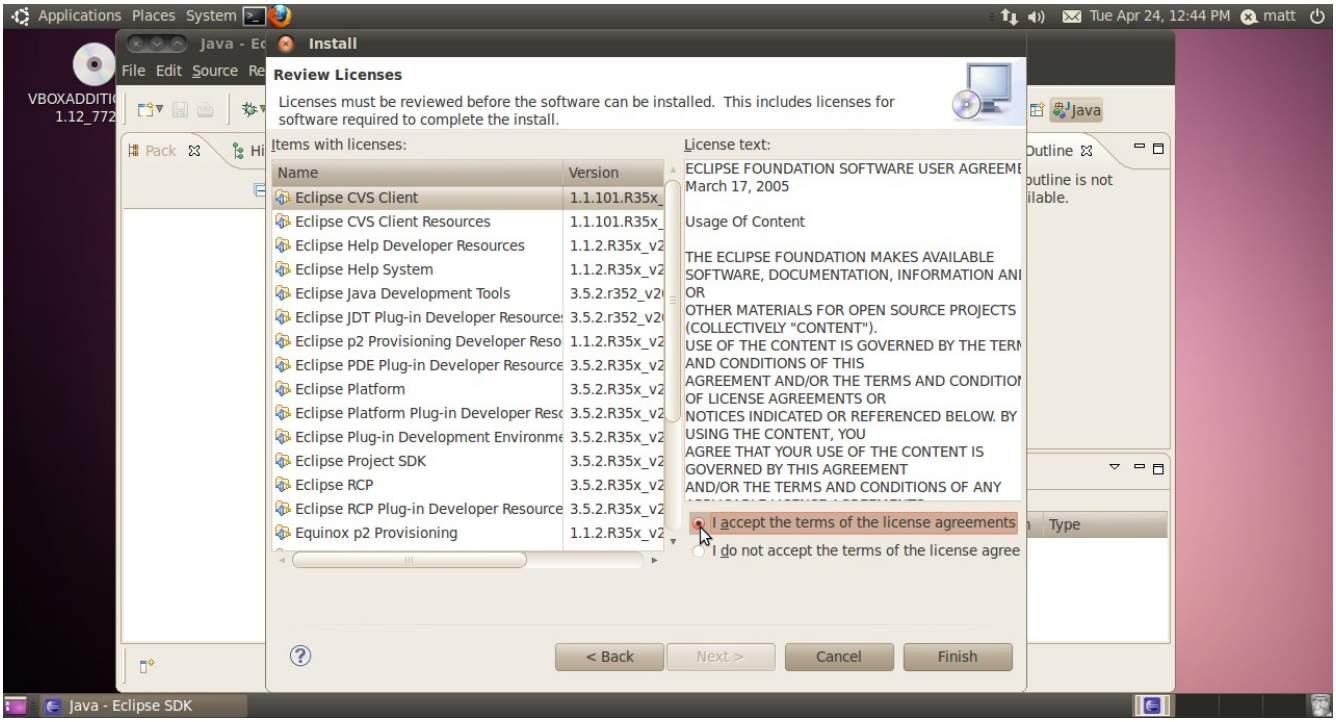


Under "Yeti 2 – Optional" install everything except Yeti 2 - Refactoring (Optional)1.0.1



Click Next and follow the steps:

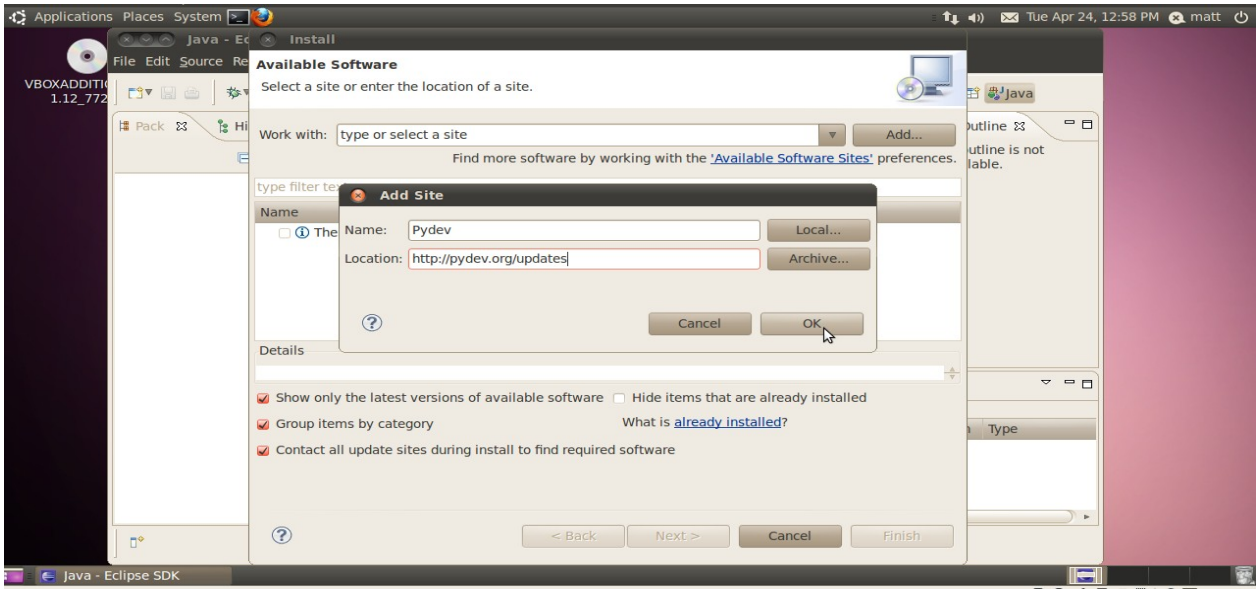




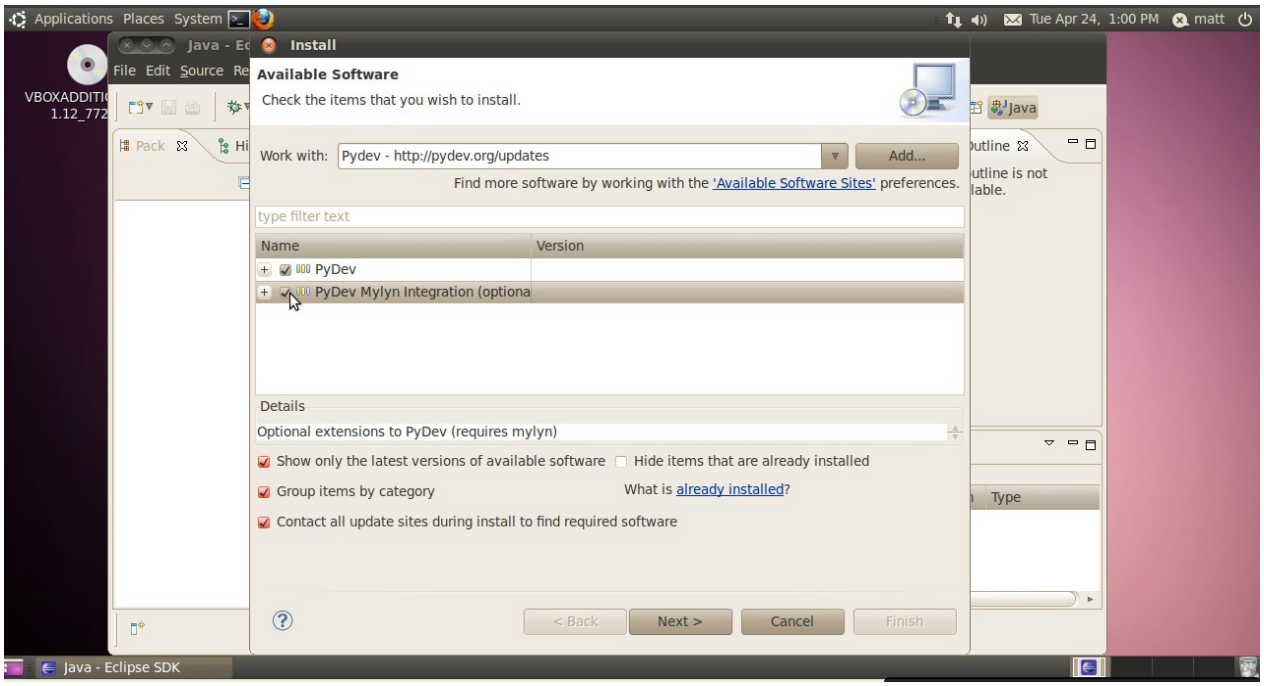
Eclipse will prompt you to restart the program, go ahead and do so.
NOTE: Yeti 2 is no longer supported and will show errors for some standard c functions such as memcpy() even though it compiles and runs properly. This is unavoidable but at the same time it is nicer to have a NesC parser rather than simply relying on the compiler to show you all the errors in your code.

Installing PyDev:

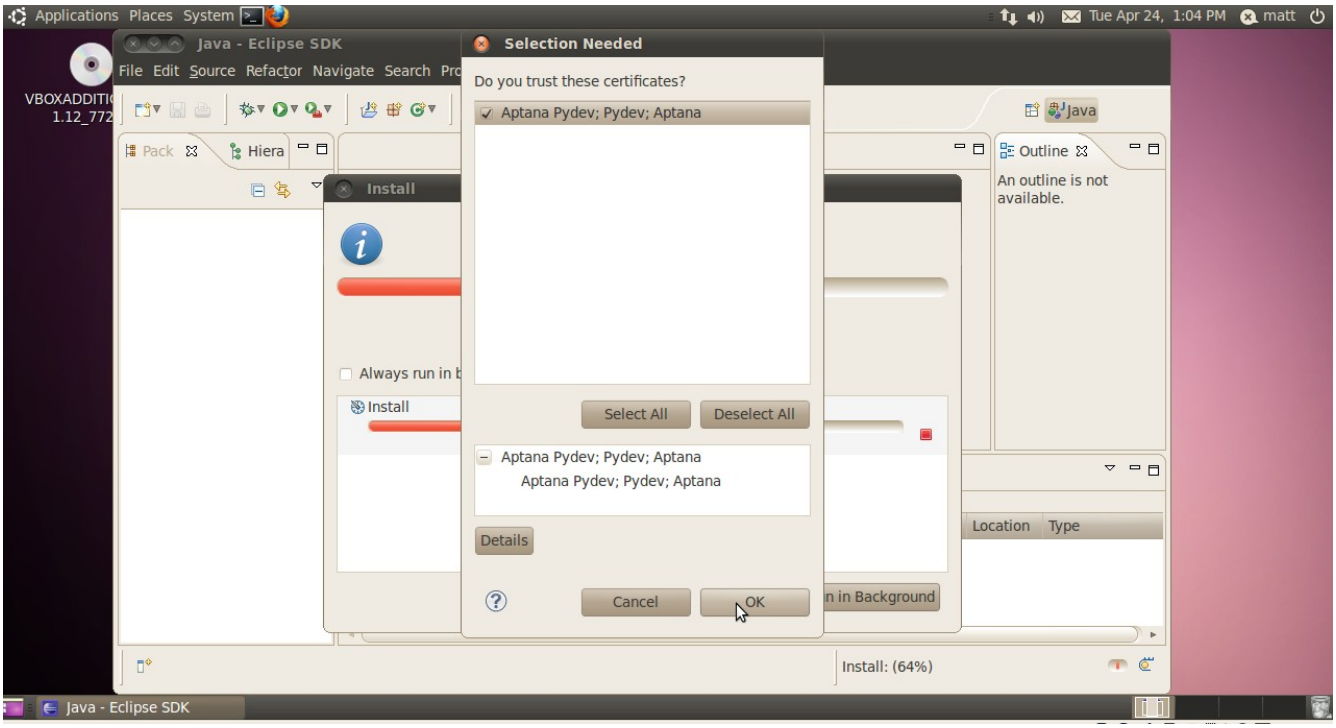
Under the Help tab click “install new software”
Click “ADD” and type in the following address:
http://pydev.org/updates



After a couple seconds 2 options will show up, install both.



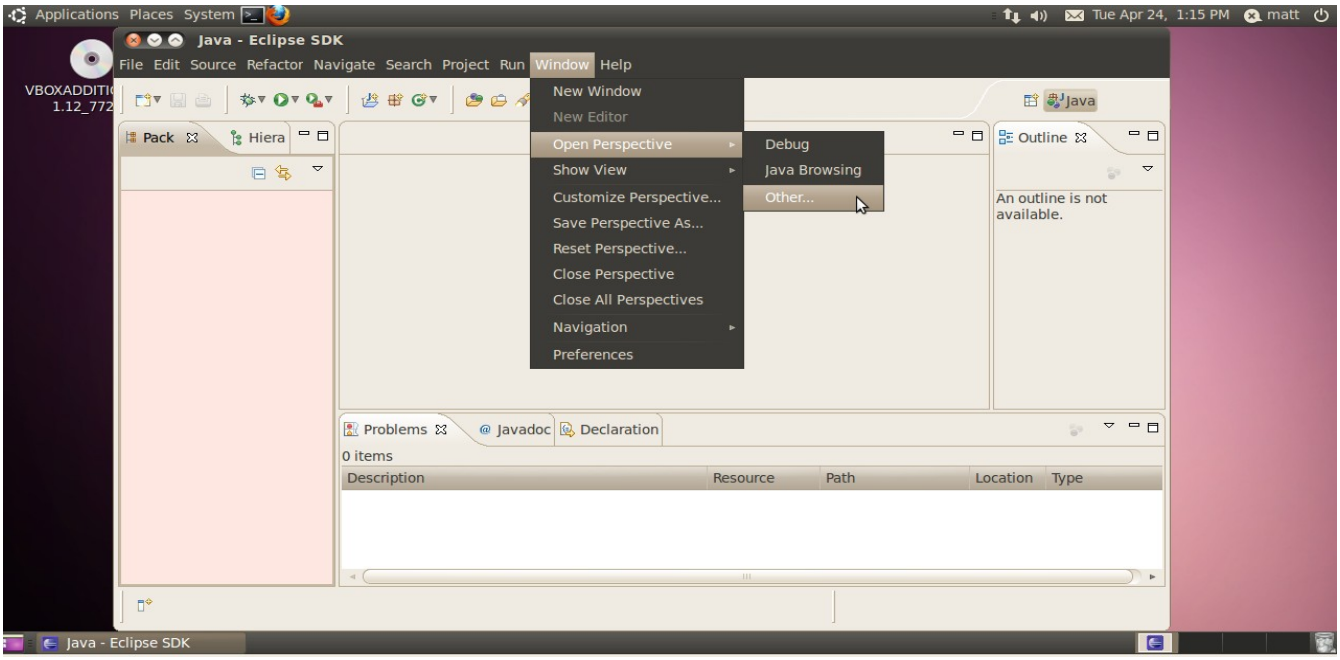
Follow the steps and install pydev, you will be prompted to select a certificate, there is only 1 option, so select it and hit OK.

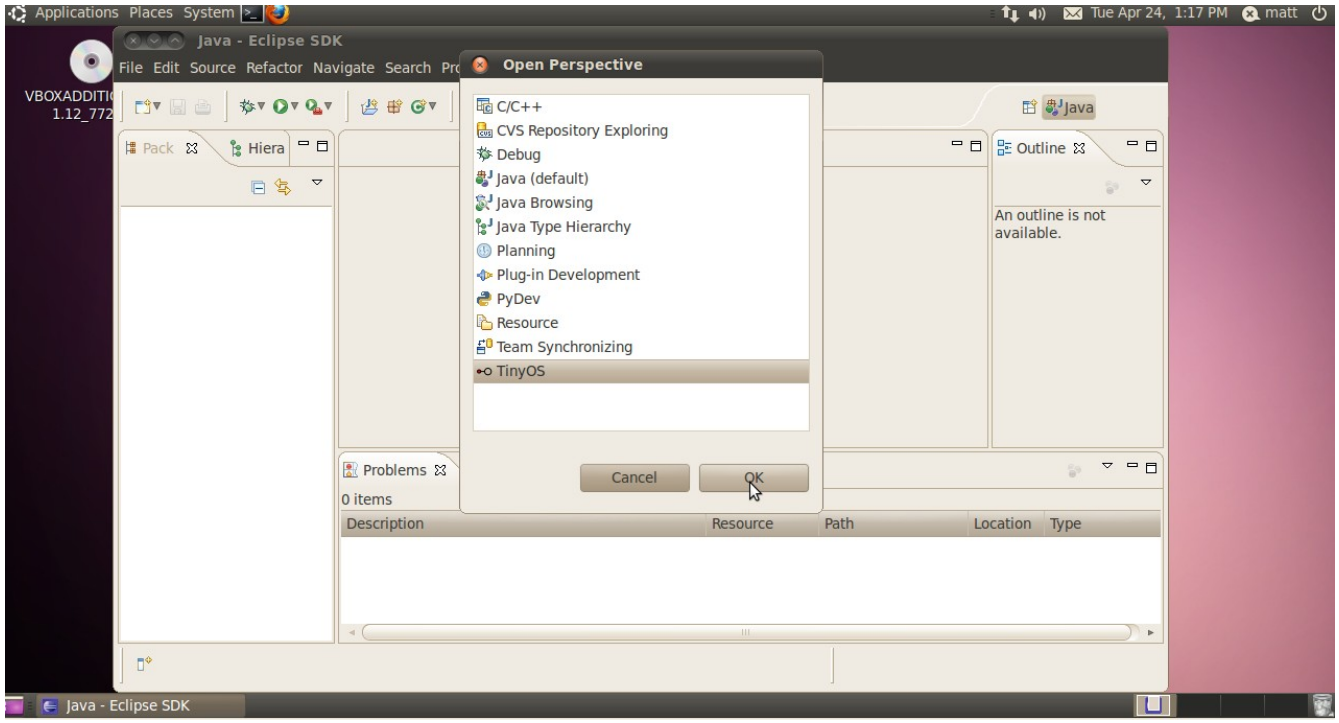


You will be prompted to restart eclipse again, do so. PyDev will not be fully configured until you open a python file within eclipse. At this point it will ask you to configure it at which point simply click “Auto-Config”. This step will be shown below.

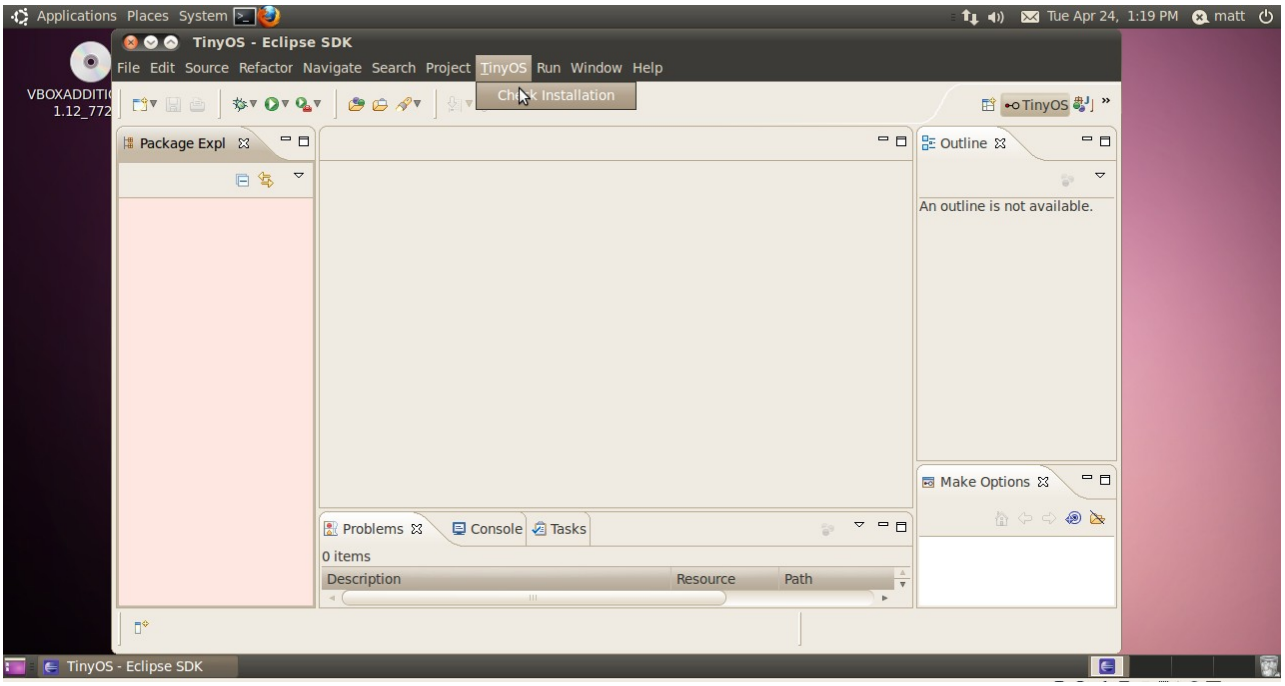
Starting a TinyOS project in eclipse:

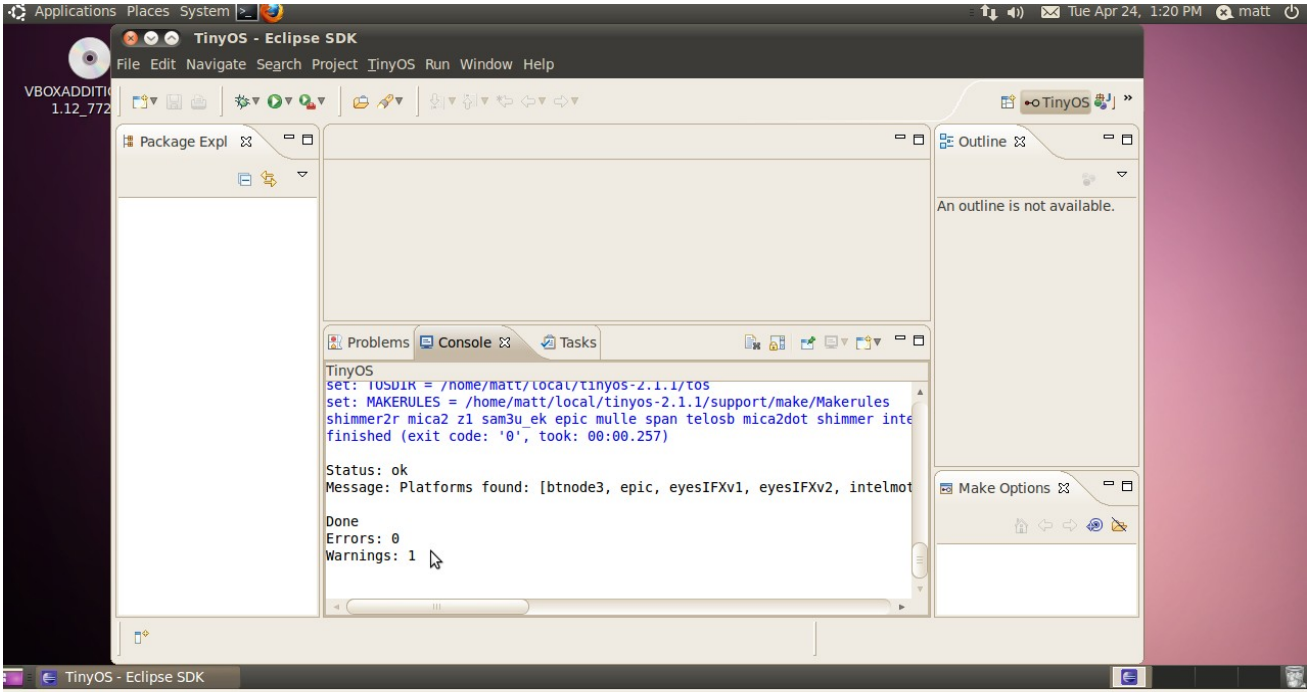
First check your installation of TinyOS in eclipse by going to Window->OpenPerseptive->Other and choosing tinyos in the menu.



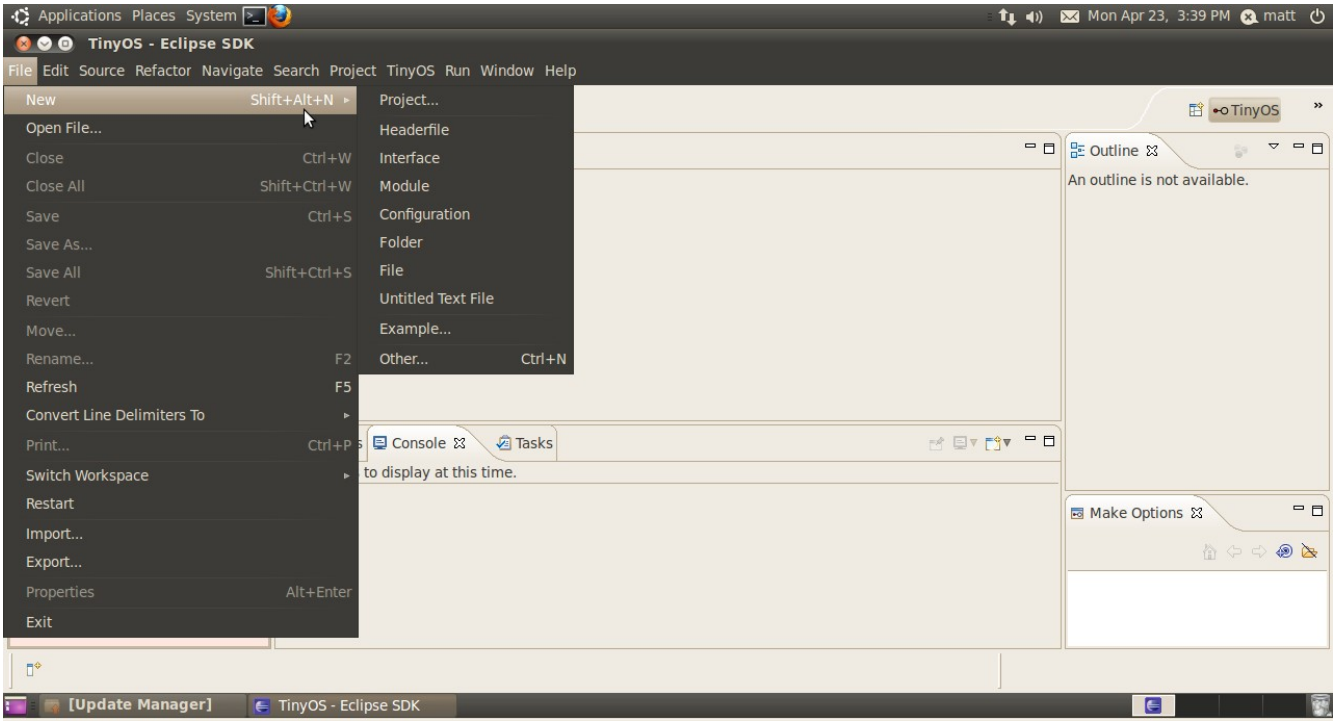


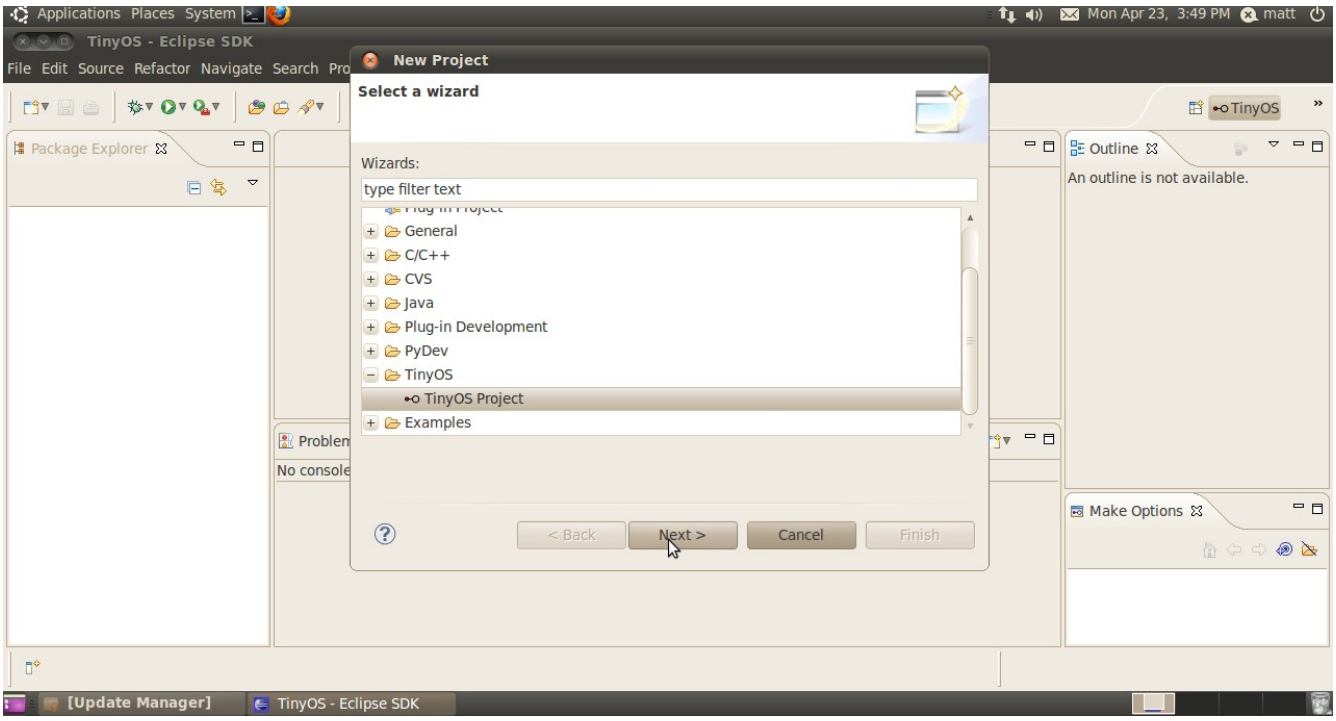
A TinyOS tab will appear between the Project and Run tab, select it and click check installation. If the check reports 0 errors than you have successfully insalled TinyOS.



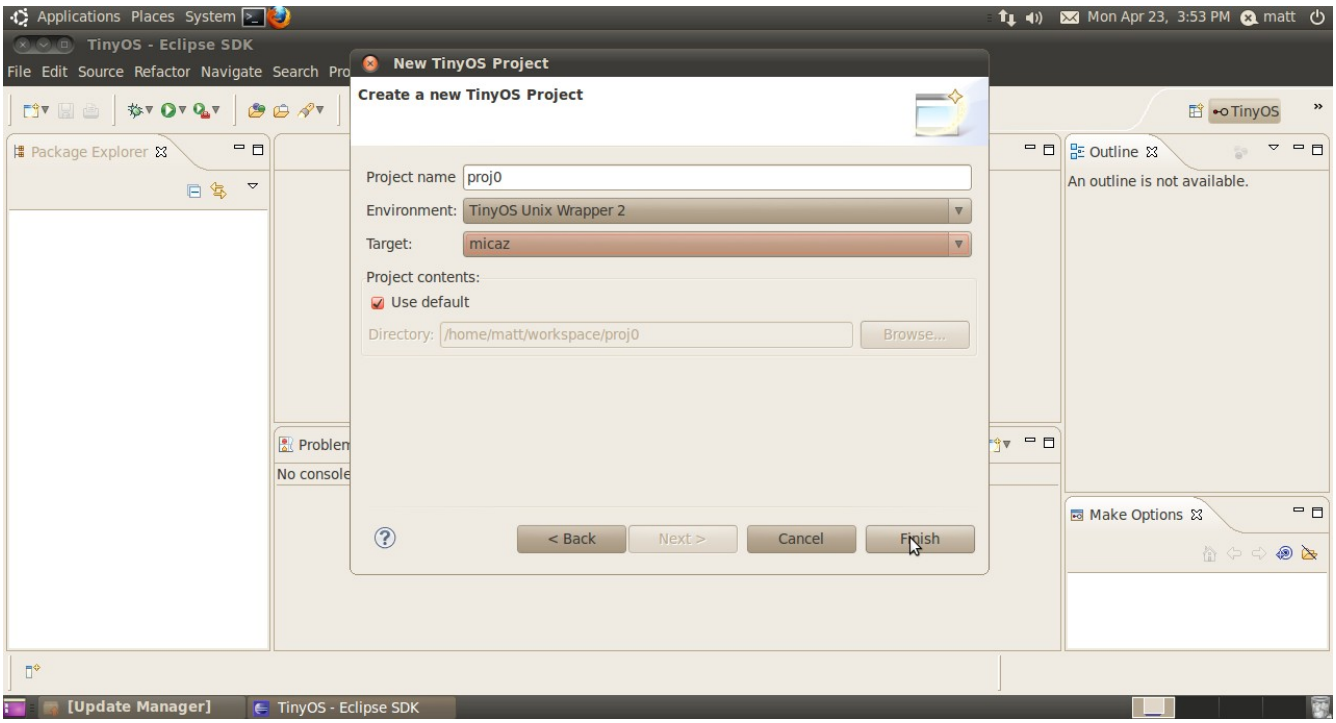


Next, create a new tinyos project:
File->New->Project and select tinyos.





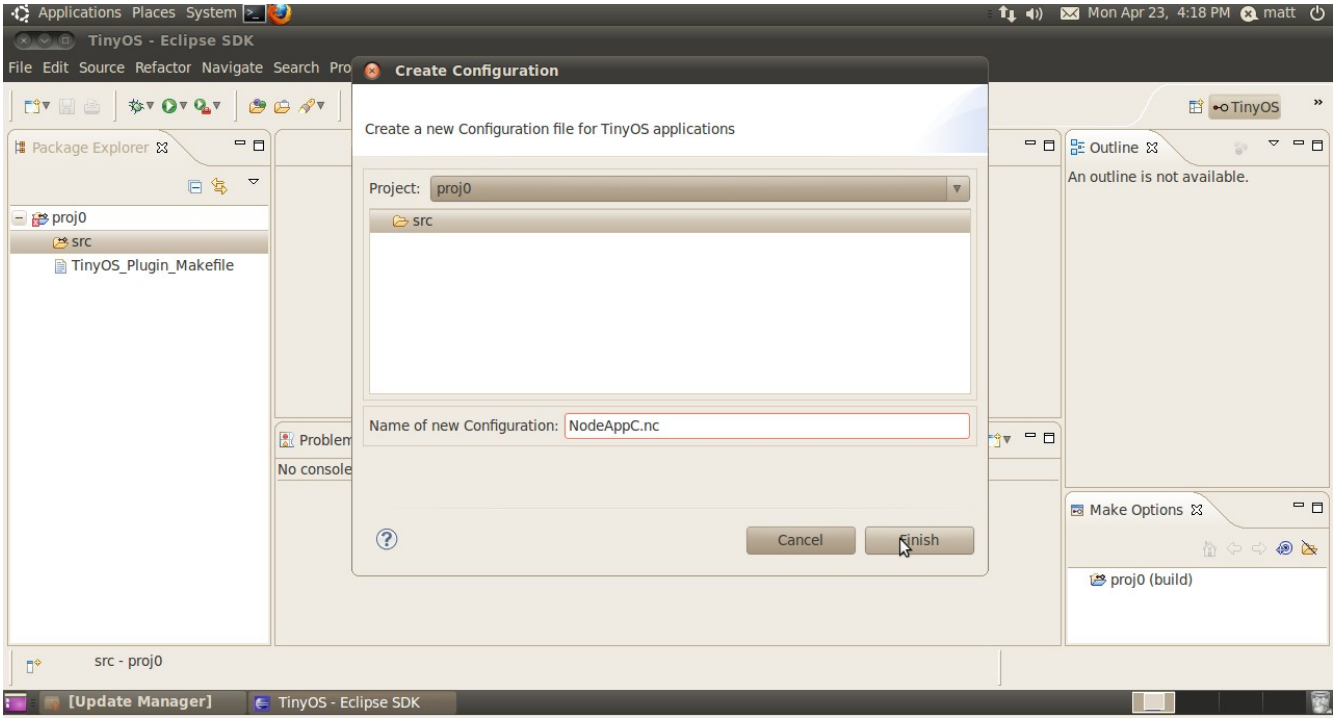
Give the project a name and set the target to micaz.



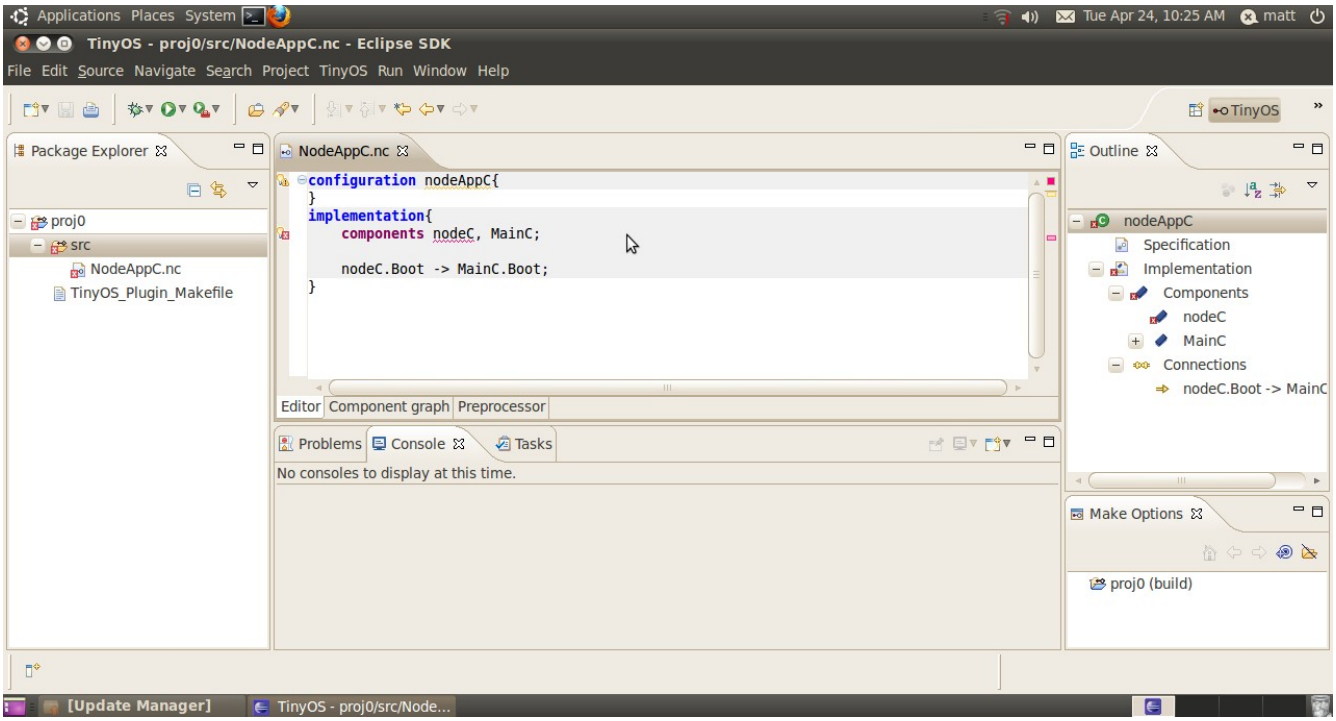
Next we will create the files NodeC.nc, NodeAppC.nc, topo1 and run.py (provided below)

Note: Copy/Paste is a nice feature but eclipse may not recognize symbols in some fonts. You may need to replace quotation marks and other letters/symbols from the code given below or it WILL NOT COMPILE.

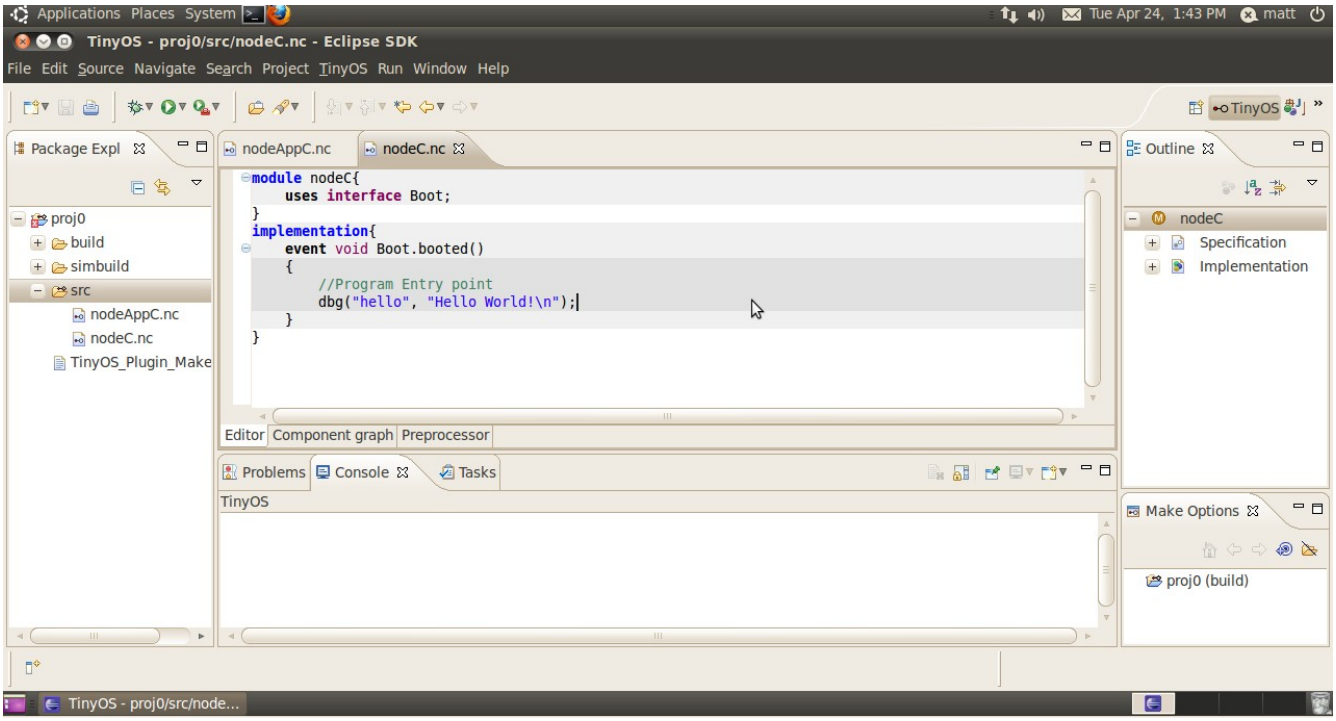
Right click the src folder inside your newly created proj0.
Select New->Configuration and name it “NodeAppC.nc”.



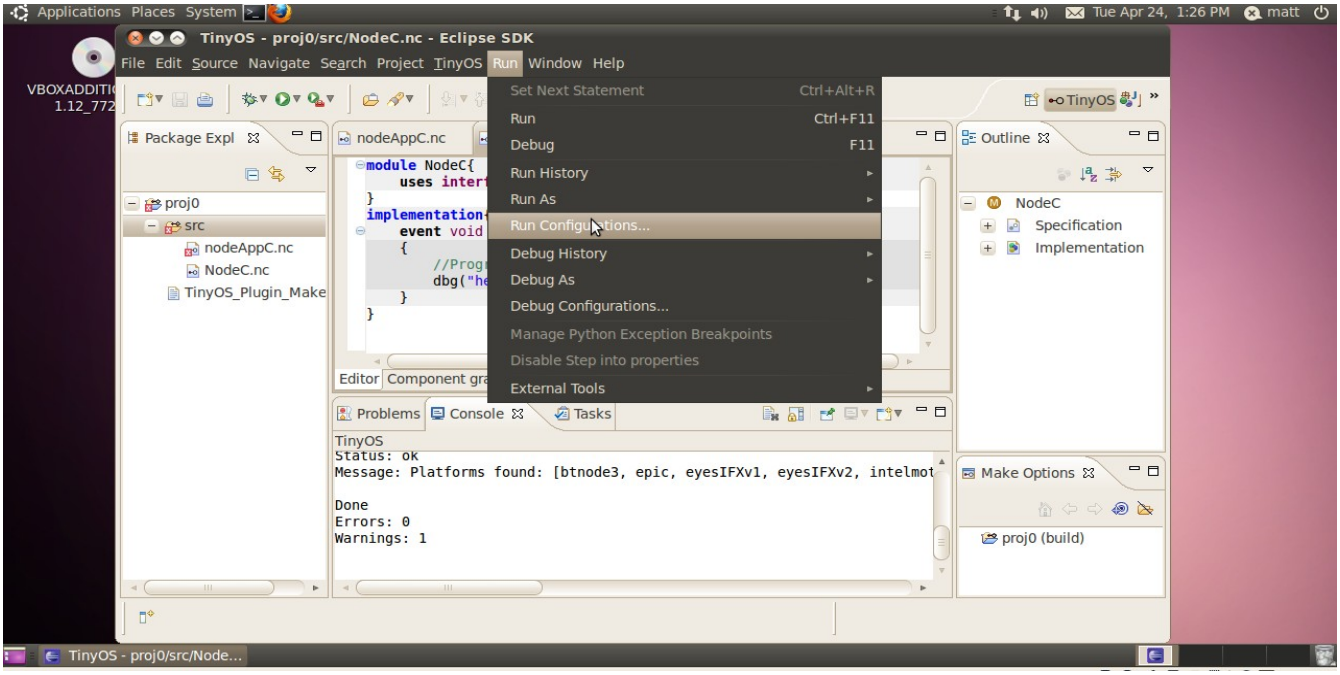
Copy the code below into NodeAppC.nc.

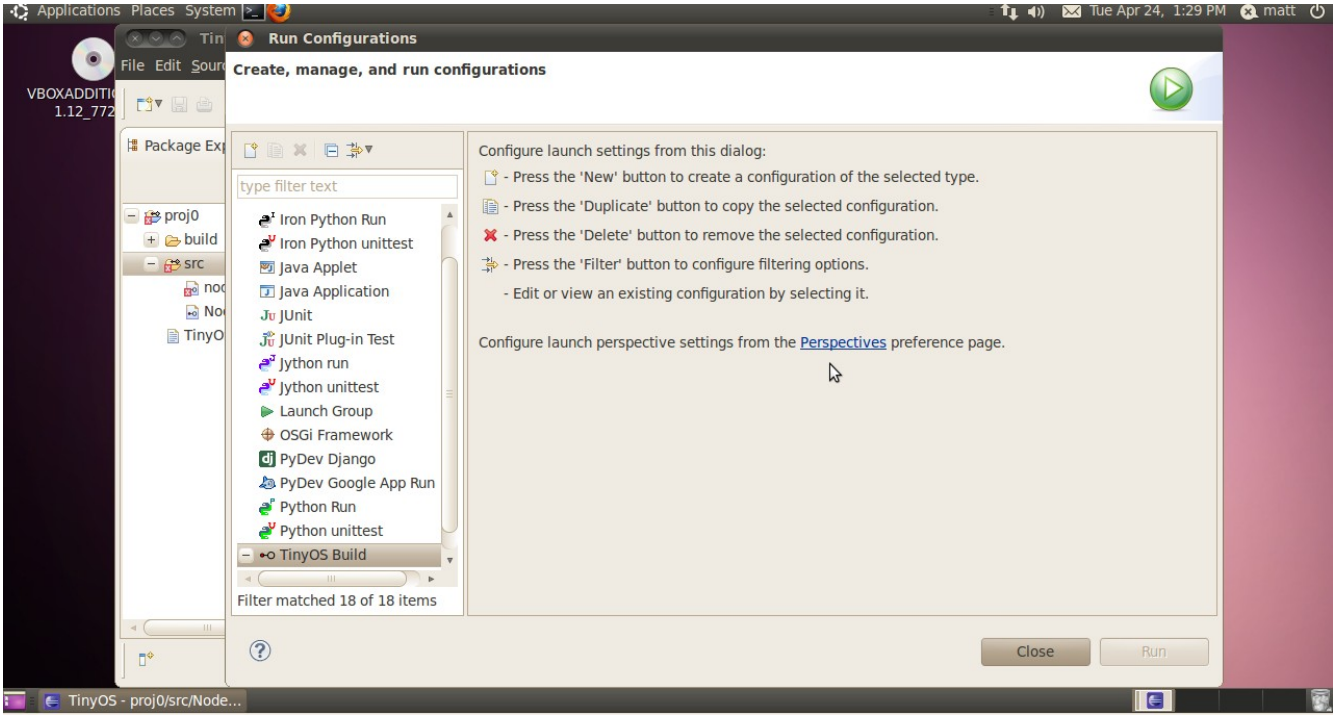


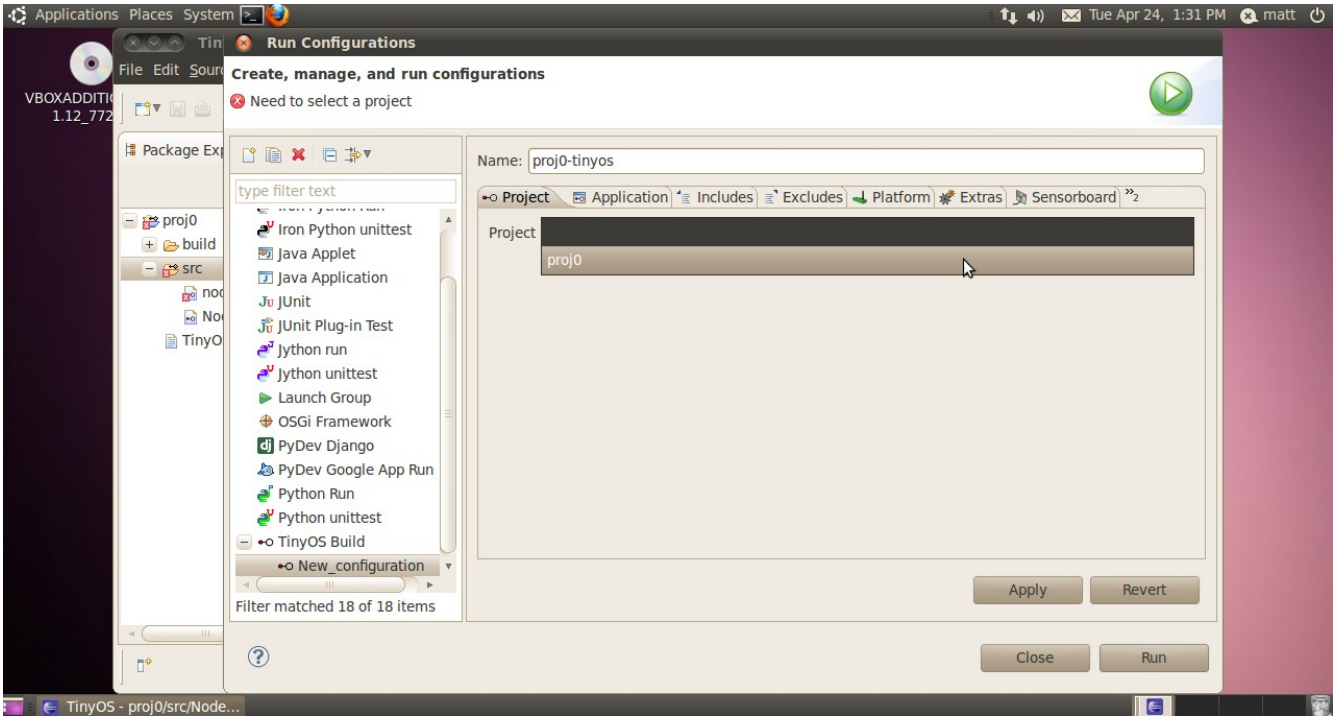
Right click the src folder as you did in the previous step. Go to New->Module and name it NodeC.nc. After filling in the code for NodeC.nc the result should look like this.



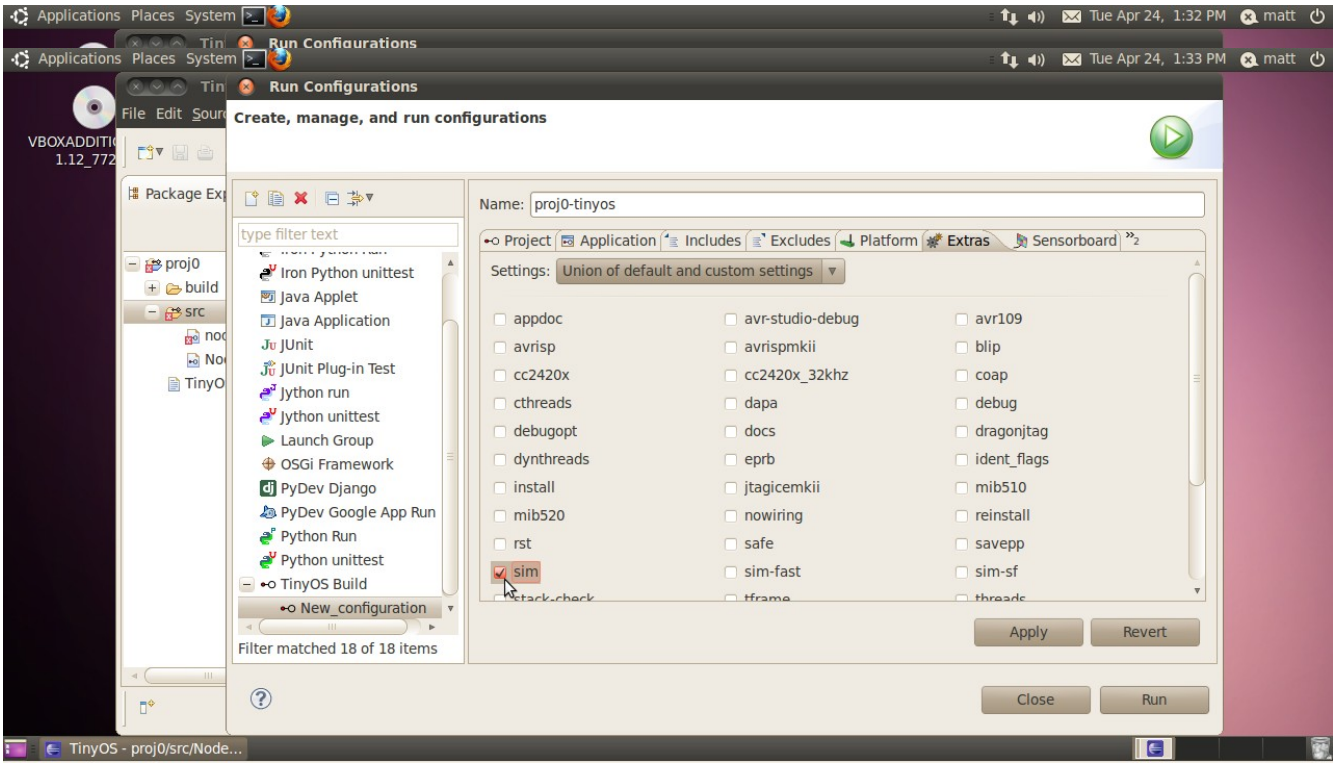
To compile your NesC code into a python script, go to Run->Run Configurations. Double click tinyos build and I suggest giving it a distinct name such as proj0-tinyos (I will refer to it as proj0-tinyos).


Applications Places System
Tue Apr 24, 1:26 PM matt
TinyOS - proj0/src/NodeC.nc - Eclipse SDK
File Edit Source Navigate Search Project TinyOS Run Window Help
Set Next Statement Ctrl+Alt+R
Run Ctrl+F11
Debug F11
Run History
Run As
Run Configurations...
Debug History
Debug As
Debug Configurations...
Manage Python Exception Breakpoints
Disable Step into properties
External Tools
Package Expl
proj0
src
nodeAppC.nc
NodeC.nc
TinyOS_Plugin_Make
nodeAppC.nc
module NodeC{
uses inter
}
implementation
event void
{
dbg("he
}
}
Editor Component gra
Outline
NodeC
Specification
Implementation
Problems Console Tasks
TinyOS
Status: ok
Message: Platforms found: [btnode3, epic, eyesIFXv1, eyesIFXv2, intelmot
Done
Errors: 0
Warnings: 1
Make Options
proj0 (build)
TinyOS - proj0/src/Node...


Applications Places System
Tue Apr 24, 1:29 PM matt
Run Configurations
Create, manage, and run configurations
type filter text
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
JUnit Plug-in Test
Jython run
Jython unittest
Launch Group
OSGi Framework
PyDev Django
PyDev Google App Run
Python Run
Python unittest
TinyOS Build
Filter matched 18 of 18 items
Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
Configure launch perspective settings from the Perspectives preference page.
Close
Run
TinyOS - proj0/src/Node...

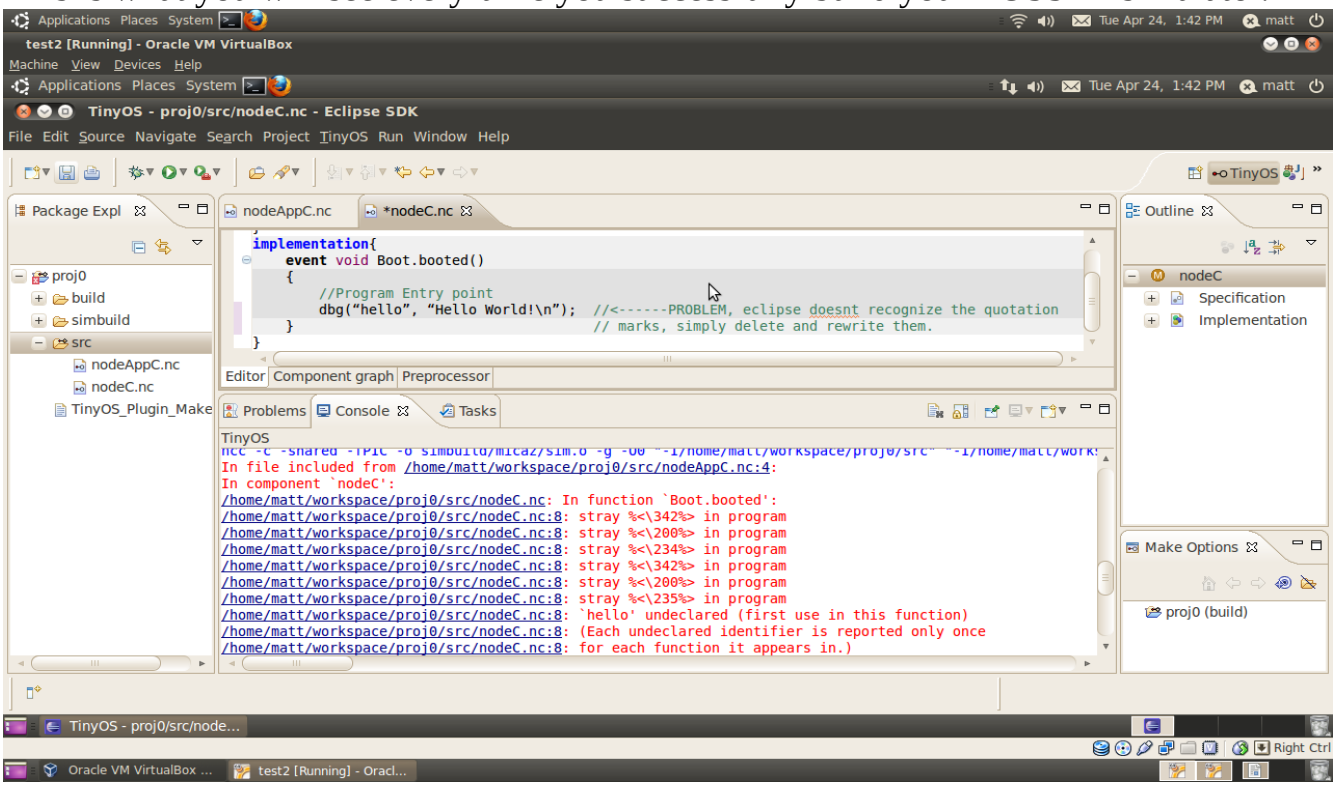


Select the project you created in the project menu. In the application tab select custom settings, open the source folder and then select nodeAppC.nc. Under the extras tab select “sim”.

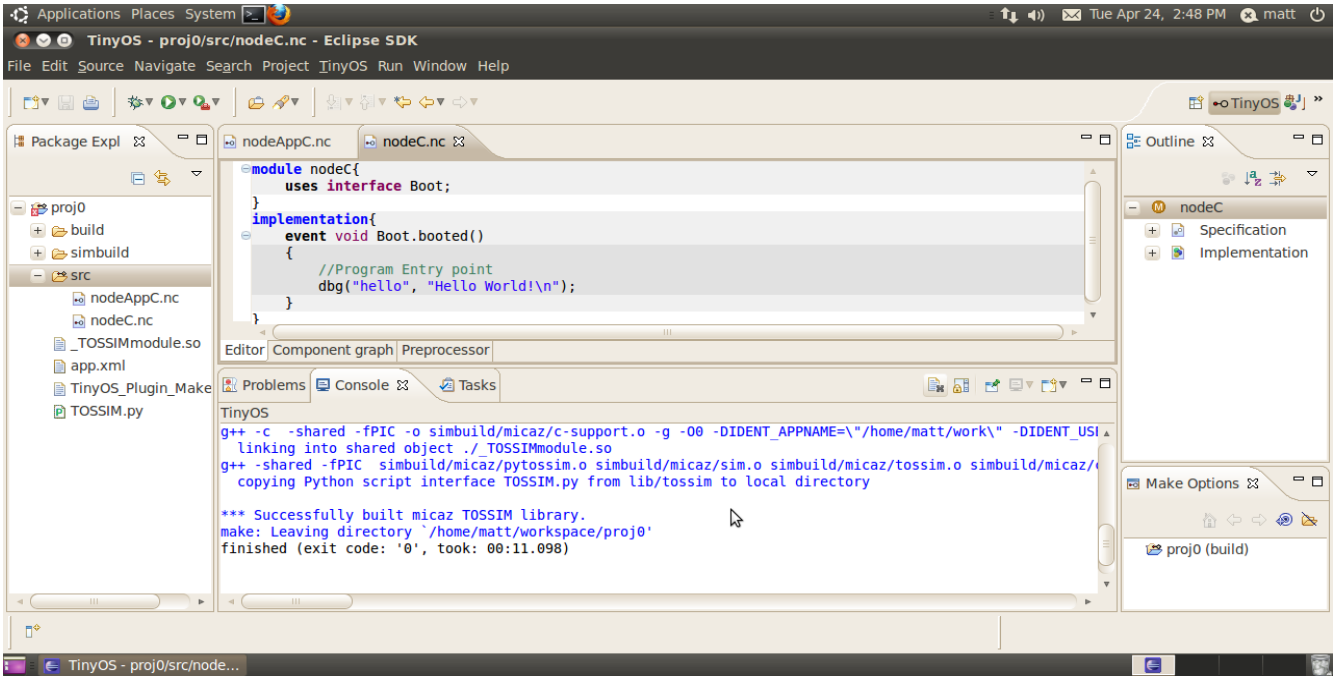


Then hit apply and run. The image below shows a copy/paste error were the quotation marks are not recognized.

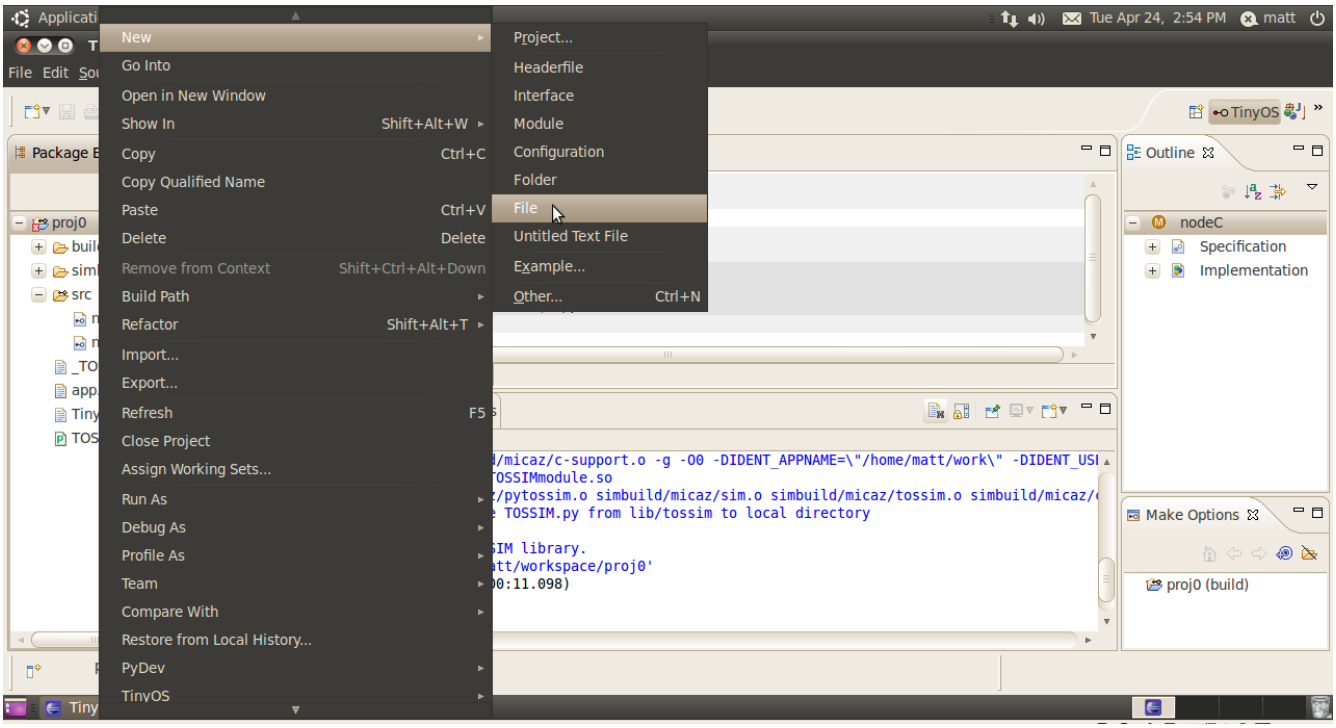
This is what you will see every-time you successfully build your TOSSIM simulator.

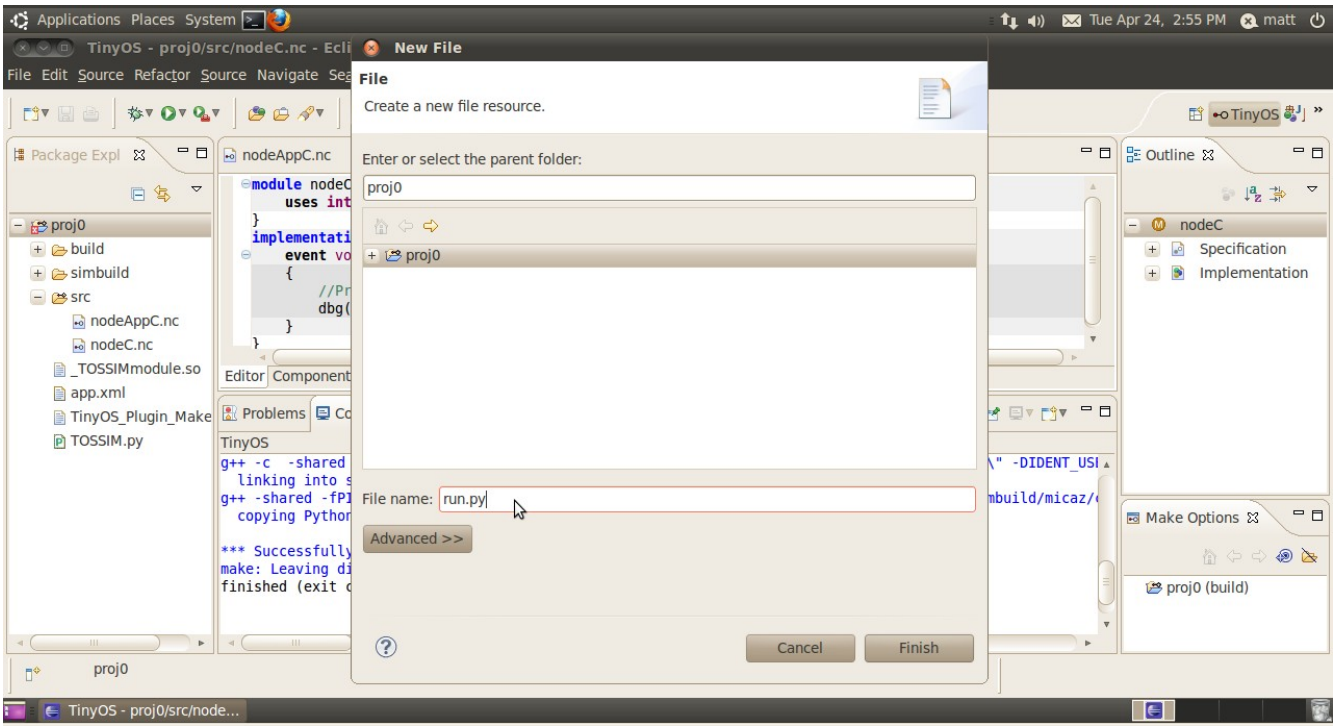


The simulator must be rebuilt any time a change is made to the NesC or C code.

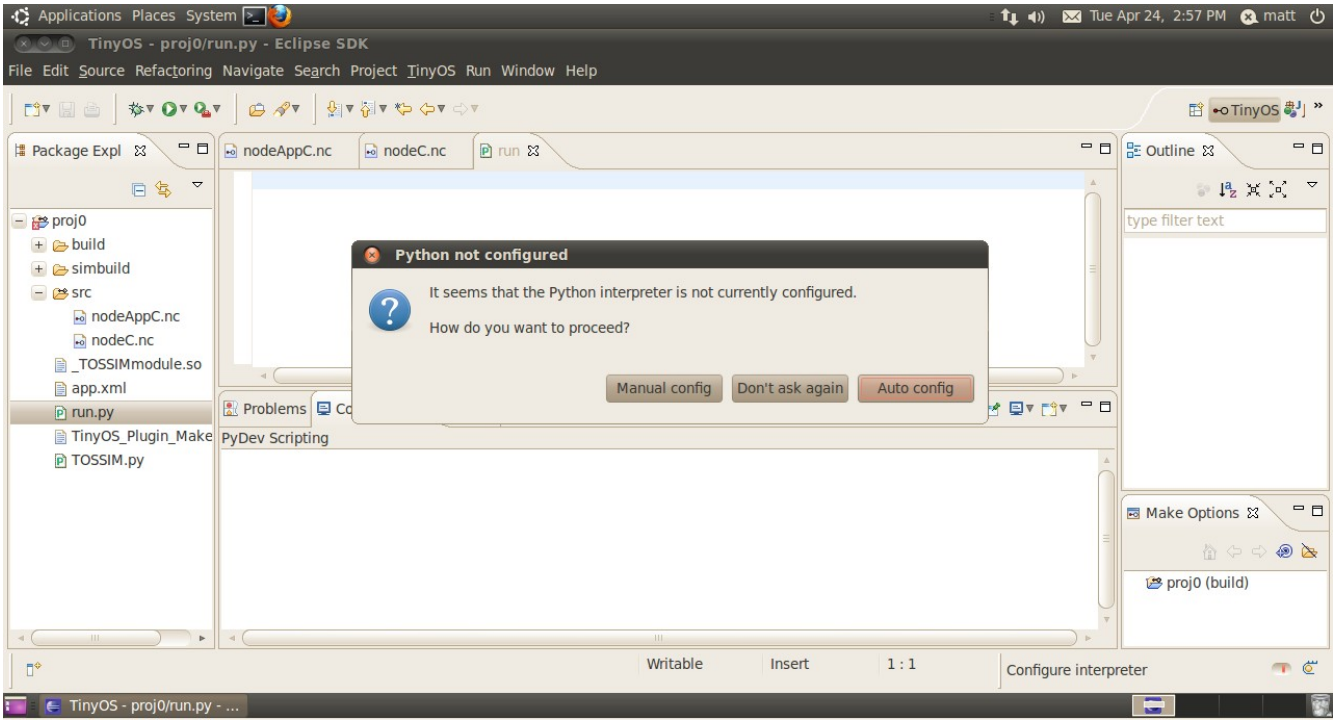


We now need to finish the configuration of our python parser in eclipse in order to be able to run the TOSSIM simulator. Right click proj0 in the left navigation bar, go to New->File and label this new file run.py.

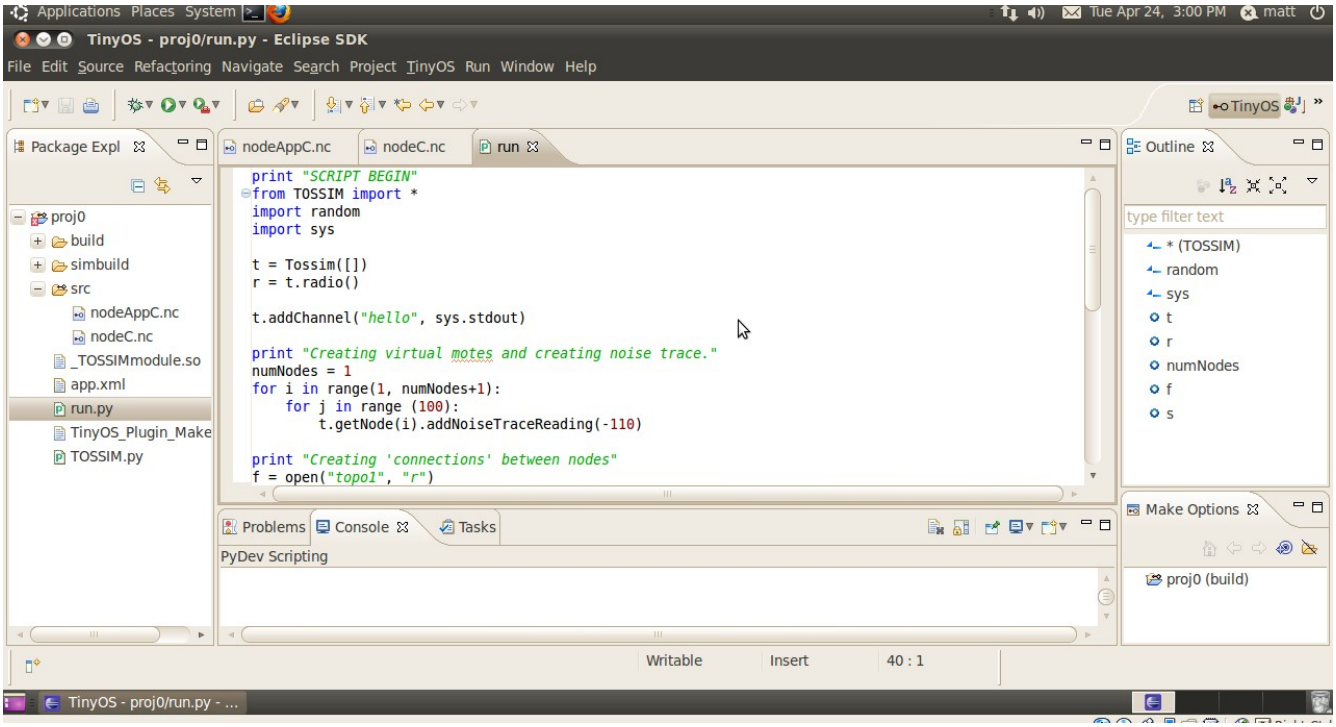




Upon clicking finish eclipse will open run.py and will ask you how to configure python, simply click "Auto-Config" and OK each time a new prompt appears.

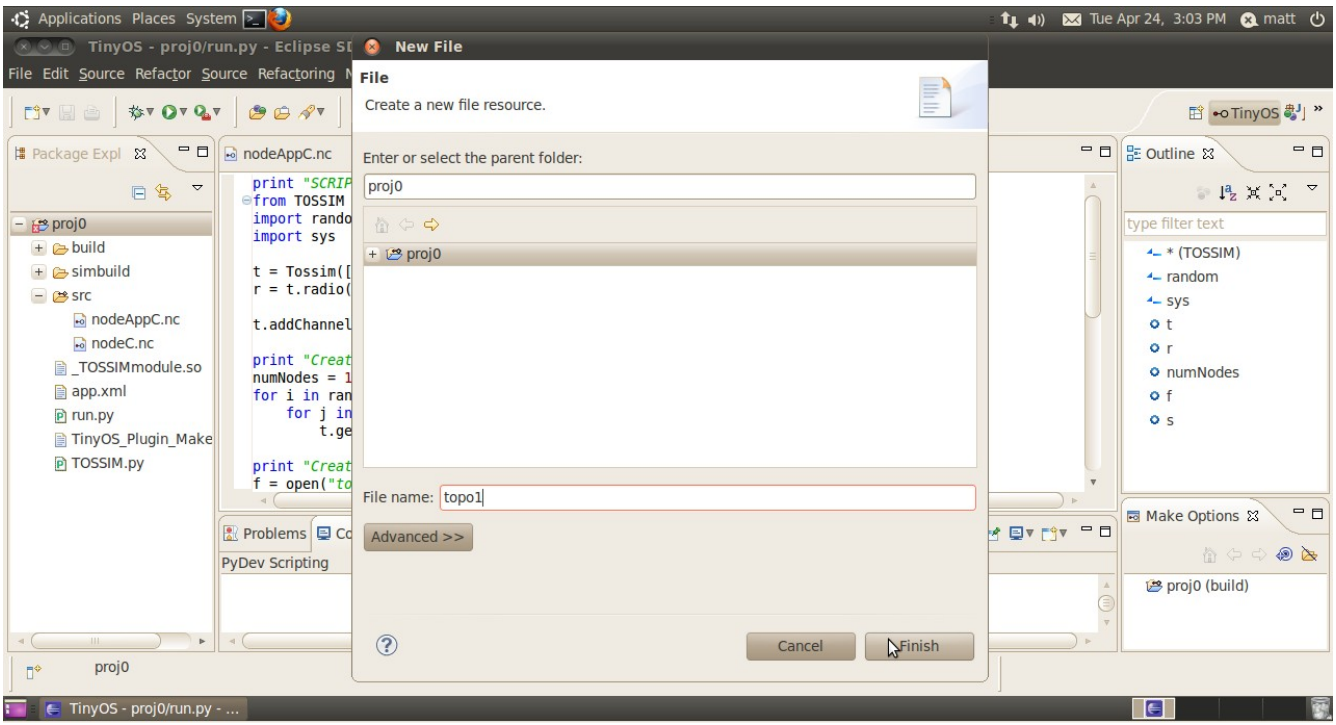


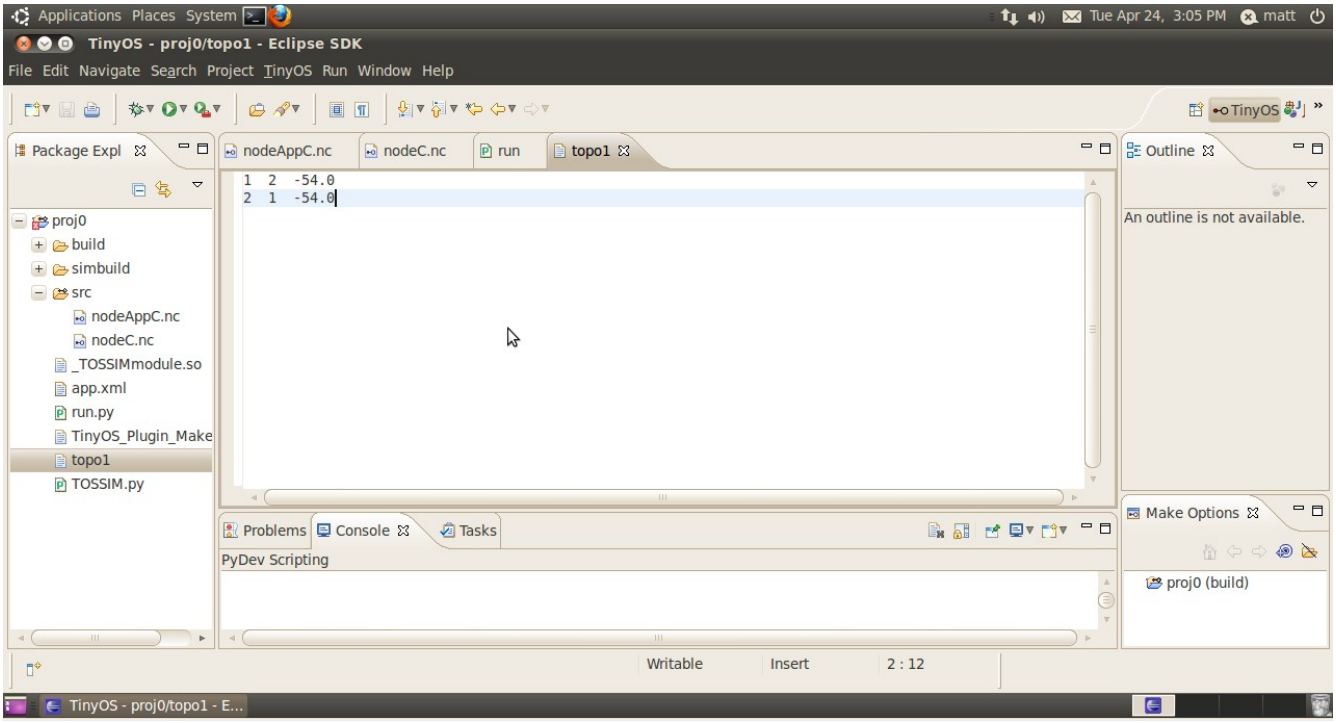
Now fill in the run.py file with the code given below.



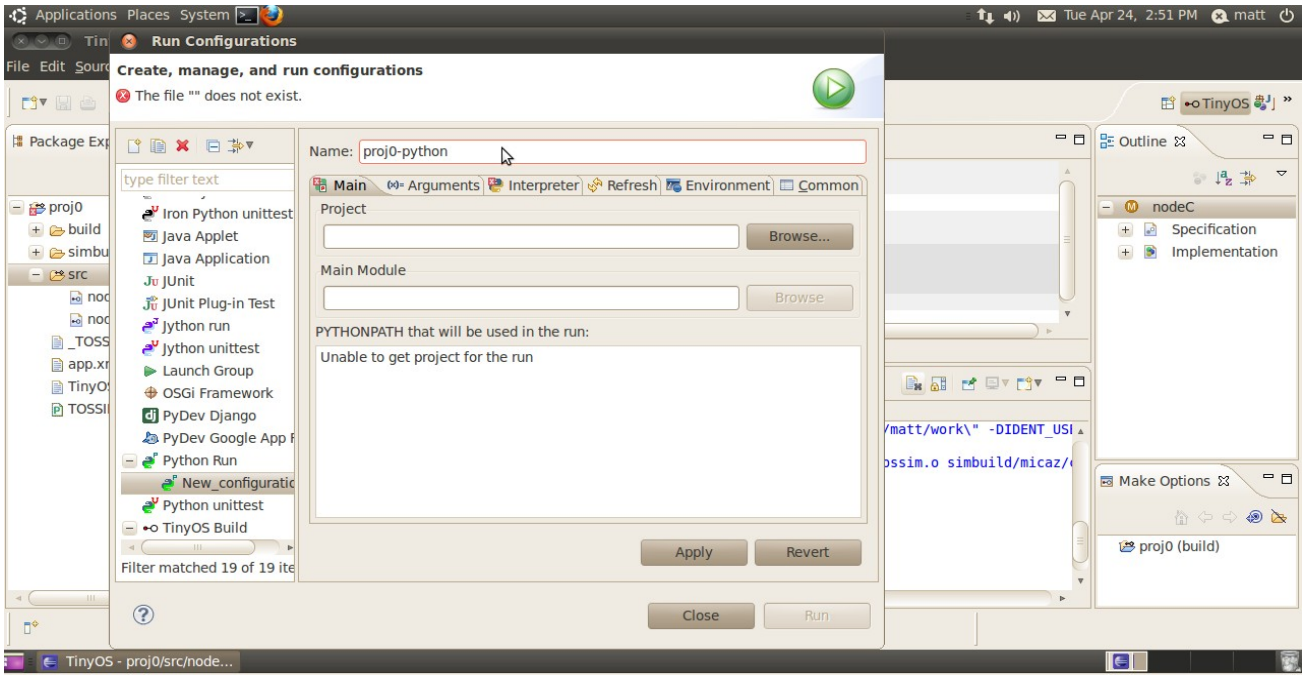
Again right click proj0 in the navigator on the left side of your screen and select

New->File and label it topo1, filling it with the code provided below.

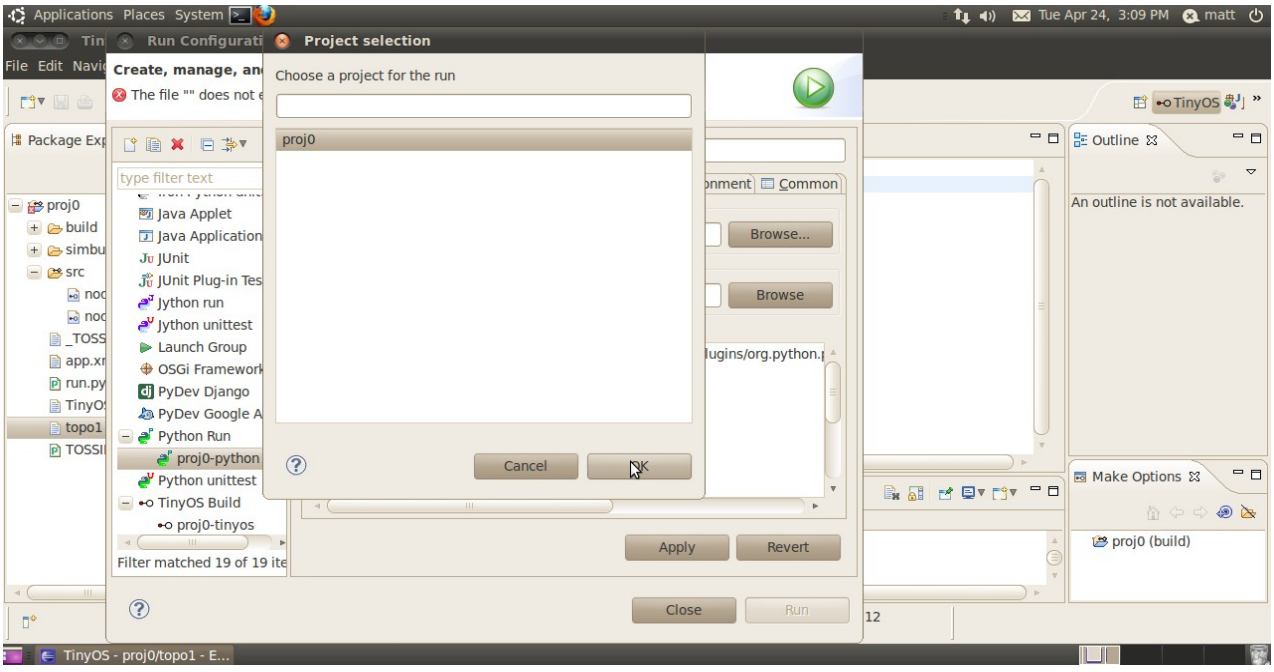


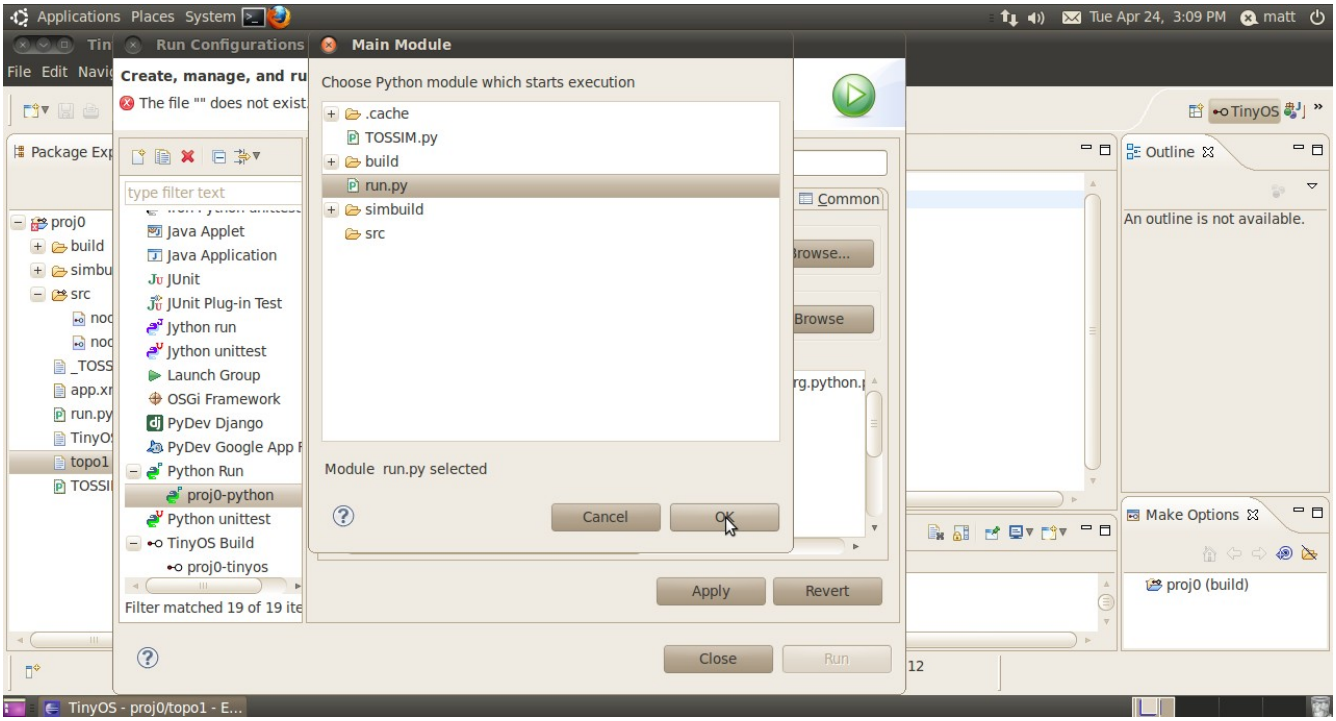


Finally, to run the simulator, go to the Run menu and select Run->Run Configurations. Select “Python Run” and I suggest giving it a distinct name such as proj0-python (I will refer to it as proj0-tinyos).

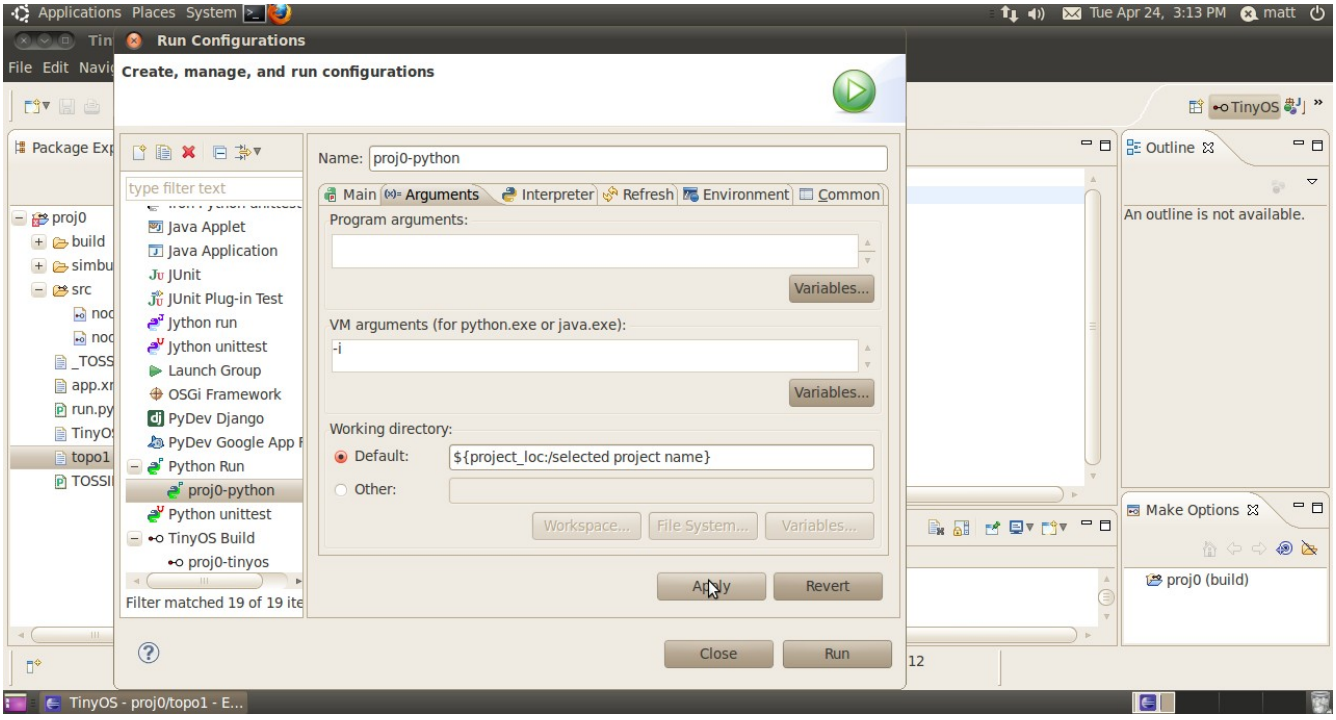


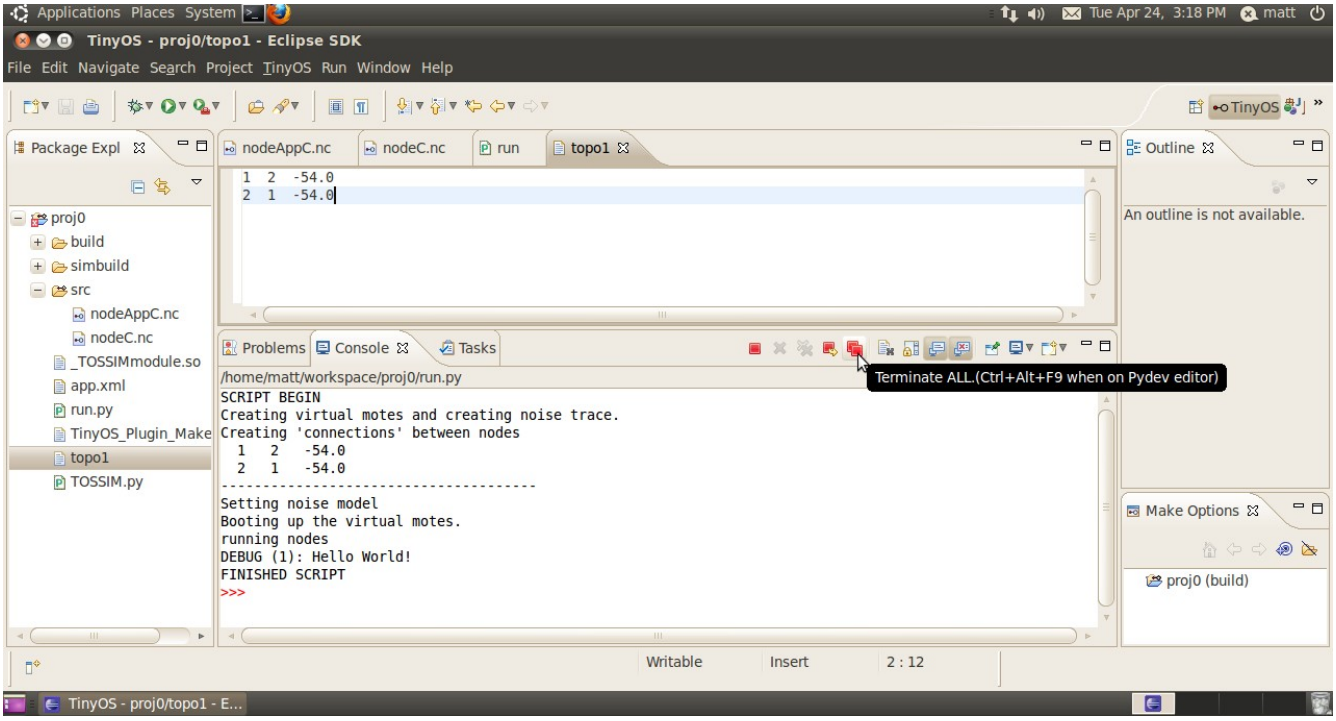
Click Browse next to the Project field and select your proj0 as the project and similarly select run.py as your module.





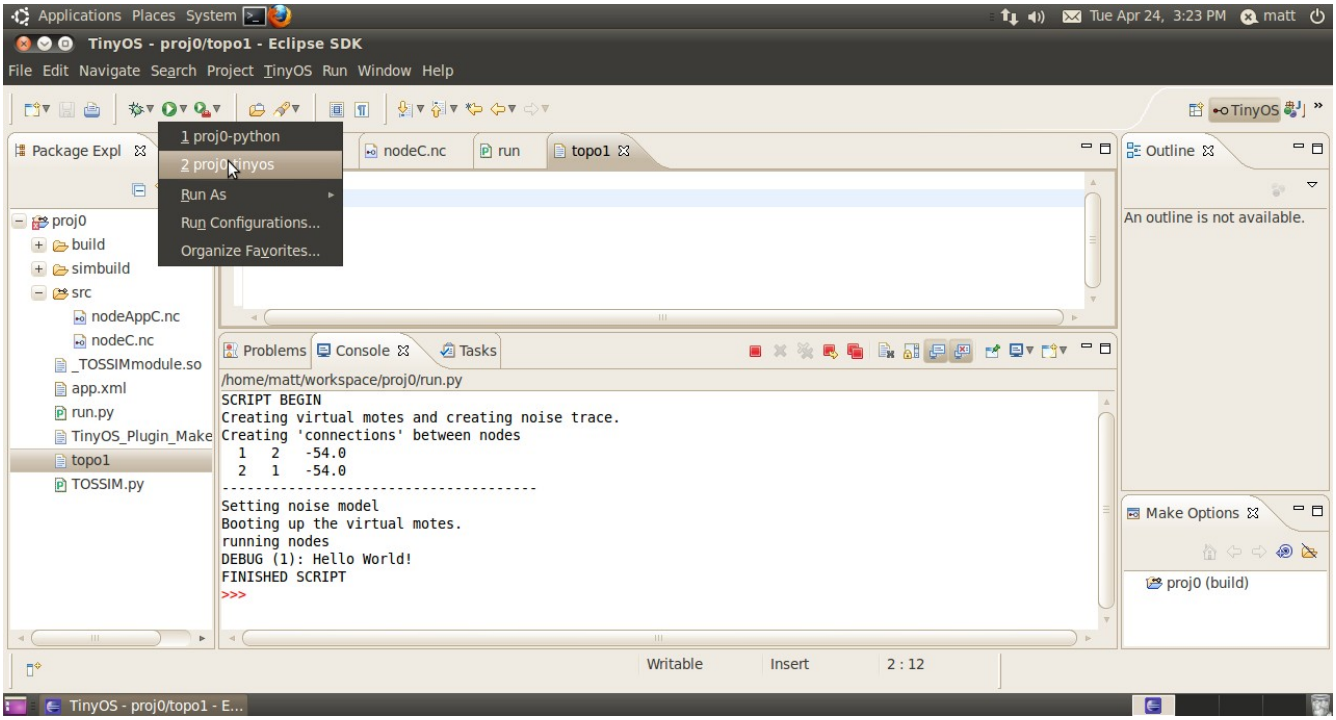
In the argument tab add “-i” in the VM argument box, then hit apply and run.



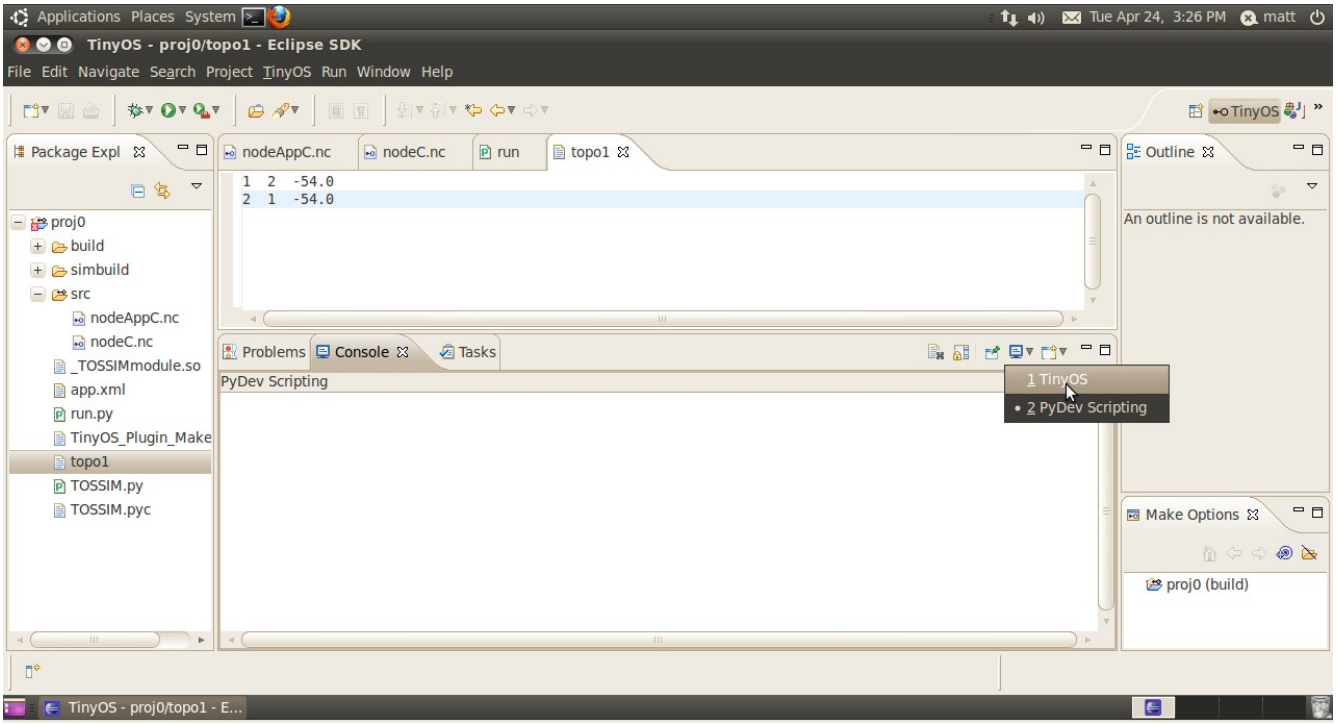


Congratulations, you have successfully run Hello World in TinyOS if you have the same output as above! Notice the >>> at the bottom of the console window, this is the same patform you would get if you typed python into the terminal and will allow you to turn on/off the simulated motes as well as inject packets into the system. Note that you should always stop the simulator by clicking the button the cursor is hovering over in the image above. Running the simulator again will not stop the current simulator but instead start a new one. If enough of these are running at the same time it noticeably hinders performance of all programs on your computer.

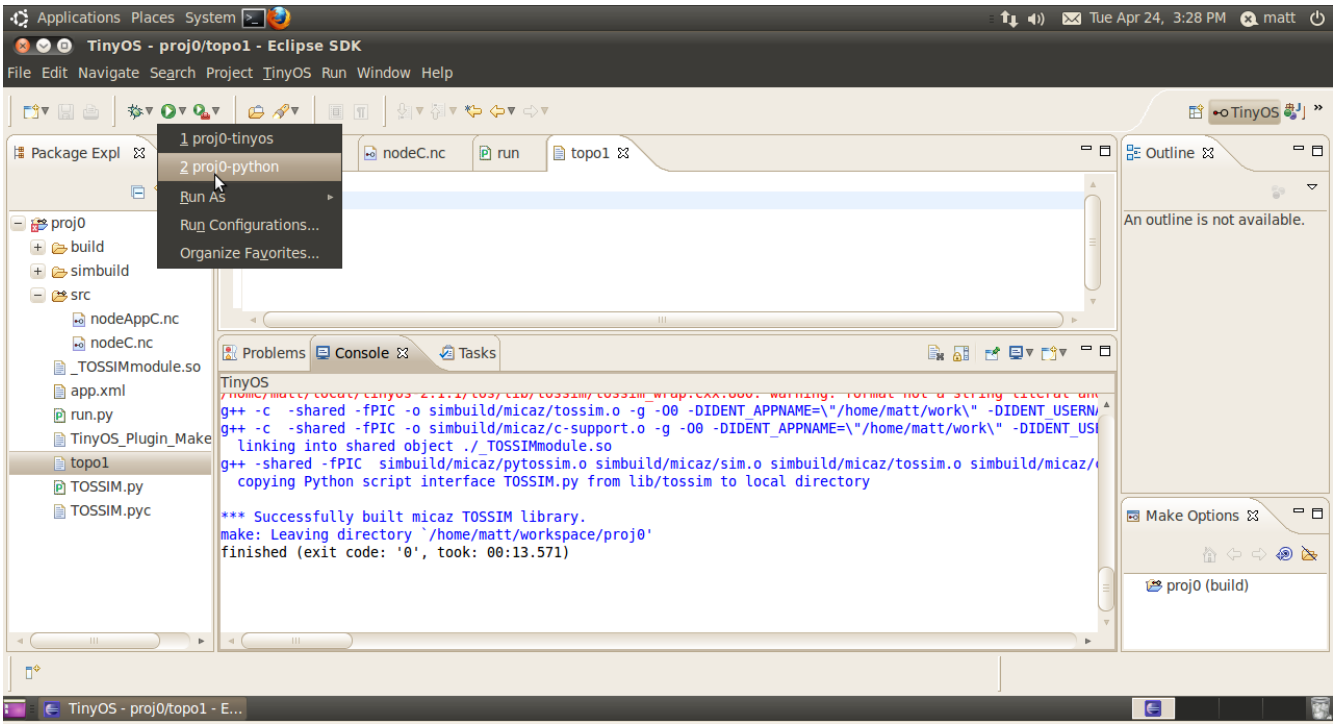
To compile the TOSSIM simulator again, all that is needed is to select the drop down menu of the green Run button at the top of eclipse and select proj0-tinyos.



If the console below does not automatically toggle to show the results of the simulator compilation, click the Select Console button and choose TinyOS.



Likewise to run the simulator again, select the drop down menu for the run button at the top of eclipse and select proj0-python.



Conclusion:

It is highly recommended that you begin looking into how packets are sent, how packets are injected, how to include timers and how to post a task, you will be using these in future projects. In addition, it is good to keep in mind that TOSSIM simulates collision of packets as well as having the ability to turn all simulated motes on at the same exact time. This means packets sent by the motes on specific time intervals will always collide and never reach another mote. To counter this add an amount of randomness to what the time interval is for each mote. When facing any issues with TinyOS or TOSSIM please look to the F.A.Q. provided, the tutorials on the wiki (links provided below), Google as well as the TA.

LINKS:

TinyOS in general:
http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials
TOSSIM specifically:
http://docs.tinyos.net/tinywiki/index.php/TOSSIM

```
//**********nodeAppC.nc**********//
configuration nodeAppC{
}
implementation{
	components nodeC, MainC;

	nodeC.Boot -> MainC.Boot;
}
//**********END**********//

//**********nodeC.nc**********//
module nodeC{
	uses interface Boot;
}
implementation{
	event void Boot.booted()
	{
		//Program Entry point
```

```
                dbg(“hello”, “Hello World!\n”);
        }
}
//*******************END*******************//

//*******************run.py*******************//
print "SCRIPT BEGIN"
from TOSSIM import *
import random
import sys

t = Tossim([])
r = t.radio()

t.addChannel("hello", sys.stdout)

print "Creating virtual motes and creating noise trace."
numNodes = 1
for i in range(1, numNodes+1):
    for j in range (100):
        t.getNode(i).addNoiseTraceReading(-110)

print "Creating 'connections' between nodes"
f = open("topo1", "r")
for line in f:
   s = line.split()
   if s:
         print " ", s[0], " ", s[1], " ", s[2];
         r.add(int(s[0]), int(s[1]), float(s[2]))
print "-------------------------------------"

print "Setting noise model"
for i in range(1, numNodes+1):
    t.getNode(i).createNoiseModel()

print "Booting up the virtual motes."
#Random boot times are very important, otherwise packet collision is unavoidable
for i in range(1, numNodes+1):
```

```
    t.getNode(i).bootAtTime(random.choice(range(1, 250, 1)))

print "running nodes"
for i in range(1000):
    t.runNextEvent()

print "FINISHED SCRIPT"
```

```
//**********************END**********************//

//**********************topo1**********************//
1  2  -54.0
2  1  -54.0
//**********************END**********************//
```