# How To Install Tiny OS Development Environment

Host environment: Ubuntu

1. 1 Make a fresh system

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

$ sudo apt-get clean

$ sudo apt-get autoremove

1. 2 Install the key in the local apt keyring for tinyprod

$ gpg --keyserver keyserver.ubuntu.com --recv-keys 34EC655A

$ gpg -a --export 34EC655A | sudo apt-key add -

1. 3 Add repository for tinyprod

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

$ sudo echo "deb http://tinyprod.net/repos/debian squeeze main" >> tinyprod-debian.list

$ sudo echo "deb http://tinyprod.net/repos/debian msp430-46 main" >> tinyprod-debian.list

1. 4 Install the new packages:

$ sudo apt-get update

$ sudo apt-get install nesc tinyos-tools avr-tinyos

1. 4 Install source code of tinyos-2.1.2 by following the official tutorial

http://tinyos.stanford.edu/tinyos-wiki/index.php/Automatic_installation

1. 5 Test. The following output shows succeed:

hh@milkyway:~/tinyos-main/apps$ cd Blink/

hh@milkyway:~/tinyos-main/apps/Blink$ ls

BlinkAppC.nc BlinkC.nc Makefile README.txt

hh@milkyway:~/tinyos-main/apps/Blink$ make iris

mkdir -p build/iris

```
    compiling BlinkAppC to a iris binary
```

ncc -o build/iris/main.exe -Os -fnesc-separator=__ -Wall -Wshadow -Wnesc-all -target=iris -fnesc-cfile=build/iris/app.c -board=micasb -DDEFINED_TOS_AM_GROUP=0x22 --param max-inline-insns-single=100000 -DIDENT_APPNAME=\"BlinkAppC\" -DIDENT_USERNAME=\"hh\" -DIDENT_HOSTNAME=\"milkyway\" -DIDENT_USERHASH=0x3230e9aaL -DIDENT_TIMESTAMP=0x56b4ce96L -DIDENT_UIDHASH=0x28d44b9dL -fnesc-dump=wiring -fnesc-dump='interfaces(!abstract())' -fnesc-dump='referenced(interfacedefs, components)' -fnesc-dumpfile=build/iris/wiring-check.xml BlinkAppC.nc -lm

```
    compiled BlinkAppC to build/iris/main.exe
            2272 bytes in ROM
              51 bytes in RAM
```

avr-objcopy --output-target=srec build/iris/main.exe build/iris/main.srec
avr-objcopy --output-target=ihex build/iris/main.exe build/iris/main.ihex

```
    writing TOS image
```

Reference:

http://tinyprod.net/repos/debian/README-46.html

http://www.cse.wustl.edu/~lu/cse467s/slides/tinyos-installation.pdf

**MSPGCC 4.6.3 LTS20120406 Release**
**Long Term Support (LTS)**

An alternative source for the msp430-46 packages is stanford: "deb http://tinyos.stanford.edu/tinyos/dists/ubuntu natty main" This release is the primary MSP430 toolchain for the TinyOS T2.1.2 release. It also can be used for the TinyOS development trunk.

Last update: 20121019, cire

(Changelog at end)

The 4.6.3-LTS20120406 repository contains the following packages for the *i386* (32 bit) and *amd64* (64 bit) architectures:

- msp430-46: 20120715
- msp430-binutils-46: 2.21.1-LTS20120406
- msp430-gcc-46: 4.6.3-LTS20120406
- msp430-gdb-46: 7.2-LTS20120406
- msp430-libc-46: 20120224-LTS20120406+20120502
- msp430mcu-46: 20120406-LTS20120406+20120502

Release notes can be found at Release Notes 46

To use this repository you need to:

1. Add the following lines to /etc/apt/sources.list.d/tinyprod-debian.list:

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

```
$ cd /etc/apt/sources.list.d
$ sudo echo "deb http://tinyprod.net/repos/debian squeeze main" >> tinyprod-debian.list
$ sudo echo "deb http://tinyprod.net/repos/debian msp430-46 main" >> tinyprod-debian.list
```

2. Install the new packages:

```
$ sudo apt-get update
$ sudo apt-get install msp430-46 nesc tinyos-tools
```

If you want to uninstall the packages you can do it like this:

```
$ sudo apt-get autoremove --purge msp430-46
```

**Note:**

1. Other packages from the main tinyprod debian repository (**squeeze**) are needed to make a functioning nesc development sytem. In particular, you will need **nesc** and **tinyos-tools**. These have been included in the instructions above.

2. Experimental msp430 toolchains install into /opt/msp430-. 4.6.3 is the released default toolchain for the msp430 cpus for tinyos 2.1.2. It installs into the /usr heirarchy.

**Note**

This repository is signed with the following key:

```
pub   2048R/34EC655A 2011-02-28
      Key fingerprint = 2ADB 95E2 E116 3F0C 0B55  73F0 DB53 87FB 34EC 655A
uid                  Eric Decker
```

To install the key in the local apt keyring you have to do the following:

```
$ gpg --keyserver keyserver.ubuntu.com --recv-keys 34EC655A
```

```
$ gpg -a --export 34EC655A | sudo apt-key add -
```

-- cire (October 19, 2012)

**Changelog:**

```
20121019, cire: move 4.6.3 toolchain to /usr
20120716, cire: rebuild tinyprod repository
```

Jan.12, 2014
Author: Rahav Dor

# Student installation of TinyOS



## TinyOs install – Automatic installation

The instructions here follow (with fixes and clarification) the procedure on http://tinyos.stanford.edu/tinyos-wiki/index.php/Automatic_installation

This process was tested with *Ubuntu Desktop 13.10, 64 bit* on *VMware Fusion* ver 6.02 virtual machine (OS-X version of VMware) and *Ubuntu Desktop 13.10, 32 bit* on *VirtualBox* ver 4.3.6 virtual machine.

I cannot over stress how important it is that each command you run will end with no errors. Often there are gobs of output shown on the screen during the install process and no one wants to ever look at that. You should at least observe the process and generally see that it does not seem out of ordinary. We human beings are really good after seeing a few of these at identifying what is normal and what is not. If you do not follow this recommendation, you run the risk of your development environment superficially seeming to be working, while unexpectedly breaking or challenging you with odd results. Linux installations typically end very quietly, if you want to check how a process ended type

```
echo $?
```

Immediately after a process has finished. The value `0` denotes good condition.

## Get Linux

Download **Ubuntu Desktop 12.04 (LTS)** or **Ubuntu Desktop 13.10**, either 32 or 64 bit versions will work, but note that Ubuntu 64 bit needs 2 GB of memory, and that you cannot install TinyOS from source (we are not doing it here, we are using the automatic installation) on 64 bit versions of Ubuntu. As of Jan. 2014 I recommend using the 32 bit versions.

## Install Ubuntu on a Virtual Machine

Follow either the *VMware* or *VirtualBox* instructions below.

### Install Ubuntu on VMware

If you are installing on a Mac (OS-X) the correct version of VMware is called Fusion. I tried this install process on VMware version 6.02.

Start *VMware* and select the *Add* button in the top left corner, then select
*New…*
*Install from disk or image*
and click *Continue*.

In the *New Virtual Machine* window click *Use another disk or disk image…* and select the Linux .iso file you downloaded earlier. For example:

```
Rahav/Downloads/ubuntu-13.10-desktop-amd64.iso
```

Give Ubuntu the required data to build the instance:

| | |
|---|---|
| Full user name | |
| User | |
| Password | |

When the installation is completed, startup *Ubuntu* in the VM and remove from the left sidebar unneeded icons: Open office, Ubuntu One (music, pics, etc.), Amazon, Ubuntu One Music, and so on.

I also recommend that you will drag the Terminal app to the sidebar, you will be using it a lot.

Get the latest Linux updates by clicking on *Software Updater* icon (typically available on the left icon bar). If you installed version 12.x DO NOT upgrade to version 13.x because the update from 12.x to 13.x fails. If you want 13.x download it directly.

### Installing Ubuntu on VirtualBox

I tried this install process on VirtualBox version 4.3.6.

Click the *New* icon and provide the installer with the needed data. For example:

Name: **Ubuntu 32-bit w TinyOS 2.1.2**
Type: **Linux**
Version: **Ubuntu**

Set the *Memory size* to **1024 MB** (1 GB) if you are installing Ubuntu 32 bit or **2048** if 64 bit.

See that radio button named **Create a virtual hard drive now** is selected and click *Create*.

I recommend that you will use the Hard drive file type named **VDI**.

On the next screen see that **Dynamically allocated** is selected.

On the screen titled File location and size, you will be asked to name the virtual drive you are creating. To place it in your preferred directory you can type the full path and file name, or click the icon to the right of the field to select the desired directory and filename. The name and location I gave my virtual hard drive is:

```
/Users/Rahav/VirtualBox VMs/Ubuntu 32-bit w TinyOS 2.1.2/HD.vdi
```

Leave the default size of the drive at the recommended 8.00 GB.

The VM will be created and appear on the left pane ov *VirtualBox*. Start it. A window with long text asking you to "Please select a virtual optical disk file…" will appear. What they mean is for you to select the Ubuntu .iso file you downloaded earlier. Unless you already installed from this .iso *VirtualBox* will not know about it, so ignore the offered options and click the icon to their left. Locate the proper .iso file:

```
/Users/Rahav/Downloads/ubuntu-13.10-desktop-i386.iso
```

This concludes the VM preparation; next we will install Ubuntu on this instance. Ubuntu will start, you just need to follow the wizard. Start by clicking the *Install Ubunto* button (you need to be connected to the Internet for that).

Do not check the option to *Download updates while installing* (you will need to do it again after the install completes and it may screw up your 12.x version if that is what you are installing). I also do not recommend that you will check *Install this third-party software*.

In the **Installation type** screen select **Erase disk and install Ubuntu**. Do not encrypt or use LVM.

From now on, self explanatory screens will not be articulated here.

After a few screens you will get to the **Who are you?** Give it the needed data, example is provided in the table:

| | |
|---|---|
| Your name | Rahav Dor |
| Computer name | rahavd-VirtualBox |
| Username | |
| Password | |
| | Log in automatically |

I believe that Ubuntu 13.x presents you with the **Ubuntu One** screen and 12.x does not (Ubuntu One provides you with one account to log in to everything on Ubuntu). Anyway, I recommend that you will select *Log in later*. We will never *log in* actually, this is not needed for TinyOS development and the dev. environment is complicated enough without it.

The installation process will now take a bit to do its thing. Restart the computer (its actually the Ubuntu instance that you are restarting, not your computer) when you are asked to.

After you login, if you installed 12.x decline the proposal to upgrade to 13.x.

After the installation is complete remove from the left sidebar unneeded icons such as: the open office, Ubuntu One (music, pics, etc.), Amazon, Ubuntu One Music, ….

Get the latest Linux updates by clicking on *Software Updater* icon (typically available on the left icon bar). Remember, if you installed version 12.x DO NOT upgrade to version 13.x because the update from 12.x to 13.x fails. If you want 13.x download it directly.

## Get prerequisites software

We will be installing **vim**, an editor that is a bit more intuitive than the default **vi** that comes with Linux. Open a Terminal window and type the following command (not need to change directory) from the default one you start in.

```
sudo apt-get install vim
```

Make sure that the installation completed without any warnings. This is true for anything else we do here.

A command summary of vim is available here and in many other places:
http://www.fprintf.net/vimCheatSheet.html

### Install TinyOS compilers for various uProcessors

See http://tinyprod.net/debian-dev/ if more details than provide here are needed (they are not).

We will edit **/etc/apt/sources.list.d/tinyprod-debian.list** and add two repositories location. These repositories contain the compilers we need. Using vim:

```
sudo vim /etc/apt/sources.list.d/tinyprod-debian.list
```

go to the end of the file and add the following:

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

Save the file and exit back to the Terminal prompt.

Install the repository security key by running the commands:

```
gpg --keyserver keyserver.ubuntu.com --recv-keys 34EC655A
```

Sometimes you need to exit terminal here, so I recommend that you will just do it and open Terminal again. Now run the command:

```
gpg -a --export 34EC655A | sudo apt-key add -
```

You must see an Exit message: **OK**. If not you need to investigate why not.

Exit the terminal again and start a new one to make the key take affect.

Now we can install the packages. In Terminal we run:

```
sudo apt-get update
sudo apt-get install nesc tinyos-tools msp430-46 avr-tinyos
```

## Get TinyOS

Working in your **home** directory, run in Terminal:

```
wget http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
```

unzip the file and move it to a more suitable location (here we just rename it because we want to be able to have all read/write to the source code):

```
tar xf tinyos-2_1_2.tar.gz
mv tinyos-release-tinyos-2_1_2 tinyos-main
```

### Setup TinyOS dev. environment

You will need to add some environment variables to your shell. The following are the necessary ones. Using **vim** edit **.bashrc** and add the following lines at the end of the file:

```
# Setup the environment variables needed by the TinyOS make system
export TOSROOT="$HOME/tinyos-main"
echo "setting up TinyOS source path to $TOSROOT"
export TOSDIR="$TOSROOT/tos"
export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java/tinyos.jar:.
export MAKERULES="$TOSROOT/support/make/Makerules"
export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
```

In order to program motes you will need to access the serial ports. Joining the group that grants this privilege can grant you this access. The Linux command **gpasswd** administer the **/etc/group** file and **dialout** is the group that owns the serial ports, so the following command will add you to this group. Run:

```
sudo gpasswd -a <your-user> dialout
```

This change only takes effect when log out from Ubuntu and log in again.

## Test the installation

Start by testing the environment by running:

```
tos-check-env
```

If you did everything properly you will see only one WARNING about *graphviz* not being version 1.10 (it will appear twice). This is actually OK because you will have a newer version (likely 2.x). To check run:

```
dot –V
```

Now lets compile a TinyOS application. Probably the most known one is **Blink**. The following should compile with absolutely no errors:

```
cd tinyos-main/apps/Blink/
make telosb
```

Now lets install the app on a mote. The following command should compile and install on a TelosB note with no errors, blinking the 3 LEDs:

```
make install telosb
```

Note that *VirtualBox* may not discover when you connect a mote to the host's computer USB port. If this occurs do it manually. To connect the mote to the VM click the USB devices icon at the bottom.


and attach the desired USB device to Ubuntu by selecting it from the pop-up list (should be something as ... TelosB …).

That's it. You should now have a perfectly working TinyOS development environment.