# How to set up the TinyOS environment on Debian

Marc Stoecklin

May 10, 2005

## 1 Introduction

This is a description how a TinyOS environment is set up on a GNU/Linux Debian system. It includes the following components:

- Installing the basic TinyOS system
- Setting up NesC (Network Embedded System C)
- Configure Java and the serial interface

## 2 Notation

Let us agree on a number of coventions for the notation in this HOWTO before starting with the setup of the TinyOS environment.

When describing commands, file names or code that has to be typed in (or copied to the right place), the `typewriter` font is used. Command line instructions are preceeded by the dollar sign (`$`) or the sharp sign (`#`) depending if the subsequent command should be executed with user or root permissions respectively. If a line isn't preceeded by one of these two signs, a configuration file is normally edited. A command or configuration that does not fit on a single line in this HOWTO is indented on the following line.

Web resources (URLs) are centered on the page such that they can be easily copied to the clipboard.

# 3 The guide

## 3.1 TinyOS

In order to get the TinyOS system you may choose several ways. Possibly the most simple one is to download the *rpm*s that are available on

http://www.tinyos.net/dist-1.1.0/tinyos/linux/

Make sure that you have the `alien` package (and possibly `fakeroot` to convert it as non-root user) installed in order to generate Debian packages (`*.deb`) from the *rpm*s.

```
# apt-get install alien fakeroot
$ fakeroot alien -d tinyos-1.1.0-1.noarch.rpm
# dpkg -i tinyos_1.1.0-1.noarch.deb
```

As an alternative, you may downlaod TinyOS directly from its CVS repository. To do so, you need to have installed the CVS package, as root or with `sudo`

```
# apt-get install cvs
```

Then you can directly choose the directory where the TinyOS system should be installed (we assume `~/tinyos/`) and let CVS do the rest for you (if you are prompted for a password, just hit enter; no password is required).

```
$ cd ~/tinyos/
$ cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/tinyos login
$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/tinyos
    co tinyos-1.x
```

## 3.2 AVR tools and GraphViz

From http://www.tinyos.net/dist-1.1.0/tools/linux/, download

- avr-binutils-2.13.2.1-2.i386.rpm
- avr-gcc-3.3tinyos-2.i386.rpm
- avr-libc-20030512cvs-2.i386.rpm

and transform them in Debian packages with `alien` (if you haven't installed `alien`, refer to the description in the previous section).

```
$ fakeroot alien -d avr-binutils-2.13.2.1-2.i386.rpm
$ fakeroot alien -d avr-gcc-3.3tinyos-2.i386.rpm
$ fakeroot alien -d avr-libc-20030512cvs-2.i386.rpm
```

```
# dpkg -i avr-binutils_2.13.2.1-2_i386.deb
# dpkg -i avr-gcc_3.3tinyos-2_i386.deb
# dpkg -i avr-libc_20030512cvs-2_i386.deb
```

### 3.3 NesC Compiler

Download the NesC Compiler *rpm* package `nesc-1.1.2a-1.i386.rpm` from

http://www.tinyos.net/dist-1.1.0/tinyos/linux/

Install it, using the Debian package manager again.

```
$ alien -d nesc1.1.2a-1.i386.rpm
# dpkg -i nesc_1.1.2a-1_i386.deb
```

You may download the source files from http://sourceforge.net/projects/nescc/ as an alternative, i.e. from

http://prdownloads.sourceforge.net/nescc/nesc-1.1.3.tar.gz?download

Extract them in a separate directory and install them.

```
$ tar xvzf nesc-1.1.3.tar.gz
$ cd nesc-1.1.3
$ ./configure TOSDIR=~/tinyos/
$ make
# make install
```

### 3.4 Java Platform 2 Version 1.4.x

Most probably you are going to make use of Java when accessing the motes connected to your computer. You can either install Java with Sun's installer from http://java.sun.com/j2se/1.4.2/download.html (not recommended) or use the prepared Debian package. To do so, you have to modify your `sources.list` file and update your package management system before you install the `j2sdk1.4` package.

```
# echo "deb ftp://ftp.tux.org/java/debian/ sarge non-free" >>
    /etc/apt/sources.list
# apt-get update
# apt-get install j2sdk1.4
```

Java will normally be located in `/usr/lib/j2se/1.4/` which is what we assume in the follwing. To simplify the next steps, we export this path as an environment variable.

```
# export JAVAROOT=/usr/lib/j2se/1.4/
```

### 3.5 Javax.comm

Communication with a mote is done via the serial interface in Java. This requires to install the Java Comm API. We will use an open source implementation.

First, obtain the RXTX binaries from

http://www.linux.org.uk/~taj/rxtx-bins.1.tar.gz

and install the libraries for the serial and parallel interface as well as the `jcl.jar` archive.

```
$ tar xfz rxtx-bins.1.tar.gz
# cp rxtx-bins.1/1.4/i386-pc-linux/libSerial.so
     $JAVAROOT/jre/lib/i386/
# cp rxtx-bins.1/1.4/i386-pc-linux/libParallel.so
     $JAVAROOT/jre/lib/i386/
# cp rxtx-bins.1/1.4/jcl.jar $JAVAROOT/jre/lib/ext/
```

Then, you need to install the Comm API which you can download from

http://java.sun.com/products/javacomm/downloads/index.html

Make sure to find the Solaris/SPARC version. Install the `comm.jar` archive and set the properties for the driver accordingly.

```
$ tar xfz javax_comm-2_0_3-solsparc.tar.Z
# cp commapi/comm.jar $JAVAROOT/jre/lib/ext/
# /bin/echo Driver=gnu.io.RXTXCommDriver >
     $JAVAROOT/jre/lib/javax.comm.properties
```

### 3.6 Configuration and setup

Now, you have to allow access to the serial interface or add your user to the `dialout` group.

```
# chmod 777 /dev/ttyS0
```

Note that you may add the follwing lines to your `.bashrc` or `.bash_profile` file to facilitate work (note the ‘...‘ signes in the classpath's setup. Here the `javapath` script is called):

```
export TOSROOT="~/tinyos/tinyos-1.x"
export TOSDIR="$TOSROOT/tos"
export TOSAPPS="$TOSROOT/apps"
export TOSTOOLS="$TOSROOT/tools"
```

```
export CLASSPATH="$CLASSPATH:‘$TOSROOT/tools/java/javapath‘:
  $TOSROOT/tools/java/net:/usr/lib/j2se/1.4/jre/lib/ext/"

export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/lib/:
  /usr/lib/j2se/1.4/jre/bin/:/usr/lib/j2se/1.4/include/"

PATH="$PATH:$TOSROOT/tools/java/net/tinyos/sim"

# Depending on your system and the serial port you are using
# /dev/ttyS0 (COM1) or /dev/ttyS1 (COM2)

export MIB510=/dev/ttyS0
export MOTECOM=serial@/dev/ttyS0:mica2
```

Since the TinyOS environment for Linux has been prepared for the Fedora distribution, we have to modify two configuration files to indicate the correct paths to the Java inludes. Please keep in mind that in this HOWTO we assume Java to be installed in `/usr/lib/j2se/1.4/`.

In `$TOSROOT/tools/java/jni/Makefile` replace the lines starting with `JNI=` and `JAVAC_DIR=` with the following lines (depending on your Java installation):

```
JNI="/usr/lib/j2se/1.4/include/"
JAVAC_DIR="/usr/lib/j2se/1.4/bin/"
```

The file `$TOSROOT/tools/java/jni/Makefile.Linux` has also to be modified, more precisely on line 4 starting with `gcc` the following include must be added (place it after `"-I$(JDK)/include"`):

```
"-I$(JDK)/include/linux"
```

Now you can get started by building the TinyOS environment helpers. To do so, you need to have installed the Debian packages `automake` and `g++`.

```
$ cd $TOSTOOLS
# make install
```

Next, you have to load the newly installed library `libgetenv.so` which is installed in the directory `/usr/lib/j2se/1.4/include/`.

```
# ldconfig
```

Notice: If you encounter problems with the library when running Java applications, copy `libgetenv.so` to `/usr/lib` and re-run the `ldconfig` command as root-user.

Finally, try the `toscheck` script that indicates if you installed correctly your TinyOS environment.

```
$ tools/scripts/toscheck
```

Note: this may fail for some component such as `graphviz`, `avarice` or the classpath. If you have set up your environment as described, you should be able to use it without problems now.

Now, you are ready to compile your first TinyOS application, e.g. the `Blink` application. To do so, go into the applications directory and execute the compile command.

```
$ cd $TOSAPPS/Blink
$ make pc
```

To install the application on a (serially) connected *Mica2* mote, hit

```
$ make mica2 install
```

Notice: please keep in mind to have the `MIB510` evironment variable set correctly. This variable indicates the install (upload) destination. For a *Mica2 Dot* mote, use `make mica2dot install` instead.

Finally, you may also build the Java applications provided with TinyOS (this takes quite a while on slower PCs).

```
$ cd $TOSTOOLS/java
$ make
```

In order to compile your own Java application, you have to put the `Makefile` into its package directory and adjust the fields appropriately. To launch an application, e.g. `Example.class` in the `net.tinyos.example` package, the following command is required (you have to be root user or use `sudo` or `fakeroot`):

```
$ cd $TOSTOOLS/java
# java -noverify net.tinyos.example.Example
```

Notice: This time the environment variable `MOTECOM` has to be set correctly.