Setting up TUnit

From TinyOS Wiki

Contents

Minimum Requirements

The minimum requirements are:

- Install (http://docs.tinyos.net/index.php/Main_Page#Installing_TinyOS) the latest tinyos-2.x, or get it from the repository (http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x) .
  - Make sure tunit.extra is in your `tinyos-2.x/support/make` directory.
- Check out a copy of the latest tinyos-2.x-contrib (http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/) repository (instructions here (http://sourceforge.net/cvs/?group_id=28656)).
- Ant (http://ant.apache.org/) is completely *optional*, but may be used to generate nicer looking HTML reports (http://www.lavalampmotemasters.com/reports/html/index.html) .

Directory Structures

Simplistic TinyOS Directory Structure

Your projects and directories may differ. On the automated unit testing (http://www.lavalampmotemasters.com/) server, the tinyos-2.x and tinyos-2.x-contrib directories are located in a simplistic structure:

```
opt
|-- tinyos-2.x
|-- tinyos-2.x-contrib
|   |-- tunit
|   |   |-- tests
|   |   |   |-- tinyos-2.x
```

```
|   |   |   |   |-- [ All tinyos-2.x baseline tests ]
|   |   |   |-- tinyos-2.x-contrib
|   |   |   |   |-- [ All tinyos-2.x-contrib tests ]
|   |-- ....
```

Complex TinyOS Directory Structure

In my own private projects, I prefer a more complex structure:

```
opt
|-- myProject
|   |-- tinyos-2.x    <-- A snapshot of TinyOS my project is compatible with
|   |-- tinyos-2.x-myProject    <-- Project's files that override the TinyOS baseline
|   |   |-- tests
|   |   |   |-- [bunch of test directories]
|
|-- tinyos-2.x-contrib
|   |-- tunit
|   |   |-- ...
```

For the remainder of this setup guide, I will assume your TinyOS is setup in the simplistic structure, with directories located at /opt/tinyos-2.x and /opt/tinyos-2.x-contrib.

Environment Variables

TUnit Tip
TUnit comes with an example .tunitclasspath file you can use to setup environment variables. This file is in tinyos-2.x-contrib/tunit/.tunitclasspath Edit the file to match your local settings and source it to apply the environment variables.

TOSCONTRIB

TUnit needs to know where the tinyos-2.x-contrib is located so it may find the tinyos-2.x-contrib/tunit directory. Because TUnit is driven by a Java application, the TOSCONTRIB path must be absolute so Java can find it.

Linux

```
  export TOSCONTRIB=/opt/tinyos-2.x-contrib
```

Windows/Cygwin

```
  export TOSCONTRIB="`cygpath -w /opt/tinyos-2.x-contrib`"
```

TOSMAKE_PATH

TUnit creates a .extra file in your local application's directory when compiling. To allow nesC to locate this file, you must set the TOSMAKE_PATH environment variable to include your local directory:

```
export TOSMAKE_PATH="./"
```

CLASSPATH

Your classpath must be updated to include tunit.jar (http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/tunit/support/sdk/tunit/) and all of its dependencies (http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/tunit/support/sdk/tunit/depends/) .

```
Linux
  export LOG4J=$TOSCONTRIB/tunit/support/sdk/tunit/depends/log4j.jar
  export TUNIT=$TOSCONTRIB/tunit/support/sdk/tunit/tunit.jar
  export XERCES=$TOSCONTRIB/tunit/support/sdk/tunit/depends/xerces.jar
  export JFREECHART=$TOSCONTRIB/tunit/support/sdk/tunit/depends/jfreechart-1.0.5.jar
  export JCOMMON=$TOSCONTRIB/tunit/support/sdk/tunit/depends/jcommon-1.0.9.jar

  export CLASSPATH="${CLASSPATH}:${LOG4J}:${TUNIT}:${XERCES}:${JFREECHART}:${JCOMMON}:."
```

```
Windows/Cygwin
  export LOG4J="`cygpath -w $TOSCONTRIB/tunit/support/sdk/tunit/depends/log4j.jar`"
  export TUNIT="`cygpath -w $TOSCONTRIB/tunit/support/sdk/tunit/tunit.jar`"
  export XERCES="`cygpath -w $TOSCONTRIB/tunit/support/sdk/tunit/depends/xerces.jar`"
  export JFREECHART="`cygpath -w $TOSCONTRIB/tunit/support/sdk/tunit/depends/jfreechart-1.0.5.jar`"
  export JCOMMON="`cygpath -w $TOSCONTRIB/tunit/support/sdk/tunit/depends/jcommon-1.0.9.jar`"

  export CLASSPATH="${CLASSPATH};${LOG4J};${TUNIT};${XERCES};${JFREECHART};${JCOMMON};."
```

First Run

After setting up the TOSCONTRIB and CLASSPATH environment variables, you should able to execute the TUnit Java application. The initial errors are ok, we'll fix those next. I'm running TUnit from cygwin, hence the windows paths indicated below.

```
$ java com.rincon.tunit.TUnit
0 [main] INFO com.rincon.tunit.TUnit  - Base package directory located: C:\
0 [main] INFO com.rincon.tunit.TUnit  - Found a TOSCONTRIB environment variable
0 [main] DEBUG com.rincon.tunit.TUnit  - Found TUnit! C:\cygwin\opt\tinyos-2.x-contrib\tunit
Running TUnit from C:\cygwin\opt\tinyos-2.x-contrib\tunit
0 [main] FATAL com.rincon.tunit.TUnit  - Cannot locate C:\cygwin\opt\tinyos-2.x-contrib\tunit\tunit.xml
0 [main] FATAL com.rincon.tunit.TUnit  - Does tinyos-2.x-contrib/tunit/tunit.xml exist?
```

XML Files

tunit.xml

TUnit requires the tunit.xml file to tell it what hardware you have connected to the computer and how to talk to it. There is an example tunit.xml file located at tinyos-2.x-contrib/tunit/tunit_example.xml (http://tinyos.cvs.sourceforge.net/*checkout*/tinyos/tinyos-2.x-contrib/tunit/tunit_example.xml?content-type=text%2Fplain) , which you can use to start creating your own tunit.xml file:

```
<tunit>

  <testrun name="1_telosb">
    <mote target="telosb" motecom="serial@COM18:tmote" installextras="bsl,17"/>
  </testrun>

  <testrun name="2_telosb">
    <mote target="telosb" motecom="serial@COM22:tmote" installextras="bsl,21"/>
    <mote target="telosb" motecom="serial@COM18:tmote" installextras="bsl,17"/>
  </testrun>

</tunit>
```

You need to create a tunit.xml file in the tinyos-2.x-contrib/tunit directory, and list the types of hardware you have connected to your computer. If you change that hardware, you must manually update the tunit.xml file.

build.xml

This file is not required to run TUnit. It is optional in the case that you want to use Ant to run TUnit. There is an example build.xml file located at tinyos-2.x-contrib/tunit/build_example.xml (http://tinyos.cvs.sourceforge.net/*checkout*/tinyos/tinyos-2.x-contrib/tunit/build_example.xml?content-type=text%2Fplain)

When running TUnit standalone on your own computer, the build.xml file is mostly responsible for creating TUnit HTML reports (http://www.lavalampmotemasters.com/reports/html/index.html) . The build.xml file is also used for automated unit testing, which can be configured to check out the latest TinyOS CVS repositories, run tests, and generate reports.

Read the Ant documentation (http://wiki.apache.org/ant/FrontPage) to find out more about the build.xml file.

You can use CruiseControl (http://cruisecontrol.sourceforge.net/) or a custom infinite-loop bash script to run the build.xml file in an automated unit testing fashion.

lava.xml

You can safely ignore and not create the lava.xml file, unless you really want to get lava lamps displaying the results of your automated unit testing system.

Verifying the Setup

To verify everything is setup correctly, create an empty directory and run TUnit. If everything is working, you'll see TUnit results displayed at the end.

```
$ mkdir emptydir
$ cd emptydir
$ java com.rincon.tunit.TUnit

 ... [ lots of command-line output ] ...

T-Unit Results
-----------------------------------------------
Total runtime: 4.448 [s]
Total tests recorded: 0
Total errors: 0
Total failures: 0
```

Execute!

Alias the command line

First, for proper test-driven development (http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd) , we expect you'll continuously and repetitively run Unit in the background while actively editing the code under development. To make TUnit easier and faster to execute, alias the command to run it:

```
alias tunit="java com.rincon.tunit.TUnit"
```

The "tunit" alias will be used from here on out to represent "java com.rincon.tunit.TUnit".

Run a single-node test

There are lots of simple tests to try in the tinyos-2.x-contrib/tunit/tests directory. Assuming you have a test run containing a single node, I recommend trying the following:

```
$ cd $TOSCONTRIB/tunit/tests/tinyos-2.x/tos/lib/tunit/TestAssertions
$ tunit

  ... [ lots of command-line output ] ...

T-Unit Results
----------------------------------------------
Total runtime: 41.891 [s]
Total tests recorded: 14
Total errors: 0
Total failures: 0
```

See Also

- TUnit
- Flow of TUnit Java Execution
- tunit.xml
- suite.properties
- TUnit Assertions

Next

- Single-Node Unit Testing