TinyOS – Hands-on

Documentation updated and verified 19 September 2018

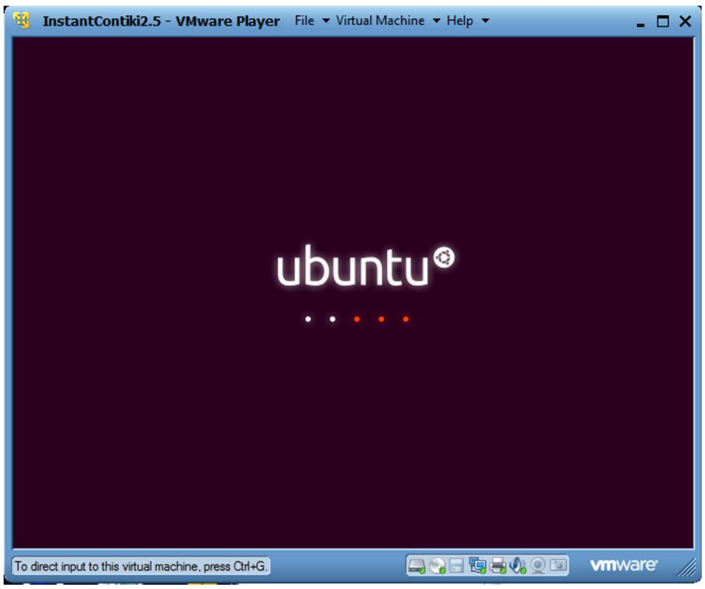
In this assignment you will set-up TinyOS in a virtual environment and run a simple program which simulates blinking leds on a hardware platform. To complete this assignment you should:

- Program TinyOS tasks
- Complete a report

To make the procedure equal on all platforms (Linux/Mac/Windows) we run this exercise inside a virtual image using VMWare Player.

After completing this assignment you will:

- Be able to program a TinyOS task
- Understand basic scheduling in TinyOS

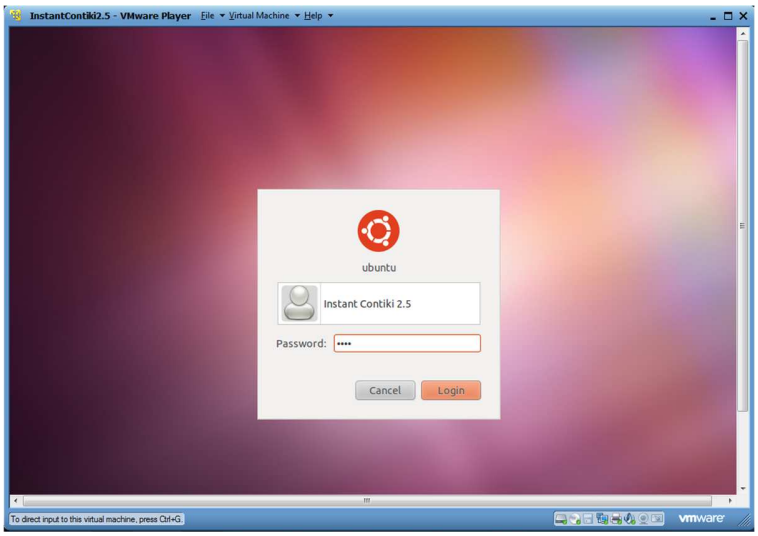


Preparation

Download VMWare Player at the url:
http://www.vmware.com/go/downloadplayer/

Download the Instant Contiki image (Ubuntu 14.04)
http://sourceforge.net/projects/contiki/files/Instant%20Contiki/
Note: You will need this image later in the Contiki assignment as well.

Extract this file.



Boot the Contiki image in VMWare Player
Start Instant Contiki by running
InstantContiki2.6.vmx. Wait for the virtual Ubuntu Linux boot up.

Log into Instant Contiki. The password is “user”.

Also the sudo password is “user”

Open a terminal window

Update the software repository
$sudo apt-get update

Install Nescc
$ sudo apt-get install nescc

Install Python
$ sudo apt-get install python-dev

Add TinyOS to sources
$ sudo nano /etc/sources.list
Add the following line to the bottom of this file
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main

Update the software repository
$sudo apt-get update
$sudo apt-get upgrade

Install TinyOS
$ sudo apt-get install tinyos-source

Go to your home directory
$ cd /home/user

Clone the TinyOS tools repository from git
$ git clone https://github.com/tinyos/tinyos-release.git

Go to the TinyOS directory
$ cd /home/user/tinyos-release

Go to the tools directory and configure TinyOS with the following commands
$ cd tools
$ ./Bootstrap
$ ./configure –prefix=/opt/tinyos
$ make
$ sudo make install

Go to the TinyOS root directory
$ cd ..

Create a file here called tinyos.sh and add the following content into this file:

```
#! /usr/bin/env bash
# Here we setup the environment
# variables needed by the tinyos
# make system
echo "Setting up for TinyOS 2.1.2 Repository Version"
export TOSROOT=
export TOSDIR=
export MAKERULES=
TOSROOT="/home/user/tinyos-release"
TOSDIR="$TOSROOT/tos"
CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java:.:$TOSROOT/support/sdk/java/tinyos.jar
MAKERULES="$TOSROOT/support/make/Makerules"
export TOSROOT
export TOSDIR
export CLASSPATH
export MAKERULES
```

Save the file.

Set up the paths by running the following commands:
$ source /home/user/tinyos-release/tinyos.sh export TOSROOT="/home/user/tinyos-release"
$ export TOSDIR=$TOSROOT/tos
$ export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
$ export MAKERULES=$TOSROOT/support/make/Makerules
$ export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
$ export PATH=/opt/msp430/bin:$PATH
$ export APP="$TOSROOT/apps"

Change permissions to allow “user” to modify the content of TinyOS
$ sudo chown -R user:user /home/user/tinyos-main/

Now TinyOS should be set-up.
For more documentation on TinyOS check this out:
http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page

Task1: Simulate blinking leds

In this task you will simulate timers used to blink leds in a TinyOS program.
The demo programs are located in the “apps” folder.
You need to compile the tasks using the TinyOS compiler and run the compiled environment in the TOSSIM simulator using a Python script.

Go to the demo program for blinking leds
$ cd /home/user/tinyos-main/apps/Blink

Compile the program with the TOSSIM simulator as a target
$ make micaz sim
It should output “Successfully built micaz TOSSIM library”

To run the program you must create and edit a file called “runblink.py”
$ gedit runblink.py

paste the following runner code in this file and save:

```
#!/usr/bin/python
from TOSSIM import *
import sys
t = Tossim([])
r = t.radio()
t.addChannel("BlinkC", sys.stdout)
m = t.getNode(1)
m.bootAtTime(100)
while (m.isOn() == 0):
  t.runNextEvent()
for i in range(0, 100):
  t.runNextEvent()
```

This runner code will use the compiled TinyOS environment and execute the task related code in this environment
Run the TinyOS environment and study the output.
$ python runblink.py

Task2: Add another timer

The two following modifications should be made to the Blink application code:

1) Add another timer called “Timer3” and schedule it every 100ms.
User this time to print out the message “I am Timer 3 and I have the shortest period!”

2) Make the simulation run for 2000 events instead of 100

Submission

- Create a report as a PDF file over the functionality of the application code, explain how it works and how the priorities are set in this example.
- Provide screenshot of the running original application.
- Explain how you did the modifications to the code and provide a screenshot of the updated application.
- Compress all files in the apps/Blink folder. Make sure you include both the original and the modified application code. **Submit the application folder and the PDF report as a single zipfile on Coursera.**