

Basic TinyOS Programming

This lab provides a brief introduction to TinyOS programming in nesC. For those of you who already have experience with C or C++, writing basic nesC programs is fairly simple: all you need to do is implement one or two modules and wire them together. The difficulty comes when building larger, more complex applications. While this lab focuses specifically on helping you get experience writing simple applications, the lessons learned are fundamental, and can be applied to all TinyOS applications no matter how complex they may be.

The lab itself consists of a series of exercises designed to get you both comfortable with the environment within which your TinyOS applications will be developed, as well as get you started writing simple applications using the libraries provided by TinyOS.

All of the exercises in this lab are based on the source code found in the tarball at the location below:

```
http://sing.stanford.edu/klueska/handsOnTinyOS-lab1.tar.gz
```

If you haven't downloaded this yet, please do so now. Each exercise builds on the previous one, so make sure you complete them each in order.

Exercise 0: Null

This exercise is purely meant as a sanity check to make sure your basic TinyOS setup is correct. All we are going to do is run 'make' on this application and make sure it compiles.

```
cd handsOnTinyOS-lab1/Null
make telosb
```

If everything went smoothly, you should see output similar to the following:

```
mkdir -p build/telosb
  compiling NullAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -O -mdisable-hwmul -Wall -Wshadow
-Wnesc-all -target=telosb -fnesc-cfile=build/telosb/app.c -board=
-DDEFINED_TOS_AM_GROUP=0x22 -DIDENT_APPNAME=\"NullAppC\"
-DIDENT_USERNAME=\"klueska\" -DIDENT_HOSTNAME=\"38-200.mops.rwt\"
-DIDENT_USERHASH=0xaab24772L -DIDENT_TIMESTAMP=0x498918c9L
-DIDENT_UIDHASH=0x0c856deaL NullAppC.nc -lm
  compiled NullAppC to build/telosb/main.exe
      1398 bytes in ROM
      4 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
  writing TOS image
```

Exercise 1: PushAndToggle

This exercise is designed to get you comfortable writing a minimal TinyOS application that actually does something. Your task is to take the skeleton application found in the `PushAndToggle` directory and implement logic that turns Led0 on whenever the telosb user button is pressed and turns it off whenever the user button is released. After you've gotten this part working, try changing the logic so that the led toggles each time

the button is pressed instead of simply turning on and off when it is pressed and released (Hint: will require a local state variable in the module).

As a starting point, take a look at the comments found in the skeleton application as well as the header file found in: `tinyos-2.x/tos/platforms/telosb/UserButton.h`

Concepts explored by this exercise are:

- Modules - the implementation of a functional element that can be composed into a larger elements through configurations. This file, with a name ending in either 'C' (for common) or 'P' (for private) is an example module (e.g. `PushAndToggleC`). Modules contain state (variables), internal functions, functions that implement one side of an interface, and tasks that provide logical concurrency (not demonstrated in this application). The implementation of a component is in terms of the namespace provided by its interfaces.
- Configuration - a component that composes a set of components by wiring together their interfaces.
- Commands - externally accessible functions that can be called across an interface between components, typically to initiate actions.
- Events - externally accessible handlers that can be signaled across an interface between components, typically to notify of an occurrence.
- Interfaces - bidirectional collections of typed function signatures for commands and events. This module uses three interfaces provided by lower level subsystems. See `tinyos-2.x/tinyos/tos/interfaces/`

Exercise 2: BlinkAndCount

This exercise is designed to get you familiar with using the Timer subsystem in TinyOS. Your task is to take the skeleton application found in the `BlinkAndCount` directory and implement logic to toggle the red LED (`led0`) whenever the user button is pressed, while simultaneously blinking the green LED (`led1`) every second. After you've gotten this part working, modify the application to cause pressing the user button to rotate the leds through a counter among 0, 1, 2, and 3, with the following properties:

- At 0, no timer is fired.
- At 1, Timer 0 with period 1s is fired.
- At 2, Timer 0 and Timer 1 (period 2s) are fired.
- At 3, Timer 0, Timer 1, Timer 2 (period 4s) are fired.
- Timer 0 toggles `led0`, Timer 1 toggles `led1`, Timer 2 toggles `led2`.

Note: Multiple instantiations of the same interface can be made by using the 'as' keyword. For example:

```
interface Timer<TMilli> as Timer0;
interface Timer<TMilli> as Timer1;
```

Concepts explored by this exercise are:

- Timer a critical subsystem TEP102
- Virtualized resource provided as a parameterized interface.

Exercise 3: Time To go to the TinyOS Tutorials

Now that you've got the basics down and have had the chance to ask me questions, its time to move onto the actual TinyOS tutorials on the TinyOS documentation wiki. You should be ready by this point to start working from Lesson 1.3 onwards.

The URL for the TinyOS tutorials is:

`http://docs.tinyos.net/index.php/TinyOS_Tutorials`

Try and see if you can make it through lessons 1.3-1.7 before the end of the lab today. I'll be around to answer any questions you may have.