Lab1 : Introduction to TinyOS and NesC

NesC is an extension to the C language to program WSNs. Main difference is the addition of interfaces, a concurrency model and the removal of dynamic memory. Interfaces allow the creation of components: modules and configurations.

Exercise 1: Read Lesson 1 in the TinyOS Tutorial to learn about the special features of TinyOS and NesC. Stop when you reach the instructions for compiling. Instead follow the instruction below to compile and run the Blink application.

1. Make yourself familiar with the TinyOS directory tree. It is found under /opt/tinyos-1.x/ as follows:

apps TinyOS applications, many examples
doc Documentation
tools TinyOS simulation tools
tools/java Base directory for all TinyOS java packages
tos
tos/interfaces Source for all interface declarations
tos/lib Libraries
tos/platform Platform specific sources
tos/platform/pc The platform for TOSSIM simulation
tos/system Source for system interface implementations
tos/types A few system wide definitions and types

To compile the Blink application, open a terminal at VMware Server Console (see http://klueska.doesntexist.com/installing_xubuntu_vm.html for details). Now change to the Blink application directory and simply issue make pc to build the application, or make pc docs to auto generate documentation for the application. Documentation will end up under doc in the directory tree. Note that the NesC compiler automatically finds and compiles all modules directly or indirectly used by the application, both when compiling and when generating documentation.

To run the application, issue DBG=led build/pc/main.exe 1. This will activate the debug messages for leds and start simulating one mote. You will see messages when the LEDs turn on and off. Issue ./build/pc/main.exe -h to get a list of available debug modes and other options. You can enter multiple debug modes as a comma separated list: DBG=usr1,led,am,crc. The simulator can simulate any number of motes, but only with the same program on each. Try to simulate with different number of nodes. Try other debug modes.

Start the simulation with the -gui flag. Then open another cygwin window and issue tinyviz. This will start a graphical interface to the simulator. You can start and pause the simulation using
the play button, and adjust the simulation speed with the delay dial. Hit play and you should see the mote LED blink. You will have to be gentle with the tinyviz application, or it might hang. Always pause the simulation before doing anything else. To quit, pause the simulation and select quit from the file menu. Then hit CTRL-C in the other window to also stop TOSSIM.

**Exercise 2:** Modify Blink application to display the lower three bits of a counter in the LEDs.

**Exercise 3:** Study the given labs/BlinkTask application and the Timer interface. The Timer can be started with TIMER_REPEAT and TIMER_ONE_SHOT. Modify the BlinkTask application to blink with 1s on-time and 9s off-time! You should do it so the timer only fires when needed, that is: only when it is time for the light to change! Compile and test your solution.

**Exercise 4:** Read Lesson 2 in the tutorial. It will provide you with more details how to use and wire components. To compile and run, apply what you learned in Exercise 1 above. Try the adc debug flag. If you learn how to browse and read the many example applications provided with TinyOS, you will be able to find many hints and examples for your later assignments. Find and look at the source for the BlinkTask application module, BlinkTaskM.nc. As you remember, by convention, module implementations have filenames ending in M. Find the interface specification for the used interfaces. How many commands are provided by the Leds interface? List the commands and events that the Timer interface provides!