Sensor Networks Lab (PR WS14/15)

Lab 2: TinyOS Basics

Sanjeet Raj Pandey [313631]
Sanjay Santhosh Kumar
Muhammad Tauseef Khan
Rajibul Alam Khan

Supervisor:
Vlado Handziski
Tim Bormann

Telecommunication Networks Group
Technical University Berlin

Excercise 2.1: *LED Flash* Write an application that uses the introduced interfaces to let a LED flash with a given frequency. This means that the on-time of the led is shorter than the off-time.

In this exercise, we are using two timers. Timer0 is 25msec and Timer1 is 1000msec. When Timer0 is fired the led is turned off and when the Timer1 is fired then the led is turned on. In this way, the on time is shorter than the off time since the Timer0 is fired much often than the Timer1.

Excercise 2.2: *Binary Counter interface*

The attached zip file contains four files for this exercise, BinCounter is the interface, BinCounterP is the module that that provides the interface BinCounter and BinCountertestP is the module that uses the interface BinCounter. This 3-bit counter runs four times and then stops.

Excercise 2.3: *Hardware peripherals* To which pins are the LEDs connected, and how does the software control these pins?.

The LEDs LED1, LED2, LED3 are connected to the pins 48, 49, 50 respectively. The LEDs are connected in series to a resistor so that the current is regulated in the circuit and the microcontroller is protected. When in an application we call the command to turn 'ON' and LED the respective pin is made to '0' from its default state of '1'. This is quite clear from the schematic of the board. And when the application calls the command to switch off the LEDs the pin is made ('1') and the LED goes to OFF state.

Excercise 2.4: *nesC and C* What are the main differences between nesC and the C language? You may use the blinking LED example to explain it.

There are three main differences between nesC and C programming.

1. nesC uses components and C uses files instead. This makes nesC programming more modular. In the blinkingLED example, both "BlinkC" and "BlinkApp" are components.

2. Components contain the interfaces that they either provide or use. These interfaces are very different from C interfaces. Interfaces in nesC describe the relationship between different components. An interface declaration consists of one or more functions (commands and events). The implementation of these commands and events depends on whether a component provides that interface or uses that interface. Interfaces simplify the code i.e. many components can use this standard pattern (interface) to interact with other components. It also provides code reuse. In the

blinking LED example, the "BlinkC" module uses Timer, Boot and Leds interfaces and through these interfaces it utilizes the commands and events that exist in these interfaces.

3. nesC provides wiring to make components interact with each other. This is different from C where the declarations and definitions must have the same name. In the blinking example, the configuration file "BlinkAppC" contains the wiring information which is given by "→ " symbol. The wiring information is given below.

```
BlinkC -> MainC.Boot;
BlinkC.Timer0 -> Timer0;
BlinkC.Timer1 -> Timer1;
BlinkC.Leds -> LedsC;
```

It says that BlinkC is connected to the MainC through the Boot interface provided by the MainC. And the BlinckC is further connected to the Timer0, Timer1 and LedsC through the Timer0, Timer1 and Leds interfaces respectively.

Excercise 2.5: *TinyOS readability* In TinyOS, applications tend to be composed of many small components. This can make an application quite difficult to understand at a first read. Why is the component architecture used anyway?

The reason for such an architecture being used in TinyOS is that

1. It has an extremely small foot print.
2. It has an extremely low system overhead.
3. It has extremely low power consumption. The TinyOS architecture is designed in such a way to reduce the memory usage and the system overhead.