Wireless Embedded Systems and Networking

Lab Day 5: Part 1: TinyOS Programming on Open Source Distribution

Jaein Jeong

University of California, Berkeley

How to get TinyOS open source dist.

- **Latest version of TinyOS is TinyOS 2.x and available in several formats.**
- **Linux Platform**
  - **XubunTOS Live CD (To be used in this lab):**
    - » **Bootable Linux, no installation needed.**
    - » **Recommeded for classroom environment.**
  - **Debian or Ubuntu Linux:**
    - » **Easy installation using TinyOS Debian package.**
    - » **Recommended for development environment.**
  - **Other Linux:**
    - » **Manual installation is needed.**
- **Windows Platform**
  - **Manual installation is needed.**

Organization of Lab 5

- **Run TinyOS live CD on each machine.**
  - **Insert a live CD in the CD drive and boot the machine with it.**
- **Copy the skeleton projects.**
  - **Skeleton projects will be distributed electronically.**
  - **Copy each of skeleton projects in directory $TOSROOT/apps**
- **Build the skeleton projects and try them.**
  - **Go to the directory of each skeleton project.**
  - **Type 'make telosb install' for building an image.**
  - **Type 'make telosb reinstall.<node id>' for program loading.**
  - **Optionally, run java client for user interaction.**
- **Produce solutions based on skeleton projects and idea for extension.**

Key Ideas of Each Project

	TinyOS Concept	Start	Produce
1	module, configuration, command, event	**1_Push**	**1_Toggle**
2	generics, virtualized services	**2_Blink**	**2_Count**
3	network debugging	**3_PrintSerial**	
4	sensing, split-phase, parameterized	**4_Single**	**4_Dual**
5	post, task	**5_Raw**	**5_Smooth**
6	sending radio message	**6_Counts**	**6_Readings**
7	receiving radio message	**7_Request**	**7_RequestSample**
8	sending / receiving radio message	**8_CountToRadio**	**8_RadioToCount**
9			
10			

Project 1: Push and Toggle

Push - a minimal TinyOS 2.0 application

Behavior: While the user button is pressed the red LED (led0) lights.

Concepts Illustrated:

module - the implementation of a functional element that can be composed into a larger elements through configurations. This file, with a name ending in 'M' is an example module. Modules contain state (variables), internal functions, functions that implement one side of an interface, and tasks that provide logical concurrency (not shown here). 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 and interface between components, typically to notify of an occurance.

interfaces - bidirectional collections of typed function signatures for commands and envents. This module uses three interfaces provided by lower level subsystems. See $TOSROOT/tinyos/tos/interfaces/

Project 1: Push and Toggle

- **Push**
  - **Behavior: While the user button is pressed the red LED (led0) lights.**
- **Toggle**
  - **Behavior: When the user button is pressed, the red LED (led0) is toggled.**
- **Key point**
  - **Notify.notify() event is triggered whenever the user button is either pressed or released. Change the logic of the event handler so that red LED is toggled each time the user button is pressed.**

Project 2: Blink and Count

- **Blink**
  - **Behavior: Pressing the user button toggles the red LED (led0). Green LED (led1) blinks every second.**
- **Count**
  - **Behavior: Pressing the user button rotates a counter among 0, 1, 2 and 3.**
  - **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.**
- **Concepts Illustrated**
  - **Timer – a critical subsystem – TEP102**
  - **Virtualized resource provided as a parameterized interface.**

Project 2: Blink and Count

- **Key Point**
  - **Instantiating a virtualized resource.**
    - » **“interface Timer<TMilli> as Timer0;” instantiates a milli-second timer among different timer types.**
  - **Instantiating multiple interfaces.**
    - » **interface Timer<TMilli> as Timer0;**
      interface Timer<TMilli> as Timer1;
      interface Timer<TMilli> as Timer2;
    - » **Access each timer as Timer0, Timer1, or Timer2.**

Project 3: PrintSerial

PrintSerial - A TinyOS application that shows the concept of network debugging.

Behavior: When the user button is pressed, a counter variable is set to (counter + 1) mod 10. After that, a debugging message "Hello <counter>\n" is sent over the serial port (UART).

In order to see the debugging message, plug the mote to a serial port and run the listen client by typing 'source ./run.sh'. The listen client reads a message from the serial port and prints it on console.

Concepts Illustrated:

Mote-PC serial communication - TEP113
Tinyos 2.0 tutorial lesson 4

Project 4: Single and Dual

Single - A TinyOS application that shows the concept of sending.

Behavior: On boot, a timer of 1s period is started.

Each time the timer is fired, sampling of the node voltage is requested. When the node voltage reading is available, this program sends the voltage reading over serial port. This program also allows sending an error message and the temperature reading (to be used at 'Dual' application).

This program samples only node voltage.
If you want to extend this program to sample node internal temperature as well, wire DemoTemperatureSensorC.nc module to configuration module (SingleAppC.nc).

In order to see the debugging message, plug the mote to a serial port and run the listen client by typing 'source ./run.sh'. The listen client reads a message from the serial port and prints it on console.

Concepts Illustrated:

ADC and split-phase - TEP101
- Tinyos 2.0 tutorial lesson 5

Mote-PC serial communication - TEP113
- Tinyos 2.0 tutorial lesson 4

Project 4: Single and Dual

- **Key Points**
  - **Sampling sensor value is handled in a split transaction in TinyOS.**
    - » **At first step, the application requests the sampling by calling read().**
    - » **When sampling is done, the system notifies the sensor reading as readDone() event.**
  - **When sampling multiple sensor readings, cascade the sensing request and event processing.**

Project 5: Raw and Smooth

Raw - A TinyOS application that shows the concept of task.

Behavior: This program samples node voltage whenever timer is fired. When MAX_READINGS samples are collected, this program report the samples over the serial port. The sampled data can be further processed in a task.

'Smooth' application extends this application as follows:

(1) calculates the statistics: 'Smooth' calculates the minumum, maximum and mean for sampled data in raw_reading[].

(2) smoothens the sampled data: 'Smooth' calculates the exponential moving average of raw_reading[] into smooth_reading[].

In order to see the debugging message, plug the mote to a serial port and run the listen client by typing 'source ./run.sh'. The listen client reads a message from the serial port and prints it on console.

Concepts Illustrated:

Schedulers and Tasks	**- TEP106**
	- Tinyos 2.0 tutorial lesson 2
ADC and split-phase	**- TEP101**
	- Tinyos 2.0 tutorial lesson 5
Mote-PC serial communication	**- TEP113**
	- Tinyos 2.0 tutorial lesson 4

Project 5: Raw and Smooth

- **Key Points**
  - **Calculating statistics and exponential moving average can take time.**
  - **It is recommended to process a time consuming job in a task rather than to process directly in the event handler.**

Project 6: Counts and Readings

Counts - A TinyOS application that shows the concept of sending a radio message.

Behavior: 'Counts' program starts a timer at an interval of 1s. Each time the timer is triggered, this program sends a radio message that contains its node ID and counter variable. The counter variable is incremented each time the timer is triggered.

'Readings' extends this program by sampling the node voltage as well.

In order to see the debugging message, a base station node needs be prepared. A base station node is a node that is programmed with 'BaseStationCC2420' program, which forwards all its received radio messages to and from the serial port.

Try program multiple nodes with 'Counts' program and see the behavior with a base station node and java client application. You can see the debugging messages from multiple sensor nodes.

Concepts Illustrated:

Mote-mote communication - TEP111, TEP116
- Tinyos 2.0 tutorial lesson 3

Project 6: Counts and Readings

- **Key Points**
  - **Wire DemoSensorC to the application and get the sensor reading using the previous lessons.**

Project 7: Request and RequestSample

Request - A TinyOS application that shows the concept of receiving a radio message.

Behavior: When 'Request' program receives a radio message Receive.receive() event is triggered. Depending on the contents of the received message, the program can take an appropriate action. Our program simply fills in the reply message with the node id and send it.

The exercise in this lesson is to extend this simple version of program so that it fills the contents of the reply message in response to the user request. The request message can choose any combinations of counter, voltage reading and temperature reading. For this, you need to parse the received message in Receive.receive().

In order to send a message and receive the reply message, a base station needs be prepared. A base station node is a node that is programmed with 'BaseStationCC2420' program, which forwards all its received radio messages to and from the serial port.

Try program multiple nodes with 'Request' program and see the behavior with a base station node and java client application. You can see the debugging messages from multiple sensor nodes.

Project 7: Request and RequestSample

- **Key Points**
  - **In order to take appropriate actions to the request message, you need to parse the incoming message in Receive,receive() event handler.**

Project 8: CountToRadio and RadioToCount

CountToRadio - A TinyOS application that shows the concept of sending a radio message.

Behavior: 'CountToRadio' program starts a timer at an interval of 1s. Each time the timer is triggered, this program sends a radio message that contains its node ID and counter variable. The counter variable is incremented each time the timer is triggered. LED is set to last 3-bits of the counter value.

The goal of this lesson is to write a receiver program 'RadioToCount' program that receives the counter message and sets its LED as last 3-bits of the receiverd counter value.

Concepts Illustrated:

Mote-mote communication - TEP111, TEP116
- Tinyos 2.0 tutorial lesson 3