ICTP-ITU School on Wireless Networking for Scientific Applications in Developing Countries
Lab 01-02 on Embedded Wireless Sensors, Tue/Wed 20/21 Feb 2007
Bhaskaran Raman, Department of CSE, IIT Kanpur, INDIA

This lab uses code/material from the following references:

- TinyOS 2.0 Tutorials http://www.tinyos.net/tinyos-2.x/doc/html/tutorial/
- Moteiv Boomerang http://www.moteiv.com/

We will use the Tmote Sky mote from Moteiv, and the Boomerang software (based on TinyOS 2.0) for this lab's exercises. Below, we shall assume that the software is installed on a PC running WindowsXP. Proceed in the following steps.

1. We will first start with compiling an application called `Blink` and installing it on a mote.
   - Open a `cygwin` window on the PC and go to the directory called `MyBlink`
   - Type `make tmote` and enter. This compiles the program for the tmote platform. Observe that a directory called `build` would have been created. See the contents of the directory `build/tmote` and look for `main.exe`
   - We now need to install this into the tmote. Before that, we need to find out the COM port with which the tmote is associated. For this, use the command `motelist`.
   - Now issue the command `make tmote reinstall,1 bsl,n`. Here, the "1" corresponds to the network address you are assigning to the mote. More important for now, "n" would correspond to a number one less than the number of the COM port you observed in the `motelist` command. For instance, if your PC showed COM4 in the `motelist` command, you would type `make tmote reinstall,1 bsl,3`. You should now see fast blinking LEDs, which indicates transfer of data via the USB interface.
   - After a few seconds, you should see the the red LED blinking periodically. Congratulations! You have installed your first TinyOS application.
2. Look at the source code `MyBlinkC.nc` and `MyBlinkApp.nc`.
   - Observe which components are used and how they are linked together.
   - Also look at the `Makefile`.
   - Look also at the interfaces which you have used, namely Boot, Leds, and Timer2. You will find these files in `/opt/moteiv/tos/lib/sched/` , `/opt/moteiv/tinyos-1.x/tos/interfaces/` , and `/opt/moteiv/tos/lib/timer/` respectively.
   - Observe in `MyBlinkApp.nc` as to which components provide these interfaces.
3. Now you can experiment with the MyBlink application to make the green or yellow (actually blue) LEDs glow, or to change the period of the glow.
4. We are now going to introduce a task in the code. A task is nothing but a subroutine which encompasses some computation. Write a task called `compute()`. You will have to use the keyword `task` in front of the subroutine name, just like the keyword `command` or `event`.
   - Use a timer period of say, 5 seconds.
   - For now, just make the task run a loop for N times, for say N=1000. This is just a delay loop.
   - You can make the green LED on before the delay loop, and off after the delay loop.
   - You have to `post` the `task` in the timer fired event code. Posting a `task` is done by using the keyword `post`, just like a `command` is called using the keyword `call`.
   - You may have to calibrate your delay loop to make the green LED glow on for sufficient time.

5. Now revert to the code without the task. We shall next introduce a variable in the component, called `count` of type `uint8_t`.
   - Where will you initialize this variable?
   - Increment the variable for each timer fired event.
   - Can you now write code to have the three LEDs display the last three bits of the `count` variable at any point of time?
6. You are now warmed up in TinyOS programming. Let us now write code to use the radio. We will modify the Blink application above such that one mote will send its current count variable on the radio, in a message. And another nearby mote will receive this message, read the count variable, and set its LEDs accordingly.
   - You have to make two applications called `MyBlinkRadioSend` and `MyBlinkRadioReceive`. Each will have a configuration file and a module definition file, like earlier.
   - Critical to sending/receiving are the interfaces `SendMsg` and `ReceiveMsg` , both of which are implemented by the `GenericComm` component. You can find these interfaces in the directory `/opt/moteiv/tinyos-1.x/tos/interfaces/` .
   - You will be using a component variable of type `TOS_Msg` to send and receive packets. You can look at the `TOS_Msg` structure in the file `/opt/moteiv/tos/lib/CC2420Radio/AM.h` .
   - You will have to enclose the body of whatever data you want to send within the `data` field of the above `TOS_Msg` structure. You may define a structure of your own, to be enclosed within the data field. You can define this in a header file called `MyBlinkMsg.h` and include it in the relevant other files using `#include` .
   - At the sending end, you have to ensure that you do not issue a send command before the previous one has finished. One way to do this is by using a `bool` variable. Another is by simply having the timer period large enough (say 1 sec).
   - At the sending end, you can make the red LED go on and off in-between the `send` command and the `sendDone` event.
   - For both the `MyBlinkRadioSend` and `MyBlinkRadioReceive` configuration files, you will have to bind the `SendMsg` or `ReceiveMsg` interfaces to `GenericComm.SendMsg[N]` and `GenericComm.ReceiveMsg[N]` respectively, where `N` is say "1". This `N` represents the AM message type and serves to parametrize the GenericComm component.
   - Be sure to give different radio addresses for the sender and the receiver while installing.
   - You can check that the receiver is actually responding to the sender's messages, by temporarily turning off the sender or by holding the reset button at the sender. During this period, you can observe that the receiver's LEDs do not change.
7. Debugging a program with just three LEDs can be hard. We will next learn to dump packets onto a console on the PC. For this, use the `MyBlinkRadioSend` from above. And instead of the receiver `MyBlinkRadioReceive`, use the `TOSBase` program located at `/opt/moteiv/apps/TOSBase` .
   - The TOSBase program receives packets on air and passes them on to the PC via the USB interface, which shows up as a serial port (COM port).
   - Now we need to run a program which reads from the serial port. The `SerialForwarder` java program is one such program which comes with Boomerang. This program reads from the serial port and forwards that onto any client connecting to it on TCP port 9001 (by default). To run this, suppose that the mote loaded with `TOSBase` is connected to COM4.

Now type the command `export MOTECOM=serial@COM4:tmote`. Then type `java net.tinyos.sf.SerialForwarder&`.

- Finally, we have to run a program which connects to `SerialForwarder`, gets the packets and prints them on the console. For this, run `java net.tinyos.tools.Listen`.
- Now you should be able to see the contents of the packets sent by the mote running `MyBlinkRadioSend`. Can you match up the fields with those from `AM.h`?

8. Now we will learn to access sensor readings. For this, you have to use the `ADC` interface as implemented by the `HamamatsuC` module, provided as part of the Boomerang installation. You can wire the `HamamatsuC.PAR` interface to the ADC interface to be used. This will give you the Photosynthetically Active Radiation (PAR) sensor readings.
   - To learn how to use the `PAR` interface, you can either look at the interface source code, or look at the example usage in the `/opt/moteiv/apps/Oscilloscope` example program.
   - You can change the structure of the radio message sent on air to accommodate the sensor readings.
   - Run the sensor program on one mote, and `TOSBase` on another. As earlier, run the `SerialForwarder` and `Listen` programs on the PC.
   - You should now be able to see the sensor readings as part of the message structure printed on the screen. Gently cover the mote, preventing light from falling on the sensor. Can you see the PAR readings change?
9. Advanced-1:
   - Learn to use the `Button` interface as implemented by the `ButtonC` component. This gives an event on press or release of the "user" button on the tmote.
   - You can change the `MyBlink` program to reset the counter each time the user button is pressed (or released after pressing).
10. Advanced-2:
    - You may have noticed that the `TOS_Msg` structure also includes strength and lqi fields. These correspond to the RSSI and LQI values as reported by the receiving radio chip.
    - Modify `TOSBase` to include these values as part of the data received. You will now be able to see the RSSI and LQI values.
    - Move the sending mote away from the `TOSBase` receiver and observe the variation in RSSI and LQI values.
    - The RSSI value printed is just a register value. Look-up the CC2420 radio data sheet to find out the mapping between this reported register value and the actual RSSI in dBm.
    - Modify the `Listen` program to print the actual dBm of the RSSI value instead of the register value.