Program 2 {Modified April 21, 2011} 40 Points

Sensing and Communicating with TinyOS and TelosB Motes

Due: Tuesday, April 26, 2011 at 9 a.m.

Assignment Preparation

Each programming team should obtain two Crossbow TelosB motes from the TA, Jeff Zhou. When you receive your motes, they will be assigned a unique node ID that you need to use for unambiguous identification for program 3. Note, these motes are expensive and your team is responsible for returning the motes in working condition at the end of the term. Moreover, use caution and be aware of static discharge and other things that can "fry" a mote. We do NOT have any replacements!

While a fresh pair of AA batteries will last only a few days if you leave the mote running, remember that your team is likely to be doing quite a bit of testing and reprogramming. Since the TelosB motes do not have a power switch, leaving the batteries in the mote will draw power even if the LEDs are not on. Thus, remember to pop out the batteries when the mote is not in use. The mote can be programmed without the batteries in place because the mote receives power from the USB port when plugged in.

If a mote that was behaving fine at first, later acts erratically (e.g., will not reprogram or radio transmissions are not working properly) that is usually an indicator that you need fresh batteries. You probably will need to buy more batteries (at your team's expense) to complete programs 2 and 3.

{This sensor assignment assumes your group has access to laptop's or home machines where you can install TinyOS.}

1. Install TinyOS 2.1

See the instructions for Installing on Windows or Linux platforms on the TinyOS Community Forum web page:

http://docs.tinyos.net/index.php/Getting_started#Installing_TinyOS_2.1

In Step 3 of installation, use the **TI MSP430 Tools**. If you run into installation problems, you can check the TinyOS help archives or send an email to the class mailing list.

2. Verify installation

The best way to verify your TinyOS installation is to work through Tutorial #1 and get the Blink application running on one of your motes.

Settings before creating applications:

a. Changing your Radio Channel

Since there are six programming teams in the class, you need to set a unique radio channel for each group to avoid packet collisions between group transmissions. The radio for the TelosB has a total of 16 channels, numbered from 11-26. The default setting is 26.

To set the channel, add the following line to the beginning of your **Makefile** before the ***include Makerules*** command (of each application that you create/modify). Replace **x** below with (your group number +10).

CC2420_CHANNEL=**x**

b. Changing the Radio Transmit Power

Since this course involves small groups in single room test spaces, you can save battery life by lowering your motes transmission power. Valid transmit power levels are 1 to 31 with power 1 equals to -25dBm and 31 equals to max (0dBm).

To set the transmit power, add the following line in the **Makefile** before the ***include Makerules*** command:

CFlags = -DCC2420_DEF_RFPower= **y** where **y** is in the range [1-31].

A reasonable setting for experiments in this course would be:

CFlags = -DCC2420_DEF_RFPower= 3

3. Work Through a Few Tutorials

Before embarking on writing the program, members of your team should work through Tutorials 2-6 and 13.

Assignment

4. Write a simple nesc program to display the value of the light sensor on two motes. ***(Note this assignment was originally designed by Matt Walsh at Harvard.)***

Your assignment is to write a program in which the status of both motes' light sensors are shown on the three LEDS of both motes. When the light sensor on an individual mote is below a threshold, the respective LED on both motes should be off. One mote will effectively be the Red mote and the other will be the Green mote. When an individual light sensor reads above some light threshold value, the appropriate Red or Green LED should be lit on both motes. If both the Red and Green LED are lit, then also light the Blue LED (this LED is a Boolean AND of the other LEDS) on both motes. When the light sensor readings are below the light threshold, the respective LEDs should be off.

Light Sensor

There are two light sensors on the TelosB motes. A "total solar radiation (TSR) sensor and a "photosynthetically active radiation" (PAR) sensor. You are to use only the TSR sensor. Since this sensor is fairly sensitive, a fairly low threshold is needed (between 30 and 50) to distinguish between "light" and "dark" in your application.

This program is fairly straightforward. The Red mote should periodically sample its light sensor once per second and the Green mote's sampling rate should be once every two seconds. Your program should test for the Blue state every half second. Each mote broadcasts a radio message to the other mote indicating a change in the state of its light sensor.

Additionally, on every state transition, the Red mote should **communicate serially** with the PC indicating the current state of both mote LEDs. The PC should store these mote LED states for a one minute experimental period. After the final state info is sent to the PC, the PC then prints out the state history of the motes and indicating the total number of times each mote was above its threshold.

Deliverables

Each team will have to schedule a demo with the TA to show that your two sensors operate correctly. Use turn in to provide a tarred/zipped file with all the source code and the Make file. Provide a README file that explains anything unusual about your implementation.