Lab Two

Purpose:

In this Lab the student will develop a sensor network application which senses the environment and stores the sensed data locally, in the LiteOS file system. Using simple commands sent from a PC, the stored data is retrieved back. The sensor nodes can be dynamically configured by two ways, a configuration file and a debugging interface (optional).

In this lab three MicaZ motes will be used. Two motes (NodeA and NodeB) will sample environmental data, and one mote will be used as a Base Station. Two types of sensing will be supported: light and temperature (using the code developed in Lab One). A new configuration file (type of sampling, sampling period, number of samples) is used to control the sampling application behavior.

After the lab the student should be able to understand the following:

- How to use the external file system for storing sensed data.
- How to retrieve data from a sensor network through the copy command.
- How to use MessageCenter for visualizing data.

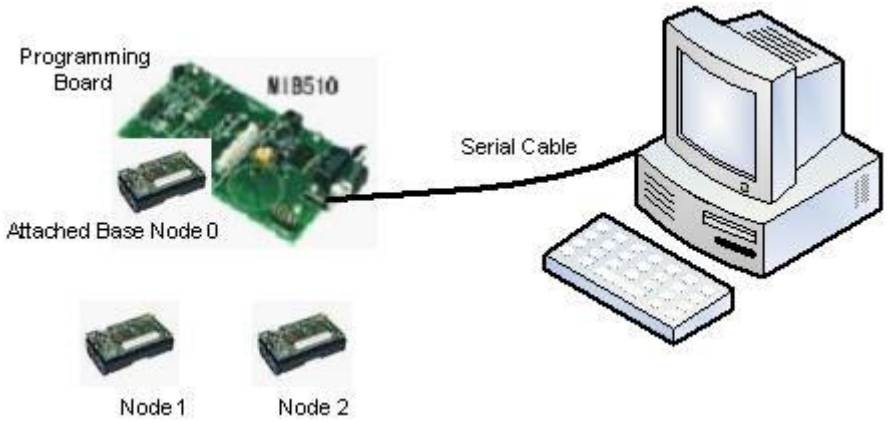
Pre-lab Suggestions:

- Review Lab One.
- Read and understand the file system API in LiteOS programmer's guide on how to use the file system. Understand that the file system uses EEPROM and external flash to store control structures and data. Read the code sample in LiteOS programmer's guide to get familiar with how to use it.
- Write and compile the code for the Sensing Node following the instructions below.
- It turns out that the latest version of GCC, 4.2.2, for WinAVR 20071221 version will cause problems for the lab 2 experiments. Please avoid use this newest version for
  now, and instead use WinAVR 20070122 (AVR GCC 4.1.1) instead. All WinAVR versions are available at
  http://sourceforge.net/project/showfiles.php?group_id=68108, and one change that is probably needed is to make sure the makefile in the Apps directory be modified to point to the right version of avr-gcc.exe. This bug does not cause problem for lab 1, so now lab 2 should be easy to finish.

In-lab Procedure:

[Setup]

1. This lab uses 3 MicaZ motes, one is used as the Base Node, and two are used as Sensing Nodes. The setup is shown in the figure below.
2. Note that LiteOS has been installed on these motes. Now connect to the LiteOS system using the LiteOS terminal.
3. Compile your sensing application, and install it on the two Sensor Nodes (NodeA and NodeB), following similar steps in the previous two experiments.
4. Start MessageCenter to view radio messages.



[Instructions]

1. Requirements:

   1.1 At any point in time, only one type of data (either light or temperature) shall be sampled by a sensor node. The sensed data shall be stored in the file system.

   1.2 The default sensing (when a mote is turned on) is: light sensing, 1 second sampling period, 30 samples. After 30 samples, the mote stops sensing.

   1.3 The sensing application shall be configurable, through a configuration file stored at the root directory on the local node. The configuration file will contain: the type of sensing (light or audio), the sampling period, and the number of samples. The configuration file format is given below (Section 2).

   1.4 You may start and stop the application using exec and kill commands. Remember to delete the old data file to avoid conflicts. Before you start your application, copy the configuration file from PC to the node, and then start the

application, so that a new sampling cycle starts, and the new sampled data shall be stored into the local file system. You may choose a file name you want.

1.5 At the end of each sampling cycle, retrieve the data back by using the cp command. For example, suppose you have a data file named mydata, do the following:

```
cp mydata /c/Temp/mydata
```

1.6 If you are faced with a decision that is not specified by the above requirements, chose the option that is the easiest for you.

1.7 Remember you can always use ps command to make sure that you have successfully started the application on the mote. Remember if things go wrong, you can always restart the mote to rollback to a clean slate state.

2. Programming the Sensing Nodes (NodeA and NodeB) using C:

   In this step, you are expected to implement the sensing application (modify Lab one), following the above requirements. You shall use the following data structures:

```
typedef enum {
    LIGHT_SENSING = 1,
    TEMP_SENSING = 2
} Constants;
```

   The configuration file format:

   Sampling Type (1 byte)
   Sampling Period (2 bytes)
   Sampling Count (2 bytes)

   You may specify the exact semantics (data type and endianness) yourself.

3. Testing Scenarios: a few testing scenarios are outlined below (others may be used):

   Start the application without a configuration file (and it should choose the default behavior) for 30 seconds. Then stop it with kill command. Retrieve the data back by copying the data file back to PC. You should get 30 samples in this file.

   Copy a configuration file to the mote. Set the configuration for 100 samples, at

a frequency of 2 samples per second. Start the application. You should get 100 samples at the end of the sampling.

4. Remove your application from all the motes. Unplug the motes and turn them off.

The following steps are optional.

5. Read the debugging guide in the programmer's manual of LiteOS and understand how to view and change variable values at application runtime.

6. Set the parameters in your sensing application (its number of samples to read and reading frequency) as global variables. Recompile the application and install it. View these variables after you start the application.

7. Write your application in such a way that you have a global variable called EnableSampling. If this variable is set to be 0, then your application does nothing. If this variable is set to be 1, then your application will start sample and record data into a local file. Your application wakes up periodically to check this variable to decide whether starting sampling. Use the debugging commands (debug, list, print, set) to change EnableSampling after you start your application to decide its behavior.

8. Use the device directory to directly retrieve data. For example, to read the light sensor 100 times at 1 second per sample, use the following command:

   cd device
   ./light 1000 100 (the first parameter is the interval in milliseconds, the second is the total number of samplings)

After-Lab Deliverables:

Submit Following items

Source Code for the Sensing Node (the same code runs on both Node1 and Node2) (4 Points)

Your configuration file (one file is enough) (2 Point)

Source Code for the Sensing node when using the debugging interface (2 Points)

Screen snapshot of using the device directory to retrieve data (2 Points)