Lab One

Purpose:

This Lab allows the student to build a simple sensor application that takes multiple light intensity readings (from the sensor board's photo sensor on MicaZ motes), reports wirelessly to the base, and displays those readings on the PC using the MessageCenter Tool. After this lab, students should be able to learn the following items.

- Single Hop Wireless Communication
- Simple Data Buffering and Aggregation
- Using MessageCenter Tools to parse and display the incoming messages
- Using the oscilloscope Java GUI to visualize the incoming messages

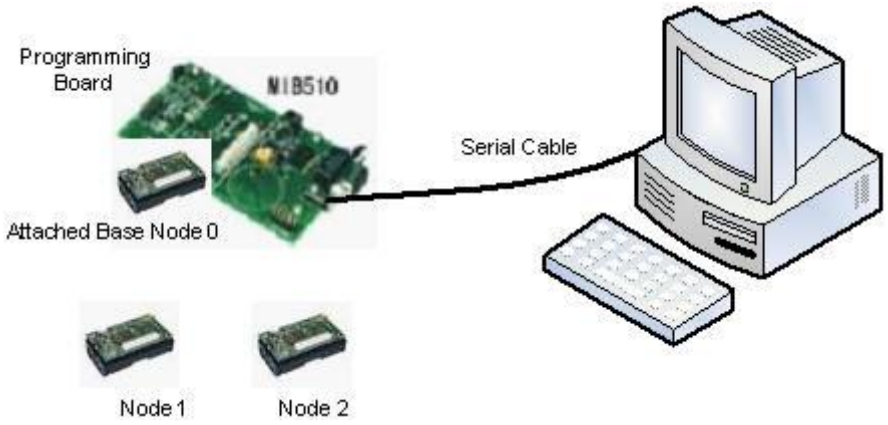
Pre-lab Suggestions:

- Learn how to broadcast a message: A very simple application that broadcasts the counter value periodically. The source code can be found at Apps/Apps/SenseAndSendOscilloscpe
- MessageCenter is a useful tool to snoop the messages from a sensor network as well as to inject messages into a sensor network. Read carefully this page before starting this lab.
- Write and compile the code for the Sensing node.

In-lab Procedure:

[Setup]

1. This lab uses 3 micaZ motes, one is used as the base mote and two others are used as the sensing motes. The LiteOS system should have been installed for both the base node as well as the sensing nodes. As you have learnt in the previous lab, these two sensing motes are named as nodeA and nodeB, respectively. The base mote should be attached to the programming board (see figure). Two sensing motes should be within 3 meters from the base to ensure a good link quality.
2. The programming board should link to the PC through serial cable (DB9).



[Instructions]

1. Programming the Sensing Node using C:

   In this step, you can build your code based on the skeleton code provided in SenseAndSend. The current code only senses the light sensor at a fixed interval. You need to read the radio communication API provided by LiteOS programmer's guide to add a communication module so that Node A and Node B can broadcast their readings in batches to the base station.

2. Requirement: The format of the broadcast message should be:

```
struct ReportMsg {
    uint16_t sourceMoteID;          //ID of the sending node. e.g. node 1 and
node 2
    uint16_t lastSampleNumber;      // The sequence number of the last
readings  within data[BUFFER_SIZE]
    uint16_t channel;               //For now this value is fixed as 1.
    uint16_t data[BUFFER_SIZE];  //BUFFER_SIZE = 10.  Put 10 readings
together
};
```

   Note: this message should be the payload within the LiteOS message structure, declared at amcommon.h.

3. Revising the sensing node code to add simple aggregation.

   In this step, you are expected to change the source code of your previous application, so that the node aggregates every 2 consecutive packets from the same node into one packet. The aggregation algorithm can simply be taking the average of two consecutive readings (i.e. For example, if 10 raw light intensity reads are 100, 102,104,106,108,110,112,114,116,118. You will aggregate them into 101,105,109,113,117).

The format of the aggregated message sent out should be the same as the previous message. lastSampleNumber of the aggregated message is the lastSampleNumber of the second broadcast message.

4. Parse the messages using MessageCenter

5. To start the MessageCenter, from a Cygwin shell window, execute the following:
   > java tools.mcenter.MessageCenter &

   In the "Serial Connector" window, select Remote Server with port 9001. Then click on "Start Port". This will start the communication with the base mote.
   In the "App Loader" window, enter "**tools.mcenter.MessageTable**" and click the "**LoadApp**" button. You may also find useful to load "**tools.mcenter.AllMSGDisplay**".
   Double-click the "Message Table" entry in the "App Loader" window. This will pop-up another window where you need to define the message format (see instruction here). The message format that you need to define is the one given in Step 1, above. Message Center will use this definition to correctly parse the message. (no java programming needed)

6. Visualize the messages: use the oscilloscope java program to display the results. You need to use the correct data format in Step 2 to see results on the oscilloscope. (no programming needed)

   > java tools.oscope.oscilloscope &

7. Remove your application from all the motes by unplugging the motes and turn them off.

After-Lab Deliverables:

⊕Submit Following items:

Source Code for the Sensing Node (the same code runs on both NodeA and NodeB) (**4 Points**)

Source Code for the aggregation program on the sensing node (**2 Points**)

Screen snapshot of the MessageCenter results (MessageTable window) (**2 Point**)

Screen snapshot of the oscilloscope results (**2 Point**)