Laboratory assignment two
Introduction to Motes
ECE 397-1

Lab checked on or before 20 January
Lab report due during lab check
Prepared by Robert Dick

Please keep track of how long you spend doing this laboratory assignment. Specifically, how much time is needed to do the problems after studying enough to understand the concepts?

In this assignment, you will be programming a two-node sensor network. One node will serve as a bridge to RS232 port attached to a workstation. The other node will gather and transmit light and temperature readings. In addition, it will detect noise, and sound its buzzer in some situations.

1 Mote access

I ordered numerous motes well before 10 December. Unfortunately, Crossbow is moving this month and still hasn't shipped. For this lab (and only this lab), you'll need to pick up a pair of motes from Ashish when you start working and return them when the last student leaves the lab. When the motes arrive on 20 January, I'll distribute them.

2 Walk-through

1. Log in at a tlab machine and update from the

   ```
   svn+ssh://tlab-login.cs.northwestern.edu/home/pdinda/HANDOUT/SVN_RTW04
   ```

   repository using subversion. Within the motes directory, the tosbase directory contains code for a MICAZ Zigbee to RS232 bridge. The xlisten directory contains code for displaying messages arriving on a workstation's serial port. The mts300 directory contains code to periodically sense the state of a MICAZ's microphone, light sensor, and thermistor, transmitting the information via its Zigbee wireless interface.

2. Make sure /usr/local/unison/bin is in your path and run

   ```
   repos-init-all
   ```

   . That initializes some useful links (a stop-gap until subversion handles symbolic links). It's probably best to have /usr/local/unison/bin in your path before /usr/bin. You'll need to use /usr/local/unison/bin/make instead of /usr/bin/make.

3. Add /usr/local/unison/lib to your LD_LBRARY_PATH environment variable. You need this so the new shared libraries can be found.

4. You won't need to change the code in the xlisten directory. Let's build that first. Go to the xlisten directory. Then build, and run xlisten.

   ```
   cd xlisten

   make
   ```

```
./xlisten
```

That should start xlisten monitoring your workstation's serial port for sensor messages. There aren't any messages now so you can terminate the program by typing Control–C.

5. Go to the mts300 directory and read the TestSensor.nc and TestSensorM.nc files. Type

```
make micaz
```

You'll see a number of warnings. In your lab report, describe the reasons for these. You'll be able to find more information in D. Gay, P. Levis, D. Culler, and E. Brewer, "nesC 1.1 language reference manual," May 2003.

6. **Make sure that the batteries are removed from the MICAZ** and its sensor board is disconnected. If you ever attach a battery-powered MICAZ to a DC-powered MIB510, you'll break them both.

7. Plug the MICAZ into the MIB510 programming board.

8. Attache the MIB510 to your workstation's serial port.

9. Again, **make sure that the batteries are removed from the MICAZ**.

10. Attach the MIB510's power supply.

11. Type

```
make micaz install
```

That should program the MICAZ as a sensor node.

12. Disconnect the power cable, attach the MTS300 sensor board to the bottom of MIB510, and reattach the power cable. It should beep and start flashing. Remember that. *The MTS300 code tells the mote how to beep.*

13. Run xlisten and look at the output. The mote is sensing temperature, light, and sound, then transmitting this information via RS232 and Zigbee.

14. Disconnect the power cable, remove the MTS310 and MICAZ from the MIB510, then attach them directly to each other.

15. Add batteries and turn on the MICAZ. It should beep. Turn it off for now. Set it down with the batteries facing down, not with the sensors facing down.

16. Go to the tosbase directory. Read the TOSBase.nc and TOSBaseM.nc files. You needn't understand every detail yet.

17. **Verify that the other MICA has no batteries**, attach it to the MIB510, and connect the DC power supply.

18. `make micaz install`

19. Turn on the sensor mote and run xlisten. The sensor mote should be transmitting data to the RS232 bridge mote, and ultimately to your workstation.

20. A thermistor is a temperature-sensitive resistor. Use the xlisten and your clean, dry finger to figure out where the MTS300's thermistor is located.

21. Note that the microphone values are pretty useless for detecting how loud noise in the mote's environment is. We'll fix that soon!

3 Software noise detection

1. Modify the TestSensor code so that it senses and transmits with a period of 250 ms.
2. Add a new timer with a frequency of 40 Hz to the application. You can easily provide interfaces to multiple timer instances. Please refer to the nesC reference manual. However, you'll find the following syntax useful.

   ```
   interface Timer as Timer1

   interface Timer as Timer2
   ```

   Please note that you'll need to change both the TestSensorM.nc and TestSensor.nc files.
3. Cause the green light-emitting diode to toggle every 25 ms.
4. Increase the frequency of microphone sampling events to 40 Hz but do not increase the frequency of Zigbee transmissions. Instead, change the application so that the raw microphone analog to digital converter (ADC) value at discrete time step t is r_t, each Zigbee transmission sends a value approximately equal to

   $$N = \sum_{t=1}^{39} |r_{t-1} - r_t|$$

   in the microphone field. In other words, send the number and magnitude of changes to the raw microphone ADC since the last transmission. You can now use xlisten to get a rough idea how noisy the mote's environment is.
5. Change the application so that the sounder is activated if $N > 1200$.
6. Clap!

4 Deliverables

Please prepare a brief lab report containing the following information. A page should be sufficient for this assignment.

1. A brief explanation of nesC's *atomic* keyword.
2. A brief explanation of the reasons for the warnings in the original mts300 code.
3. An illustration of the location of the thermistor.
4. A paragraph or two describing the design of the TestSensor application and your modifications to it.

Please visit Ashish during his office hours to have the assignment checked. He'll watch you program and test pair of sensor nodes and collect your code.

5 Ungraded question

1. Do nescc's warnings for the mts300 code point out genuine problems? If not, why not. If so, how should the code be changed?
2. How long did you spend on design
3. How long did you spend on implementation