Laboratory assignment four
Introduction to Motes
ECE 397-1

Lab checked on or before 3 February
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 designing a multi-hop Zigbee network that senses and transmits audio data to a pocket PC via Bluetooth.

1 Goal

The first motes assignment was a walk-through with detailed instructions on programming and using the motes. This assignment will explain requirements at a much higher level. This assignment will require more thought than the lab two.

You may start from the code supplied in laboratory assignment two. In addition, please see the Promi SD101 manual, ATmega128 manual, and other reading material posted to the website to help you with this assignment.

You will also find it extremely useful to look at the source code in the /usr/local/unison/packages/tinyos directory of the tlab machines. You can recursively search for files containing "fred" in nesC files using the following command:

```
find . -type f -name \*.nc -exec grep -il fred {} \;
```

You may find files with a particular name using the following command:

```
find . -type f -iname \*fred\*
```

For more information `man find` and `man grep`.

Please do the following:

1. Write the code for a node that repeatedly does high-rate audio data capture until 3000 bytes have been buffered, halts audio capture, and then transmits the buffer contents using a series of TinyOS messages.

   You can achieve 3 kHz without heroics. However, even higher rates are possible by using direct access to the analog to digital converter (ADC) instead of using the *MicC* component. You'll find MicADC's getContinuousData() command useful. You may halt ADC reads by returning *FAIL* from dataReady().

   If you decide to push the hardware to its performance limits, please see page 246 of the ATmega128 reference manual. Researchers have found that, at a sample rate of approximately 6 kHz, jitter and EEPROM write speeds prevent further increases. If you need to rate-limit wireless transmission, you can trigger transmissions with a timer that is deactivated when the audio buffer has been emptied.

Note that you may use a single message to carry multiple bytes and that you need not conform to the enum-based formatting used in the MTS300 TestSensor application. In addition, you may simplify your life by discarding the low-order two bits of the 10-bit ADC samples.

2. Write the code for a repeater node that retransmits any message it receives. You are encouraged to build debugging indicators into your software. For example, the repeater node might blink its LEDs when receiving and retransmitting.

   This part is straightforward. However, it sets the stage for the extra credit problem and the next motes lab assignment. You'll find that it is useful to either tag repeated messages or use walls to prevent the RF signal of the sensor node from reaching the Bluetooth node.

3. Write the code for a base node that uses a Promi SD100's AT commands to open itself to connection from a PPC. Wait until a connection is established. Then send incoming raw (multi-hopped) audio sample bytes to the RS232 interface.

   Please note that the standard UART component does not support byte writes. You'll find HPLUARTC more flexible. It will allow you to quickly develop an interrupt-based string transmission routine.

   You'll need to ensure that the MICAZ and Promi SD101 have the same baud settings. You'll find the source code in HPLURT0M and page 128 of the ATmega128 manual particularly useful for this. It is also possible to change the Promi SD101. They should be set to 9600 baud now. If you want another value, please email me. The *AT+BTSCAN* command will be useful to open the Promi SD101 to connections from other Bluetooth devices. When a connection is established, *CONNECT* will be transmitted by the Promi SD101.

4. Capture and play the audio data on the PPCs. This part need not be fully automated.

5. At this point, you should have a multi-hop network that streams microphone data from a sensor node to a PPC. You'll have problems with network contention if you use numerous repeaters.

2 Extra credit

1. Devise and implement a method of determining which messages should be repeated in order to minimize network load. You may assume a tree-like network structure in which two message types exist: those destined for the root node (ID 0) and those destined for all nodes. The second type of message will later be used to send commands to the motes.

   It may be useful to have the root node periodically send a beacon message allowing other nodes to determine how many hops they are from the root node. This information may be used to have repeaters selectively drop messages originating from nodes that are closer to, or the same distance from, the root node.

2. Filter the audio data to improve its quality.

3. Develop and implement a method of detecting unusual audio profiles within the Zigbee network so nodes can determine whether streaming is justified. The most basic approach may be a simple noise detector similar to the one you developed in the second lab assignment.

3 Deliverables

Please prepare a lab report describing your design and implementation decisions and pointing out anything unusual you learned while completing the assignment. In addition, please suggest future directions. The exact course we take with this sensor network will be influenced by your preferences. Please visit Ashish during his office hours to have the assignment checked.