

Sensor Network Lab Exercises Using TinyOS and MicaZ Motes

Jens Mache, Chris Allick, John Charnas, Alex Hickman, Damon Tyman

Mathematical and Computer Science

Lewis & Clark College

Portland OR, USA

jmache@lclark.edu

Abstract - *MIT Technology Review lists sensor networks as one of “Ten Emerging Technologies That Will Change the World” [1]. This paper describes three lab exercises that are suitable for activity-driven teaching of sensor networks to undergraduate students. The exercises are derived from the TinyOS tutorial, and MicaZ motes were used. The first exercise is about uploading a simple blink program to one mote. In the second exercise, several motes use radio communication. The third exercise uses a sensor board to take measurements, and the data is sent back and displayed on a traditional computer. We describe our experiences, lessons and code alterations.*

Keywords: Pervasive computing, sensor networks, education, TinyOS, MicaZ motes.

1 Introduction

In the future, agricultural fields and nursing homes may be fitted with sensor network devices letting farmers know when to water fields, and nurses when to check on something unusual [3]. With such a large potential for applications, it is surprising that only a few colleges and universities have introduced sensor networks into the undergraduate curriculum to date. It is critical that the next generation of computer scientists know how to employ and program sensor networks. Although some tutorials currently exist, their target audience is largely graduate students and professional researchers.

In this paper, we describe three exercises that we deem suitable for the activity-driven teaching of sensor networks to undergraduate students. Four undergraduate students and one faculty member spent seven days working through the original TinyOS tutorial [2]. We describe our experiences, lessons and code alterations.

2 Starting Point

2.1 Hardware Specifications

Sensor network devices (motes) are manufactured by companies such as Intel, Crossbow, Dust Networks, Millennial Net, Arched Rock, Ember, and others. We had access to MicaZ motes, including sensor boards and

programming boards. The MicaZ specifications are as follows [4]:

- IEEE 802.15.4, tiny, wireless measurement system
- designed specifically for deeply embedded sensor networks
- 250 kbps, high data rate radio
- wireless communications with every node as router capability
- expansion connector for light, temperature, RH, barometric pressure, acceleration/seismic, acoustic, magnetic, and other Crossbow sensor boards



Figure 1. MicaZ Mote

MIB510 Programming board specifications:

- base station for wireless sensor networks via standard MICA2 processor radio board
- serial port programming for all MICA hardware platforms

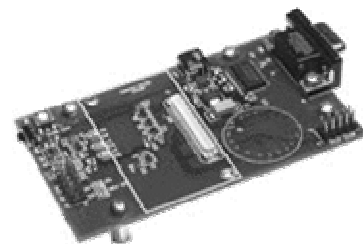


Figure 2. MIB510 Programming Board

2.2 TinyOS

TinyOS is an open-source operating system designed for embedded systems with very limited resources, like the Mica series of motes. TinyOS uses the NesC language, an

extension to C, with similar syntax, that attempts to embody the structuring concepts and execution model [5]. As an embedded operating system, it responds to hardware events with handlers, while also allowing tasks, which are equivalent to functions in other programming languages. TinyOS does not implement object sharing.

3 Hands-on Exercises

3.1 Lab Exercise I

The goal of our first hands-on exercise, designed for the undergraduate classroom, is to write one simple program and to upload the program to one mote. This exercise is derived from lesson 1 in the TinyOS tutorial. The steps are as follows:

- 1) Install TinyOS to a PC
- 2) Connect programming board to power and serial port of PC
- 3) Attach mote onto programming board
- 4) Compile and upload the *Blink* code via provided instructions

Several attempts were necessary to properly install TinyOS. Issues arose due to different combinations of Java versions and PC operating systems. We succeeded with the Windows version 1.1.11 InstallShield. This installation includes a Java environment, Cygwin, TinyOS, and platform support for MicaZ, Mica2 and others motes.

NOTE - For MicaZ motes, version 1.1.7 (or above) of TinyOS is needed.

NOTE – DO NOT place motes onto a powered serial board with the batteries in the mote. This can cause them to malfunction, and potentially no longer work.

Since our programming board had a serial connector (other models have a parallel connector), when uploading the program from the PC to the mote, we needed an additional parameter at the end of the make command, specifying <board type>, <device>. The correct command for Cygwin on Windows was:

```
$ make micaz install mib510, /dev/ttyS0
```

This will build the Blink application and install it on the MicaZ mote using the serial port as an upload channel.

The most important part of the *Blink* code, written in NesC, is:

```
implementation {
  command result_t StdControl.init() {
    call Leds.init();
    return SUCCESS;
  }

  command result_t StdControl.start() {
    return call Timer.start(
```

```
TIMER_REPEAT, 1000);
  }

  command result_t StdControl.stop() {
    return call Timer.stop();
  }

  event result_t Timer.fired()
  {
    call Leds.redToggle();
    return SUCCESS;
  }
}
```

A module that uses an interface (e.g. `Timer.start()` in the above code) must implement the events that this interface uses (e.g. `Timer.fired()` in the above code).

Our code alterations included modifications (1) to the timer's rate in the call to `Timer.start()` and (2) to the code in the `Timer.fired()` event handler. Since the MicaZ has three LEDs, displaying the lower three bits of a counter was a nice assignment for the students.

3.2 Lab Exercise II

In our second hands-on exercise, several motes use radio communication. This exercise is based on lesson 4 from the TinyOS tutorial. The steps are as follows:

- 1) Connect programming board to power and serial port of PC
- 2) Attach first mote onto programming board
- 3) Compile and upload the *CntToLedsAndRfm* code via provided instructions on one mote
- 4) Switch to other mote
- 5) Compile and upload the *RfmToLeds* code to the second mote



CntToRfm



RfmToLeds

NOTE - Different motes use different frequencies (see sticker on mote's chip). The frequency has to be specified in Hertz – not Megahertz - in the PFLAGS arguments. See the Hardware verification page of the TinyOS tutorial for more help. MicaZs do not require a compile-time argument to use the 2.4 GHz range.

When the motes are turned on one mote will have a counter and send it over the radio. The other mote will simply display the lower three bits of the integer using its three LEDs.

One modification to the *RfmToLeds* code is to also send the received number out on the radio, effectively creating a “repeater” of the original signal. This will increase the range of broadcast from the counting mote. The following code shows this modification in bold, the rest of the code is identical to that found in *RfmToLeds*:

```
configuration RfmToLedsAndRfm {
}
implementation {
  components Main, RfmToInt, IntToLeds,
  IntToRfm;

  Main.StdControl->
    IntToLeds.StdControl;
  Main.StdControl->
    RfmToInt.StdControl;
  RfmToInt.IntOutput->
    IntToLeds.IntOutput;
RfmToInt.IntOutput->
IntToRfm.IntOutput;
}
```

Problems arose when two repeaters were within range of each other, as we create a self-feeding loop! For example, the first number was sent from the counting mote to repeater1, to repeater2, to repeater1, to repeater2, ad infinitum.

This loop can be avoided if repeater1 does not re-repeat from repeater2, either using functionality from the TinyOS *Ident* program, or with a unique message ID and dropping duplicates.

3.3 Lab Exercise III

In our third hands-on exercise, the PC displays sensor readings. This exercise is based on lessons 2 and 6 from the TinyOS tutorial. The steps are as follows:

- 1) Connect programming board to power and serial port of PC
- 2) Attach mote and a sensor board onto the programming board (sensors go below)
- 3) Compile and upload the *Oscilloscope* code to the mote
- 4) Run the SerialForwarder java application (following instructions in Lesson 6). Do not terminate the program, it is essential for the next step (new Cygwin window or other terminal window)
- 5) Run the Oscilloscope Java application to see readings appear in a pop-up window.
- 6) Keep the mote attached to the board (change the light intensity and observe the graph).

Problems encountered: the SerialForwarder is very specific with regards to its serial port arguments, especially the baud rate. For MicaZ (and Mica2), we ran it at a 56700 baud rate, any lower will result in no readings and many “write failed” packet errors.

For getting readings from a remote mote, load one mote with the *OscilloscopeRF* code, and another with the *TOSBase* code. This sends your sensor readings to the base and then to the SerialForwarder. The TOSBase mote must remain connected to the programming board during measurements.

NOTE - You cannot build and install any application for TinyOS with the SerialForwarder running, you must quit the serial forwarder and then run the make install command. Also, sometimes the process does not terminate properly so if you get some errors with the SerialForwarder, run the “ps -a” command and use the “kill -9 <pid>” command to stop unwanted programs from running.

4 Discussion and Conclusion

It is critical that the next generation of computer scientists know how to employ and program sensor networks. MicaZ motes (running TinyOS) are easily available [4]. Whereas the TinyOS tutorials [2] seem to be targeted at graduate students and professional researchers, in this paper, we describe three exercises that we deem feasible in the undergraduate classroom. We describe our experiences, lessons and simple code alterations.

Similar to Lego Mindstorm, sensor networks can be a compelling experience for students, possibly attracting more than just your typical computer savvy student.

Acknowledgements

This work is supported by W. M. Keck Foundation, by the John S. Rogers Program, and by the National Science Foundation under grant DUE 0411237.

References

- [1] MIT Technology Review, 10 Emerging Technologies That Will Change the World, Feb. 2003, http://www.technologyreview.com/read_article.aspx?id=13060&ch=infotech
- [2] TinyOS tutorials, <http://www.tinyos.net/tinyos-1.x/doc/tutorial/>
- [3] Wired Magazine, “Intel's Tiny Hope for the Future”, http://www.wired.com/wired/archive/11.12/intel.html?tw=wn_tophead_5
- [4] Crossbow, Inc., <http://www.xbow.com/>
- [5] Introduction to NesC, <http://nesc.sourceforge.net/>