2006-1547: IMPLEMENTING SENSOR NETWORKS USING SENSOR MOTES AND J-DSP

VISAR BERISHA, Arizona State University

VISAR Ho-Min Doctoral student under an NSF Fellowship working in speech processing and in real-time sensor fusion.

HO-MIN KWON, Arizona State University

Ho-Min is a Doctoral student working on beamforming and on real-time sensor networks.

Andreas Spanias, Arizona State University

Dr. Andreas Spanias is professor working in the area of signal processing in the Department of Electrical Engineering.

Interfacing Java DSP with Sensor Motes

Abstract

Distributed wireless sensor networks (WSN) are being proposed for various applications including defense, security, smart stages, and other. The introduction of hardware wireless sensors in a signal processing education setting can serve as a paradigm for data acquisition, collaborative signal processing or simply as a platform for obtaining, processing, and analyzing real-life real-time data. In this paper, we present a software interface that enables the Java-DSP (J-DSP) visual programming environment to communicate in a two-way manner with a wireless sensor network. This interface was developed by writing nesC code that enables J-DSP to issue commands to multiple wireless sensor motes, activate specific transducers, and analyze data using any of the existing J-DSP signal processing functions in real time. A series of exercises were developed to provide hardware experiences to signals and systems and DSP undergraduate students. The interface, the exercises, and some preliminary assessment results are discussed in the paper.

1. Introduction

The application area of wireless sensor networks poses a series of important research problems in signal processing, communication networks, power-aware implementations, and remote sensing[1,2]. Wireless sensors have been applied to a diverse series of applications including ecological and environmental monitoring, sound and sniper localization, multiple target tracking, smart stages, and biofeedback [3-7]. Theoretical aspects of sensor networks include power versus bandwidth tradeoffs, clever parameterization schemes, robust information extraction, and collaborative sensor configurations. In addition to theoretical issues, implementation aspects present equally challenging problems owing to the lack of user-friendly software tools, poorly documented interfaces, "leaky" buffers, drifting sampling periods, etc. Although many of these problems will be eventually solved and perhaps some solutions are already available in classified literature, working with sensor motes is very useful from the education point of view. Working with inexpensive wireless sensor motes can be valuable in terms of providing undergraduate experiences with real-time heterogeneous transducer data. Simple paradigms of signal conversion, data collection and fusion, real-time filtering, and spectral estimation can go a long way in explaining both theory and applications of signal processing. The main difficulty, even with simple implementations of the above, is again the absence of user-friendly software tools. In this paper, we describe a Java/nesC graphical interface that we developed to provide capabilities for two way communications between a PC platform and a network of wireless sensor motes. This interface is integrated in our J-DSP[8] visual programming environment and enables J-DSP to issue selective commands to

multiple sensors on different wireless platforms, acquire data from a single or a group of sensors, etc. This capability enables students to use all of the existing signal processing functions in J-DSP and form and execute real-time/real-life simulations using the user-friendly environment of J-DSP.

Educational simulations include, obtaining data, characterizing the frequency spectrum using real-time FFTs, performing time-invariant or adaptive filtering, using simple non-linear functions (thresholds) to detect events at specific locations, etc. In fact virtually all non real-time online laboratories that we previously developed[8] can be adapted for use with real-time sensor data. Not only can undergraduate experiments now be performed real time, but we can also use these platforms to provide exposure to exciting new applications. For example, simple data fusion simulations can be formed by having multiple sensors acquire and parameterize distinct sub-bands of an audio signal and wirelessly transmit these sub-band parameters to a base station where data is fused and processed to recreate the signal. Studies of transmission errors, sampling drifts, the sharing of signal processing functions, and collaborative approaches are a possibility with this graphical interface; some of these have already been worked out and presented to students in the DSP class. TCP-IP control of the motes is seamless with this Java interface and remote sensing paradigms are a possibility and an example is presented later in this paper. The rest of the paper is organized as follows. A hardware overview is presented in Section 2 and the native software platform for the motes is described in Section 3. Section 4 presents the Java interface and in the same section we describe J-DSP dialogue panels that are used to issue commands to the wireless motes and acquire data from the sensors. Section 5 describes hardware experiments with the sensor motes assisted by J-DSP and provides preliminary assessment. The last section presents our concluding remarks.

2. Hardware and Software Overview

A wireless sensor network consists of nodes that have several capabilities. Each node must have low power consumption, programmable scheduling, independent operation of transducers, and some basic signal and data processing capabilities. Researchers at several universities and industrial settings developed circuits - with the efforts at Berkeley and Rice being perhaps more known in the signal processing circles. In our work, we used the Crossbow™ motes which we describe briefly below.

2.1 The Mote

The commercial version of the wireless sensor hardware is called a mote[9-13]. The sensor mode is comprised of two parts, the MICA2™ microprocessor board (Fig. 1) and the sensor daughter board (Fig. 2). The combination of the two boards can be used for data acquisition from transducers, data forwarding, or information processing. The MICA2 board has an RF transceiver, 512KB external flash

memory, an Atmel ATMega128 8-bit microprocessor operating at 7.3728 MHz clock. The ATMega128 contains 128KB internal memory and 4KB SRAM and supports a serial port and and an I^2C (Inter IC) bus; it has an interface to an 8-channel, 10-bit analog-to-digital converter. The RF transceiver supports FSK modulation and a 38.4 kBaud rate.

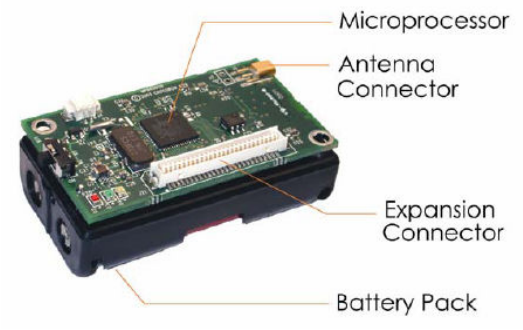

Figure 1. The MICA2 Platform (by permission from Crossbow).

The sensor board MTS310CA contains a set of five sensors, i.e., a temperature sensor, a photo sensor, a microphone, an accelerometer, and a magnetometer. A buzzer and three LEDs are also part of the mote.

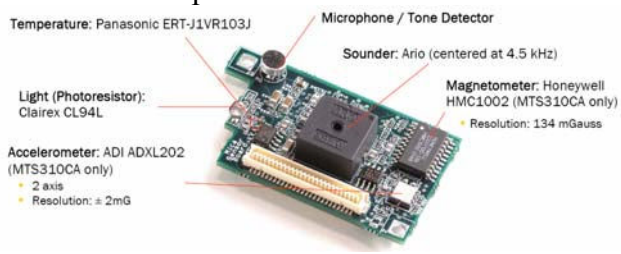

Figure 2 MTS310CA Sensor Board (by permission from Crossbow).

2.2 Software Aspects of the Mote

The TinyOS ™, is the operating system used for the distributed wireless sensor network. The TinyOS libraries, and applications are written in nesC, which is a programming language for the TinyOS. A TinyOS component hierarchy for sensing applications was established. To manage the demand of interrupts and sensing data requests, TinyOS provides an event-driven concurrency model. TinyOS schedules tasks sequentially corresponding to events caused by interrupts and all tasks are executed in a FIFO mode. The nesC is a dialect of the C language designed for component-based applications. The grammar of nesC is an extension the ANSI C grammar[10].

3. Interface with JAVA

3.1 Programming the Board (MIB510).

The programming board is connected to the COM port in the PC with a data rate of 57600bps. USB to RS232 adaptors may be used for USB connectivity. A user

can download an application onto MICA2 processor board by first communicating with the base station via the RS232 which then sends the information to the appropriate mote through the RF connection.

3.2 Serial Programming and Java.

A packet switched communication scheme is implemented between Java-DSP and each MOTE. The general structure of each data packet is shown in TABLE I. The first and last portion of the data packet contains the synchronization byte, 0x7E. This is used to detect the start and end of a packet from the data stream. In addition, each packet has an identifier that determines its type (as shown in TABLE I). Following the identifier is the active message, the CRC byte, and the termination byte[13,14].

BLE I
TinyOS Data Packet

Index	Field	Description
0	Sync.	0x7E
1	Packet Type	There are five known packet types: • P_PACKET_NO_ACK(0x42): User packet with no ACK required • P_PACKET_ACK(0x41): User packet with ACK required. It includes a prefix byte. Receiver must send a P_ACK response with prefix byte as contents • P_ACK(0x40): The ACK response to a P_PACKET_ACK packet. It includes the prefix byte as its content. • P_UNKNOWN(0xFF): An unknown packet type
2 - 33	Active Message	• Destination Address(2 bytes) • Active Message Handler ID(1 byte) • Group ID(1 byte) • Unknown (2 bytes): 0x5D & 0x1A • Node ID(2 bytes) • Message Length(2 bytes) • Channel ID(2 bytes) • Data (20 bytes)
34-35	CRC	(2 bytes)
36	Sync.	0x7E

3.3 Creating a GUI for the mote in J-DSP.

The GUI of the mote has been developed and integrated in J-DSP. J-DSP acts as an additional layer at the base station through a physically wired serial port. Java requires a signed applet to permit resource access[15]. Additionally, Java needs the Communication API to build the RS232 serial interface. A user must copy three different files, win32com.dll, comm.jar, and javax.comm.properties, to the specific directories described in Communication API manual. The Mote block in

J-DSP allows users to individually control the available motes and base station settings. The control panel has four settings, RS232 settings, the mote settings, output buffering, and the plot area. A user can seamlessly perform a set of sensing operations (Light, Temperature and MIC sensor) and activate a set of outputs (LED and Buzzer). The real-time graph plots data as it comes in and the output buffer allows manipulation of data in individual frames (8 to 256 samples each). A frame-by-frame interface was written to facilitate the application of signal processing tools for non-stationary signals. The size of the window and degree of overlap can be defined by the user. Just about all the J-DSP signal processing functions, such as the FFT, the filter, the AR spectral estimation, the correlation, and the periodogram/correlogram, etc, can be interfaced with any of the wireless motes. Commands to any of the sensor motes can be issued from the J-DSP mote dialog (Figure 3), and data can be acquired from any sensor on any mote individually or even from groups of sensors simultaneously.

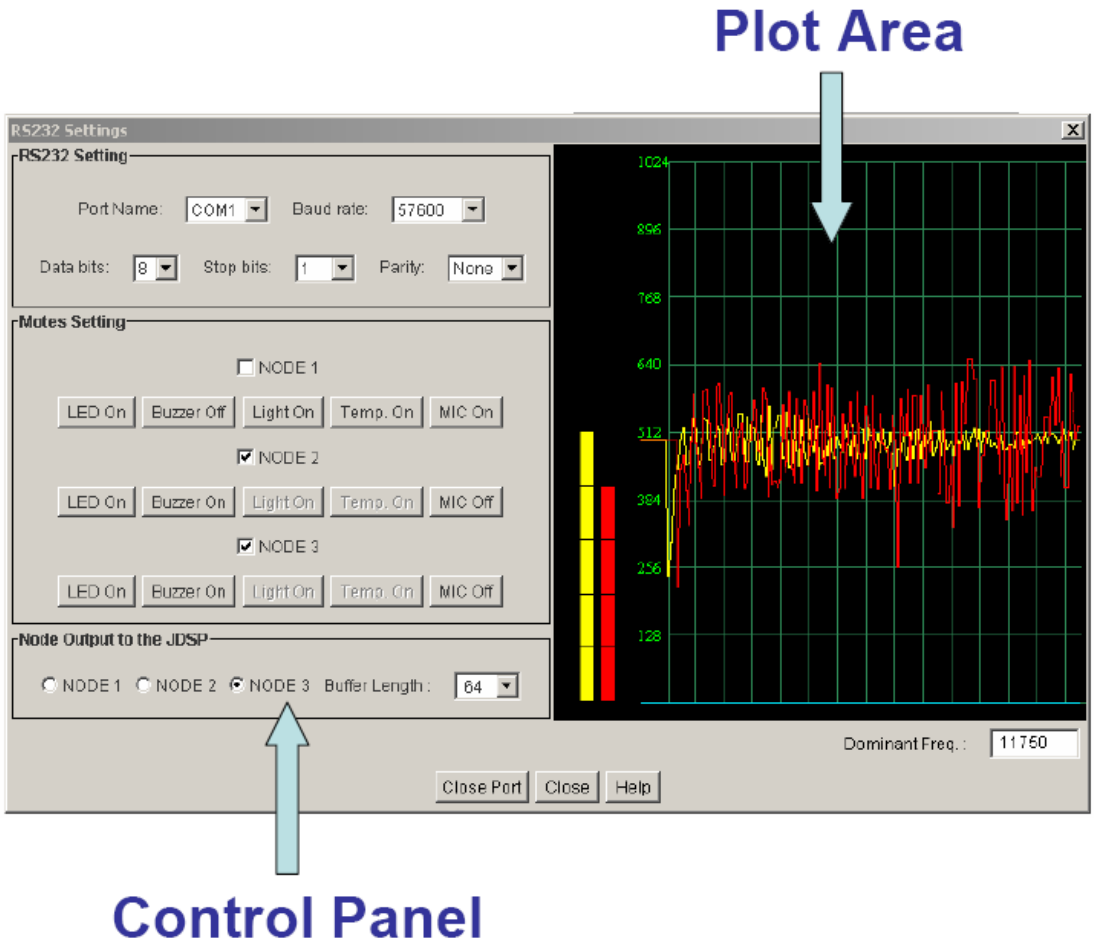

Figure 3. The Mote Block in J-DSP.

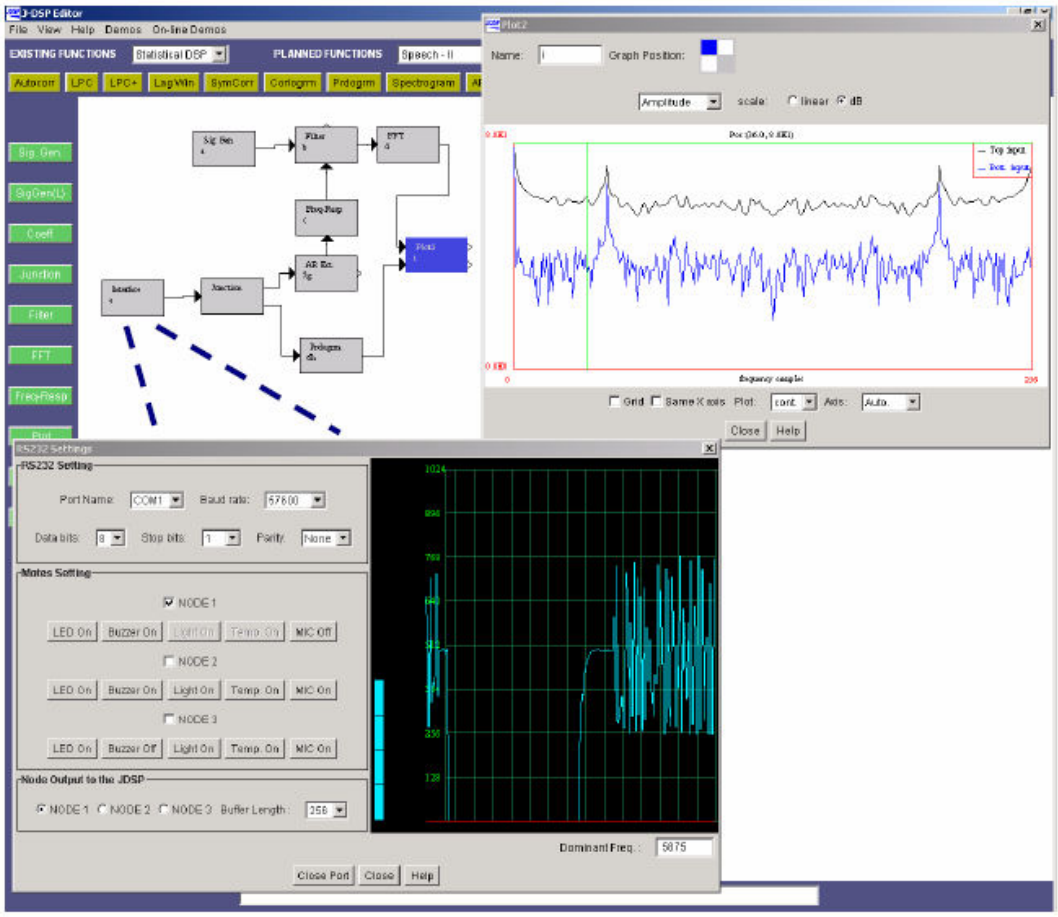

Figure 4. A simulation of an auto-regressive (AR) spectral estimator operating on a microphone sensor signal.

4. Exercises with Sensor Motes

The presence of the J-DSP interface enables students to perform real-time signal processing experiments with signals acquired from the sensor motes. Filter design exercises have been developed where frequency responses are designed to select tones generated by two motes and acquired by a third sensor mote with its microphone activated. Low pass, high pass, and notch filters are designed and programmed in J-DSP to filter the real time signal captured by the third mote. The sampling frequency of the motes is set at 8 kHz. The frequencies of the tones are measured using the FFT and the problems of spectral leakage and resolution are examined with real time data. Students are asked to observe the overall spectra with the FFT and with autoregressive spectral estimators (Figures 4 and 5).

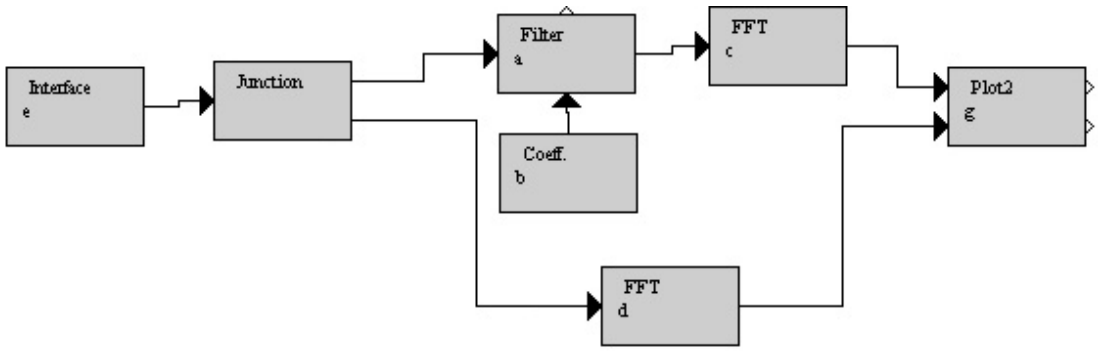

Figure 5. J-DSP real-time filtering block diagram.

In addition to the signal processing functions available in J-DSP, a simple remote sensing demonstration was also developed. An animated graphical panel, with the four motes placed in four corners of a 200 square feet room, was also programmed and integrated in J-DSP as shown in Figure 6. Every mote has four sensors that are activated from the J-DSP dialog panel and in turn pass continuous data to J-DSP. Data is obtained from all accelerometers, microphones, light sensors, and temperature sensors. At the base station data is fused and the location, temperature, sound and light intensity, and accelerometer reading are collected. An algorithm is used to locate a walking person within the room by using all the available sensor data.

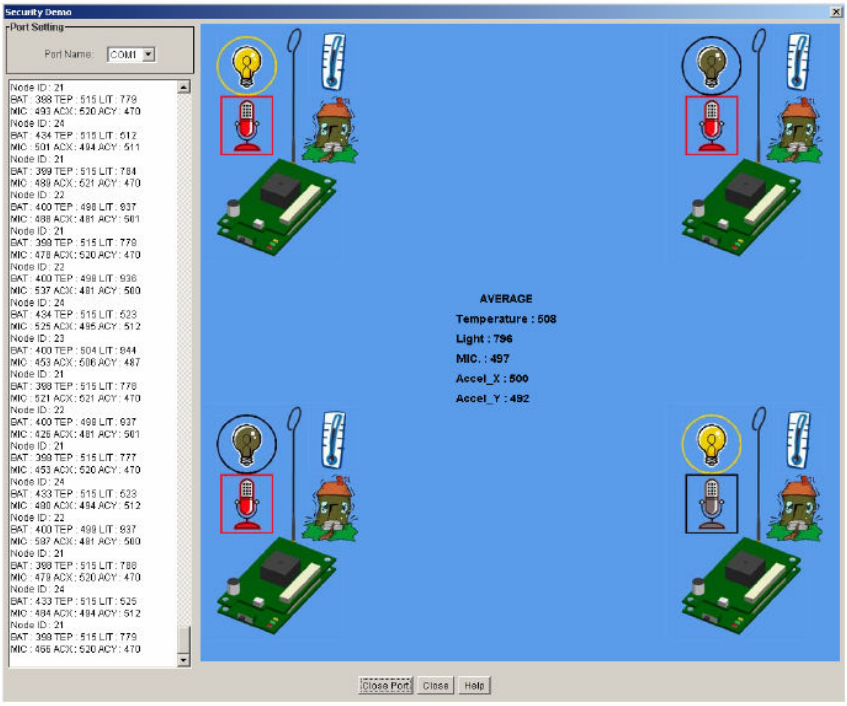

Figure 6. Demo of Remote Sensing and Localization.

One of the important aspects of this localization experiment is that the sensors can also be placed at a remote location and communicate with J-DSP through a TCP-IP connection - thereby demonstrating truly remote sensing that uses the internet infrastructure.

5. Assessment Results and Concluding Remarks

A group of students of varying technical expertise was selected to perform some of the experiments described in the paper. Pre/post-lab evaluation questions are posed to the students before and after completion of the experiments in order to evaluate the success of using the motes in an educational setting. The questions are technical and they are posed in order to evaluate the understanding of important mote concepts. We describe here some of the preliminary results obtained in this study. The evaluation consisted of 12 multiple choice (MC) and true/false (TF) questions. Multiple choice questions were of the form "What are the characteristics of individual sensors in wireless sensor networks". Preliminary results show an increase in the average score on the evaluation questions. The average score in the pre-test was 52.4% with a standard deviation of 14.9, however this increased to 71.4% with a standard deviation of 12.6 in the post test. In addition, in TABLE II, we show statistics from each of the questions individually. For most of the questions, the percent of correct answers significantly increases. More information, screenshots and AVI files associated with this study can be found on http://jdsp.asu.edu.

TABLE II
PRE/POST STATISTICS [THE PERCENT OF STUDENTS THAT OBTAINED THE CORRECT ANSWER] FOR EACH QUESTION OF THE MOTE QUIZ

Evaluation question	Pre (%)	Post (%)
What are sensor networks used for? (MC)	28.6	28.6
The processor on a sensor mote is faster than the PC processor. (TF)	57.1	85.7
The processor on a sensor mote is capable of high precision arithmetic. (TF)	42.9	100
The processor speed on a sensor mote is (circle one) (MC)	42.9	71.5
The information communicated by the mote to the base station is (circle one) (MC)	85.7	71.5
The sensors are (circle one) (MC)	14.3	71.5
The benefits of a sensor network are primarily (MC)	42.9	100
Sensor motes are optimized for (MC)	28.6	14.3
Sensing acoustic signals and temperatures require the same sampling rate. (TF)	71.4	85.7
Near real-time processing with the motes and J-DSP is possible. (TF)	100	100
Measuring the spectrum of a signal from an acoustic sensor using the FFT in J-DSP enables users to estimate the pitch period of a voice signal. (TF)	100	100
In the mote platform what is the most power consuming component? (MC)	0	28.6

† The assessment results are preliminary.

References:

[1] I.F. Akyildiz, et al, "Wireless Sensor Networks: a Survey," *Computer Networks*, Vol. 38, pp. 393-422, 2002.
[2] D. Culler, D. Estrin, M. Srivastava, "Overview of Sensor Networks," *IEEE Comp. Sos. Mag*, pp. 41-49, August 2004.
[3] R. Want, et al, "The Active Badge Location System," ACM Trans. on Info. Syst., Vol. 40, No. 1, pp. 91-102, January 1992
[4] N.B. Priyantha, A. Chakraborty, H. Balakrishnan, "The Cricket Location-Support System, 6th ACM Int. Conference on Mobile Computing and Networking, August 2000.
[5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in Proc. of 1st ACM workshop on Wireless sensor networks and applications, pp. 88-97, ACM Press, September 2002
[6] D. Li, K. Wong, Y.H. Hu, and A. Sayeed, "Detection, Classification and Tracking of Targets in Distributed Sensor Networks," IEEE Signal Proc. Mag., Vol. 19, March 2002.
[7] M. Maroti et al, "Shooter Localization in Urban Terrain," IEEE Computer, Vol. 37, pp. 60-61, August 2004.
[8] A. Spanias, V. Atti, A. Papandreou-Suppappola, K. Ahmed, M. Zaman, and T. Thrasyvoulou, "On-line Signal Processing using J-DSP," IEEE Signal Processing Letters, vol. 11, no. 10, pp. 821-825, Oct. 2004
[9] Crossbow, "TinyOS Getting Started Guide", 2004
[10] Crossbow, "MTSMDS sensor and data acquisition boards user's manual", 2004
[11] D. Gay et al, "The NesC Language: A Holistic Approach to Networked Embedded Systems," Intel Research, IRB-2002-019, Nov. 15, 2002
[12] Berkeley TinyOS project http://webs.cs.berkeley.edu/tos/
[13] Crossbow, "TinyOS Tutorial", 2004
[14] J. Thorn, "Deciphering TinyOS Serial Packets," Octave Technology, Octave Tech Brief #5-01, March 2005
[15] Java Technology http://java.sun.com

Acknowledgment: This work has been sponsored in part by the ASU SenSIP cluster and by NSF DUE 0443137.