Using a Multi-functional Sensor Network Platform for Large-Scale Applications to Ground, Air, and Water Tasks

Elizabeth Basha, Carrick Detweiler, Marek Doniec, Iuliu Vasilescu, and Daniela Rus
Distributed Robotics Laboratory
Computer Science and Artificial Intelligence Laboratory
{e_basha,carrick,doniec,iuliuv,rus}@mit.edu

Abstract

We present a modular sensor network platform capable of supporting a wide range of applications. We developed a platform to support a broad spectrum of scenarios, instantiating our system for applications on the ground, in the water, and in the air. Our system has operated in the field for over 240 days with month long continuous deployments, measuring positions, temperatures, pressures, and rainfall, while computing cattle behaviors, event locations, and future river level. We use this experimental experience to discuss the lessons learned in designing and using a modular and multi-functional system.

1 Introduction

We wish to develop a multi-functional sensing platform to enable a large and heterogeneous range of applications in air, on ground, and in water. Our sensing platform satisfies our design requirements for a flexible and modular sensor node including:

- Easy addition and use of many sensor types
- Easy addition and use of many communication types
- Easy reconfiguration and programming of the system, both within a specific application and for switching to a different application

We instantiated this sensor network system in three different application areas: (1) virtual fencing for cattle herds, (2) underwater monitoring of coral reefs, and (3) river flood prediction. Table 1 demonstrates the variability of these applications on our system. Each has different environmental needs (ranging from mobile to underwater to covering large geographic areas) and different usage models (ranging from regular, fast operation patterns to variable, very slow patterns).

Application	**Virtual Fencing**	**Coral Reefs**	**River Floods**
Mobile or Fixed	Mobile	Fixed	Fixed
Coverage Area	1 km	10s km	100s km
Operational Life	Months	Months	Years
Event Time Scales	Sec	10s Sec	Min
Number of Events	6-12	10-12	12-15
Scheduled Events	Interval	Interval	Time
Sensor Operation	Polled	Polled	On Access
Number of Log Files	4-6	4-6	8-10
Ease of Sys Access	Easy	Difficult	Moderate

Table 1. Comparison of Applications on System

In successfully implementing these applications, we developed a platform with over 240 days of experimental operation, typical active power usage of 155 mW, support for 6 sensor types and 4 communication types, and solving 3 different algorithmic problems. The modularity of our system enables 80 to 90% sharing of code between these applications, speeding development and aiding debugging. We also learned useful lessons about designing multi-functional systems, UART multiplexing, communication abstractions, and power management design among other lessons.

This paper is organized as follows. Section 2 discusses related systems. Section 3 describes the system architecture and design. Section 4 reports our experimental results characterizing the platform in terms of communication, power, operational behavior, and deployments. Section 5 describes lessons we learned through the design and implementation of our sensor platform.

2 Related Work

We build on several years of important work in designing and fielding sensor network systems on a variety of platforms including the Mote [1, 7, 8], Fleck [14], and Cricket [10]. Our own experience with these systems has led us to the design decisions described in this paper.

Many of these systems have been deployed for a variety of applications: measuring light intensity under foliage (LUSTER [12]), performing detailed studies of a Redwood tree [13], monitoring cattle (Flecks [14]), measuring high-altitude environmental data (PermaSense [3]), and monitoring fatigue in bridges [5]. While not for a specific application, some platforms provide large scale deployment for basic networking research such as the Trio Testbed which had nearly 600 Motes [4].

While all of these platforms provide good options for sensor network research, none supports a full range of communication and sensing options while also supporting complicated algorithms. As these are vital requirements for our projects, we outline in this paper the design of a new platform as well as our operating system, which takes a similar approach to TinyOS [6].

3 System Architecture and Design

The specific requirements for a modular sensor network system include:

- Low-level support for all reasonable sensor types (resistive, interrupt, serial, etc.) along with easy high-level addition of new sensors using these types
- Wireless communication support for all projects and easy addition of new communication devices
- High-level reconfiguration of the system and the ability to do so in the field without a direct cable connection to a device
- Long-term (over a year) storage of data

In response to these requirements, we have designed a hardware platform, supporting operating system, and software infrastructure for applications. Figure 1 and 2 show the block diagram of the hardware architecture and operating system, respectively. In defining our system, we focused on the following areas: processing, communication, sensing, power management, data storage, and configuration.

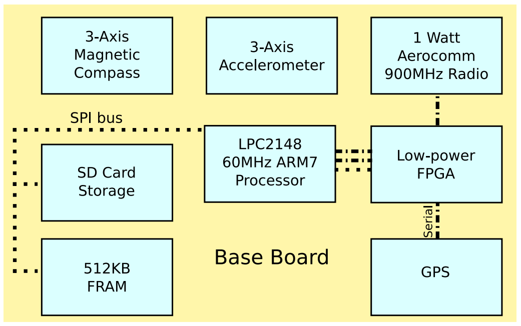

Figure 1. Hardware architecture block diagram.

3.1 Processing

The key hardware requirement is a processor with a relatively large amount of on-chip RAM (40K), flash (512K), input-output pins, and other features. We selected the LPC2148 ARM7 processor [9] which satisfies these requirements and balances the trade-off between power usage and processing. Others we considered tended towards the extremes of this spectrum with the ATmega128 on the low end and the Blackfin ADSP-BF533 on the high end. The low end systems do not provide sufficient internal memory to enable a large range of functionality; using the high end systems would allow us to run more complex operating systems such as Linux, but at the cost of higher energy usage and reduced system lifetime. Our system does support using either of these processor types in addition to the main processor where the application requires it. However, the LPC processor provides a reasonable mix of functionality for most applications.

Our base software supports all the various operations necessary for a variety of projects: measurements, communication, failure checking, logging, algorithms, and other activities. To process these operations, we designed a non-preemptive multi-tasking scheduler-based system utilizing the real time clock and millisecond timers.

3.2 Communication

Seamless transitions between communication devices require sufficient protocols and their hardware support. Here, the LPC2148 provides UARTs, SPI, I2C, and USB slave. To avoid the 2 UART limit and to successfully manage a variety of UART devices, we include a low-power FPGA which bidirectionally buffers up to four additional serial ports allowing simultaneous communication with up to 6 serial devices. Also, we add a 900 MHz Aerocomm AC4790 radio to all boards, chosen due to its claim of a 32 kilometer communication range and initial testing showing reasonable ranges on small, local scales.

We need several abstraction layers to avoid exposing communication switching complexities to the end user but still enable easy addition of other devices. We start with four low-levels interfacing with the microprocessor: the FPGA access code, the UARTs, the SPI interface, and the I2C interface. Just above this layer, we provide an AC4790 layer to interface with the Aerocomm built-in communication protocols.

On top of these low-level accesses, we provide a communication system that further abstracts what communication device and what message. Any module can create a message that is added to a message queue that the scheduler processes. A module receiving a message will only handle those it recognizes and for which it has a handler capable of processing that type of message. This enables different projects to react differently to the same message as well as to create project specific messages that do not interfere with another project's modules.

3.3 Sensing

Our system supplies several base sensors: temperature, compass, accelerometers, and GPS. These base sensors help provide

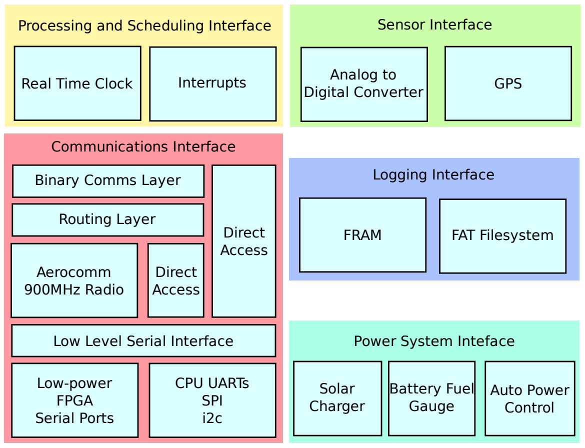

Figure 2. Operating system block diagram.

system information regarding location, position, and internal temperature, useful data for almost every project. Most projects require additional sensors of a variety of connection types. We support this by exposing as many input-output (IO) pins as possible and enabling connections to expansion boards. Our expansion board connection provides IO pins, UART ports, ADC pins, I2C bus access, and SPI bus access.

We provide ease-of access and addition through several software layers. We start with the basic hardware, developing ADC code and GPS access code in addition to existing I2C and UART layers. On top of this, the sensor layer abstracts out the different access types each sensor uses. Adding a sensor requires only placing it in the list of available sensors within the sensing layer; it is then available and supported through the existing sensor functions to all higher layers.

3.4 Power Management

To measure and control the power, we have a charge circuit allowing solar and DC charging of single cell lithium-polymer (LiPo) batteries. This circuit is based on the LTC1733 which also provides measurement of the charge current so we know the amount of power entering the system. We use single cell LiPo batteries to eliminate the need for switching regulators; supporting a wider range of batteries requires the addition of a switching regulator and different charging circuitry. For understanding the amount exiting the system, we add a battery circuit (ISL6295) to each lithium-polymer battery. Within the circuit, we measure current (both charge and discharge), voltage, and temperature.

3.5 Data Storage

The LPC2148 provides 40K of on-chip storage allowing buffering of serial devices and log files, while leaving sufficient space for user algorithms. Beyond this, we add a 32K FRAM and a mini-SD card slot. We developed code to enable fast and easy access of the FRAM and support streaming data to/from it. The SD card requires a file system so we implemented a FAT file system to ensure readability of the SD cards on regular computer systems. On top of the FAT system, we added a logging system that enables creation and writing of many concurrent log files.

3.6 Configuration

We designed a bootloader program to load new program code into the system. The bootloader reads the program file from the SD card allowing very easy reconfiguration. We update the program by swapping SD cards or by uploading a file via a serial or radio link. Once the system starts operation, the FRAM provides fine-grained configuration through variables permanently stored in it. Between these two configuration options, we can easily switch any node to a different operation within a project, a different code version, or a different project.

Device	Physical Layer Rate	Real Rate	Maximum Range	Typical Range	Success Rate (at Typical)
900 MHz Radio	57600b/s	7200b/s	3km	100m	25-50%
144 MHz Radio	1200b/s	818b/s	60km	50km	90%
Bluetooth	1Mbit/s	92100b/s	50m	5m	100%
Serial Cable	115200b/s	92100b/s	300m	1m	100%
Optical Modem	1Mbit/s	800Kbit/s	4m	3m	90%
Acoustic Modem	300b/s	22b/s	500m	400m	56%

Table 2. Communication results based on field deployments.

Component	Current (mA)
Base Board Sleep Mode	2
CPU Low Usage	16
CPU Max Usage	59
FPGA	<1
Base Sensors	6
GPS no fix	44
GPS fix	35–40
900 MHz Radio Receive Only	20
900 MHz Radio Transmit 1 Hz	73
Cow Extension Board Standby	7
Cow Extension Board On	29
Cow Shocking	200–400
Underwater Extension Standby	<1
Underwater Extension Active	15
Acoustic Receive	110
Acoustic Transmit	200
Optical Receive	15
Optical Transmit	100–500
River Extension Board	<1
144 MHz Radio Transmit	5000

Table 3. Subsystem power usage.

4 Experiments and Results

We have instantiated the sensor network system and deployed it in the field in the context of three applications: monitoring and controlling cattle, monitoring coral reefs, and monitoring and predicting river floods. These deployments were used to characterize the performance of our sensor network system and also to evaluate the application solutions.

4.1 Communications

We logged data regarding the number of bytes and the type of messages sent. These data describe the behaviors of the different communication methods, highlighting the need for a variety of different options and the trade-offs between then. Table 2 summarizes our results, showing the data rates, ranges, and success percentages for the various communication methods used. Also shown is the wide spectrum of communication ranges at which our system can operate, from 1 meter to 60 kilometers. Maximum range defines the maximum distance at which we have seen the device function, not the manufacturer specified range (Aerocomm states 32 kilometers, but we have only achieved 3). Our final column of the table demonstrates the communication success rates, based on our experiments. The lowest rates reflect the difficulty of communication medium in the case of the acoustic modems and limitations of the Aerocomm module in the case of the 900 MHz radios.

4.2 Power Usage and Charging

We characterized the power usage of our system through our field experiments and directed lab testing to understand what uses the most power and what trade-offs we incur through our design choices. Table 3 outlines the power usage of the different system components as well as the components added by each project, summarizing the total current requirement of each. Some systems have additional operational requirements affecting their energy such as the GPS which needs approximately 1 minute of operation to obtain a location fix or the radio which needs to remain in receive mode without additional protocols. Application needs including shock-

Location	Application	Average Charge Current (mA)
Massachusetts	River Flooding	11.28
Honduras	River Flooding	92.12
New Mexico	Virtual Fencing	29.67

Table 4. Daily average charge current.

ing, additional sensors, and specialized communication dominate our power usage, reducing issues with the higher power consumption of our base system.

With such high power usage, we need to focus on power management and, therefore, gather charge current and discharge current measurements. Figure 3 shows the average daily charge current for three different geographic locations: Massachusetts, Honduras, and New Mexico. In these experiments, the Massachusetts and Honduras locations used 2 watt solar cells while the New Mexico location used 1.5 watt solar cells. Table 4 summarizes this information, providing the average daily values seen by each system. Collecting this data enables intelligent power management on the system such as significantly reducing operations at night when no solar recharging occurs.

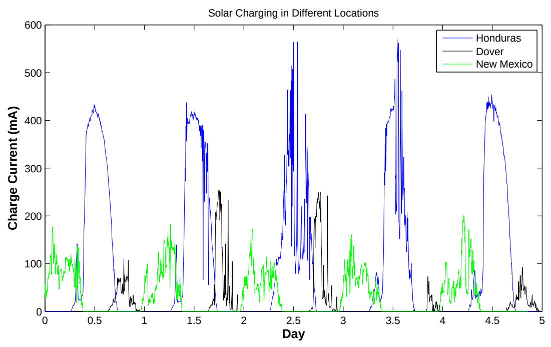

Figure 3. Charge current in Honduras, Dover, Massachusetts, and New Mexico. Days offset for visibility.

4.3 Deployments

The applications stress-tested the system in several ways, utilizing a suite of ground, aerial, and water sensors, and several modalities of communication through air (short range and long range) and water (short range and long range). Although each system has been deployed many times, we focus on the most recent experiments.

Comparison of Applications

Overall all applications share the same base hardware and base software, providing 80 to 90% of the software for any given project. Table 1 outlines key features of our system and how each application uses them. This clearly demonstrates the diversity of applications our system can support.

Virtual Fencing

Virtual fencing aims to help ranchers control their cattle through a mobile sensor network [11]. This network, with nodes attached to the heads of the cattle, monitors the location of each animal and determines if the animal has reached the edge of the "fence," a virtual boundary defined by the rancher. If the animal heads out of bounds, the system provides stimulus in the form of shock and/or sound to direct the animal back within the virtual fence.

On the hardware side, to control the cattle, the system requires an expansion board providing shock and sound capabilities as shown in Figure 5. In this application, we use all the base software, providing 87% of the software, adding complex, application-specific algorithms as described in Figure 4. These algorithms define a virtual fence within which we keep the cattle through directional queues determined by head orientation.

At the time of writing, the virtual fence system has operated for over 5 months in New Mexico at a ranch affiliated with the United States Department of Agriculture (USDA). Here we deployed 5 nodes on cattle, roaming over an area of 5 square kilometers, with a sixth node as a stationary reference and a seventh as a mobile base station.

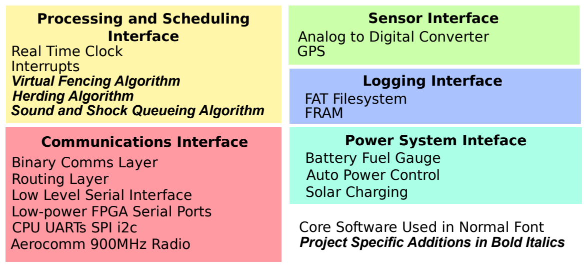

Figure 4. The core software usage and extensions for the virtual fencing project.

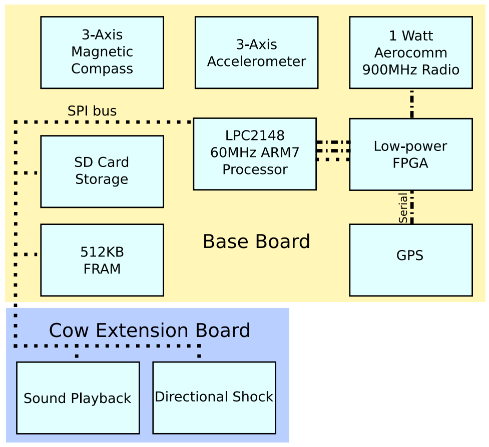

Figure 5. The core hardware usage and extensions for the virtual fencing project.

Coral Reefs: AquaNodes

Coral reef monitoring provides a valuable tool for biologists researching issues related to the flora and fauna existing in the reef habitat. An underwater sensor network provides scientists with an intelligent system, regularly transmitting data to them and informing them of the system status.

The AquaNodes require extending the base hardware and software systems as seen in Figures 6 and 7. On the hardware side, this involves adding an expansion board with a 24-bit AD converter to provide high-precision measurements of external temperature and pressure. The expansion board also has a Atmel ATmega164P low power processor to enable more power management options. Since the radio does not work underwater, we also add acoustic and optical modems. The optical system is used by divers or a robot to download the full logs, while the acoustic modem is used for periodic status updates. To aid deployment, the system has a bluetooth radio and an LCD screen operated through the accelerometer and hall effect sensors, enabling configuration and testing while in the water. The base software provides 83% of the software needed by

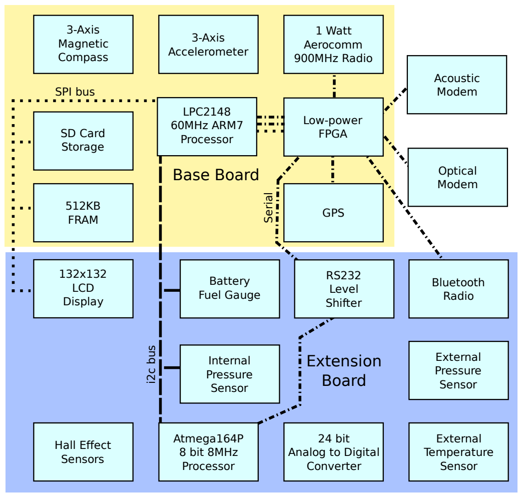

Figure 6. The core hardware usage and extensions for the AquaNodes.

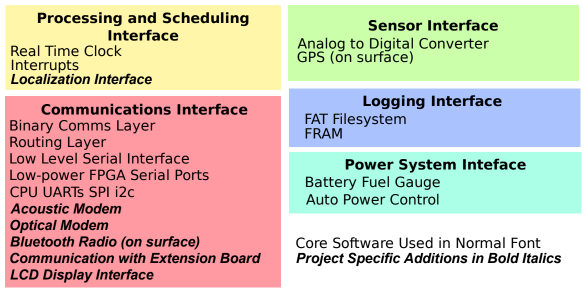

Figure 7. The core software usage and extensions for the AquaNodes.

the application, including the 900 MHz radio and GPS, which help with initial configuration on land.

We have deployed the AquaNodes in rivers, lakes and on the coral reefs at the Gump station in Moorea. During our last ocean deployment in August 2008, we placed 8 nodes in the reef for a few hours most days over the course of two weeks. We were unable to leave the system unattended due to concern over theft.

Flood Prediction

River flood prediction intends to warn communities of incoming river floods, gaining time for them to protect their property and evacuate [2]. A sensor network provides such predictions through local sensing and self-calibrated models allowing much easier installation and operation in any river basin in the world. The system measures rainfall, air temperature, and water pressure to predict river level 24 hours in advance.

This application modifies the base hardware as shown in Figure 8. All the sensors are external to the system so an extension board is needed to connect with them; each also uses the system's variety of access options as temperature is resistive, rainfall is interrupt-driven, and pressure is a RS485-based measurement. While local sensing clusters communicate via the AC4790 radios, to cover the large geographic area of the basin, some nodes communicate via 144 MHz radios. These radios require a modem, which we place on an expansion board used only in those few nodes.

Figure 9 shows the software additions necessary. Because we do not use the GPS or many of the internal sensors but do use the same sensor layer, the base code supplies 90% of the system code with the additions being the distributed modeling and prediction algorithms.

We deployed the river flood system in two locations: Massachusetts and Honduras. At the Massachusetts site, we placed 5 sensors in a square kilometer of area for 5 weeks during fall 2010. In Honduras, we deployed 6 nodes covering 100 square kilometers for 4 weeks. One node provided the office interface, 2 nodes provided 144 MHz communication, and 4 nodes provided sensing.

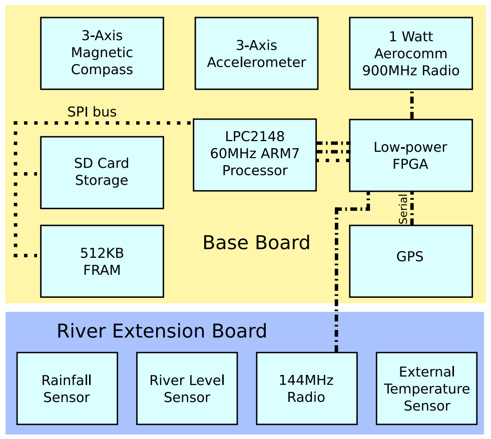

Figure 8. The core hardware usage and extensions for the river flooding project.

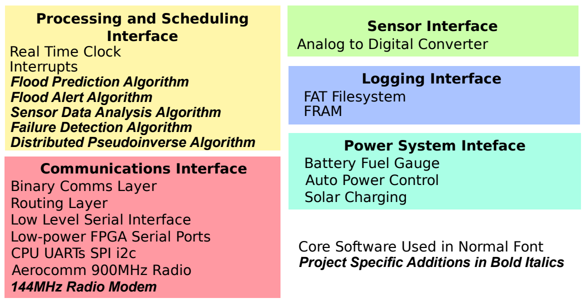
Processing and Scheduling Interface
Real Time Clock
Interrupts
Flood Prediction Algorithm
Flood Alert Algorithm
Sensor Data Analysis Algorithm
Failure Detection Algorithm
Distributed Pseudoinverse Algorithm
Sensor Interface
Analog to Digital Converter
Logging Interface
FAT Filesystem
FRAM
Power System Inteface
Battery Fuel Gauge
Auto Power Control
Solar Charging
Communications Interface
Binary Comms Layer
Routing Layer
Low Level Serial Interface
Low-power FPGA Serial Ports
CPU UARTs SPI i2c
Aerocomm 900MHz Radio
144MHz Radio Modem
Core Software Used in Normal Font
Project Specific Additions in Bold Italics

Figure 9. The core software usage and extensions for the river flooding project.

5 Lessons Learned

Our goal has been to develop an easily reconfigurable sensor network architecture. We have learned several lessons designing and using our system.

The startup costs of designing our multi-functional system have been high both on the hardware and the software sides. With such a variety of needs, we found it difficult to initially design each part with the necessary flexibility, often requiring development first for one project style and later modification for the other. However, by sharing the same base hardware and software, debugging is very fast. This shared debugging, supporting at least 80% of the software and at least as much of the hardware, far outweighs the difficult startup costs such that we suggest others use, and intend to continue using ourselves, multi-application platforms.

The predominance of serial peripherals ensures that no processor exists that provides enough serial connections. Having some form of external serial multiplexer is necessary, whether it is a simple serial multiplexer, a SPI-to-UART converter, or a more complicated FPGA as we use. This allows for simultaneous use of several communication methods and sensors, a situation that has arisen in all three of our applications.

Smart power management enables continuous operation for the applications we instantiated. However, it is easy to be optimistic about the amount of recharging available and the low power usage of the system. As we saw, the available solar current varied widely; had we designed for the worst case situation, we would be unable to take advantage of the significant solar current of the other locations. We needed, and recommend, the ability to control many aspects of the power system to intelligently maximize the operation of the system and the lifetime.

6 Conclusions

We designed a multi-functional sensor network platform that enables a large and heterogeneous range of applications in the air, on the ground, and in the water. This system allows easy addition of sensors and communication types, reconfiguration of nodes, data storage and access, and user operation. This aids new application development on the platform.

We characterized our system through three application areas: virtual fencing of cattle, coral reef monitoring, and river flood prediction. Each of these applications was deployed in the field, providing a wide variety of data on the base system in addition to application specific data. We are currently exploring the best ways of sharing the hardware and software of our sensor network platform with the community.

7 References

[1] R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C. Wan, and M. Yarvis. Intel Mote 2: an advanced platform for demanding sensor network applications. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, USA, 2005. ACM.

[2] E. Basha, S. Ravela, and D. Rus. Model-based monitoring for early warning flood detection. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, Raleigh, NC, USA, November 5-7 2008. ACM.

[3] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. Demo abstract: Operating a sensor network at 3500 m above sea level. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE Computer Society, 2009.

[4] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, TN, USA, 2006. ACM.

[5] A. Ledeczi, T. Hay, P. Volgyesi, D. Hay, A. Nadas, and S. Jayaraman. Wireless acoustic emission sensor network for structural monitoring. *Sensors Journal, IEEE*, 9(11):1370–1377, 2009.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: an operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

[7] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling. IMOTE2: serious computation at the edge. In *Proceedings of the International Conference on Wireless Communications and Mobile Computing (IWCMC)*, Crete, Greece, 2008.

[8] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel. The intel mote platform: a bluetooth-based sensor network for industrial monitoring. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, USA, 2005.

[9] Phillips. *LPC241x User Manual*, 2 edition, July 2006.

[10] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 32—43, Boston, MA, USA, August 2000.

[11] M. Schwager, C. Detweiler, I. Vasilescu, D. M. Anderson, and D. Rus. Data-Driven identication of group dynamics for motion prediction and control. *Journal of Field Robotics*, 25(6-7):305–324, 2008.

[12] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: wireless sensor network for environmental research. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, Sydney, Australia, 2007. ACM.

[13] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, USA, 2005. ACM.

[14] T. Wark, P. Corke, P. Sikka, L. Klingbeil, Y. Guo, C. Crossman, P. Valencia, D. Swain, and G. Bishop-Hurley. Transforming agriculture through pervasive wireless sensor networks. *Pervasive Computing, IEEE*, 6(2):50–57, 2007.