Fast-prototyping Using the BTnode Platform

Jan Beutel*
Swiss Federal Institute of Technology (ETH) Zurich
8092 Zurich, Switzerland
beutel@tik.ee.ethz.ch

Abstract

The BTnode platform is a versatile and flexible platform for functional prototyping of ad hoc and sensor networks. Based on an Atmel microcontroller, a Bluetooth radio and a low-power ISM band radio it offers ample resources to implement and test a broad range of algorithms and applications ranging from pure technology studies to complete application demonstrators. Accompanying the hardware is a suite of system software, application examples and tutorials as well as support for debugging, test, deployment and validation of wireless sensor network applications. We discuss aspects of system design, development and deployment based on our experience with real wireless sensor network experiments. We further discuss our approach of a deployment-support network that tries to close the gap between current proof-of-concept experiments to sustainable real-world sensor network solutions.

1. Introduction

In their seminal articles, Estrin [4] and Kahn [9] presented a far-reaching vision of wireless sensor networks (WSNs), where collections of tiny autonomous computers would collaboratively and unobtrusively monitor a variety of real-world phenomena with unprecedented quality and scale, bringing substantial benefits to a variety of application areas. Since then, numerous hardware platforms have been developed, operating system abstractions have been established, a large number of protocols and algorithms for networking, communication, and information processing have been proposed, and various fundamental capabilities and limitations of these sensor networks have been examined. Based on these ingredients, prototypical applications, e.g. [8, 14, 15], have been developed, some of which consist of more than 100 sensor nodes. Such networks are formed from a set of small sensor devices, the nodes, that are deployed in an ad hoc fashion and cooperate in sensing a physical phenomenon.

Initially, these demo applications form a proof-of-concept of the visions and suggest the basic feasibility and applicability of this novel platform class of tiny, wireless embedded systems to real applications. On the other hand, these experiments have revealed the complexity of cross-layer design on extremely resource constrained devices with their tight coupling of application, nodes and the environment. However, taking a closer look at the development process of such prototypical applications reveals that putting such a running sensor network in place is currently an art. Reports from implementations such as Great Duck Island (GDI) [16] or the macroscope in the redwoods [17] document in an impressive way the difference between the rather clean and easy world of simulations and theoretical studies opposed to the outdoors deployment case with many sources for errors, failures and imperfections found in the real world. While GDI suffered from random node failures and the need for additional hardware deployments for calibration and monitoring of the wireless sensor network, the redwood forest deployment reported on being only able to retrieve 40% of the sensor data in usable form due to many influences on the system and various sources of error.

Nevertheless, such experiments are valuable sources of experience and necessary to validate concepts that cannot be accounted for in models and simulations where access to and interaction with the environment is limited. In order to close the gap between current proof-of-concept and real-world sensor networks, this artwork has to be replaced with an efficient, coordinated design and development process to be able to achieve industry-grade quality in applications. The BTnode platform [3] has been specifically designed for functional prototyping of wireless networking applications at different layers: Initially both ubiquitous computing scenarios characterized by interaction with human users and ad hoc and sensor networking experiments characterized by

* The work presented here was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

3-9810801-0-6/DATE06 © 2006 EDAA

Figure 1. The BTnode rev3 uses a four layer PCB with all components mounted on the top and a 2xAA battery case underneath.

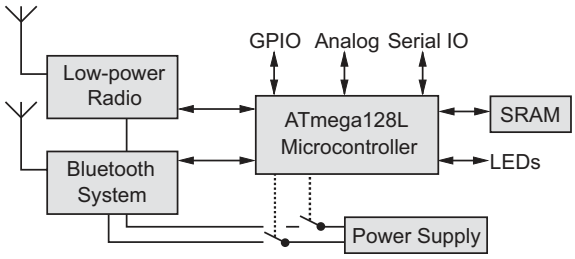

Figure 2. The BTnode rev3 is based on two wireless communication systems, an Atmel microcontroller computational core and generic IO peripherals.

large amounts of devices and self-organization necessitated design goals such as flexibility, easy of use, a steep learning curve, accessibility, transparent debugging capability, visibility and versatility.

2. The BTnode Platform

The BTnode is an versatile, lightweight, autonomous wireless communication and computing platform based on a Bluetooth radio and a microcontroller. The BTnode rev3 [1] features an additional low-power radio, generic IO peripherals and switchable power conversion and distribution systems. The low-power radio is the same as on the Berkeley Mica2 Motes [5], making the BTnode rev3 a twin of both the Mica2 Mote and the older BTnode rev2. Both radios can be operated simultaneously or be independently powered off completely when not in use, considerably reducing the idle power consumption of the device. Being the only dual-radio platform for sensor networks available today, the BTnode rev3 is ideally suited for versatile and flexible functional prototyping of a broad range of applications with the tradeoff possibility of the two radios and the flexibility offered by ample memory resources (see Figure 2). The dual-radio approach provides opportunities to create tiered architectures with high-bandwidth nodes bridging ultra-low-power devices like the Berkeley Motes to Bluetooth-enabled gateway appliances [5], or to investigate duty-cycled multi-front-end devices with wake-up radios [13] or bandwidth-power-latency trade-offs.

2.1. BTnode rev3 Features at a Glance

- **Microcontroller –** Atmel ATmega 128L (8 MHz @ 8 MIPS)
- **Memories –** 64 + 180 Kbyte SRAM, 128 Kbyte FLASH ROM, 4 Kbyte EEPROM

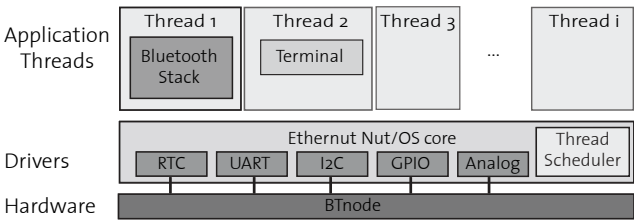

Figure 3. The BTnut operating system provides cooperative multi-threading with POSIX-like C interfaces.

- **Bluetooth subsystem –** Zeevo ZV4002, supporting AFH/SFH scatternets with max. 4 piconets/7 slaves, Bluetooth v1.2 compatible
- **Low-power radio –** Chipcon CC1000 operating in ISM band 433-915 MHz
- **External interfaces –** ISP, UART, SPI, I2C, GPIO, ADC, Timer, 4 LEDs
- **Dual power supply –** 2xAA cells with step up converter or DC input 3.6-5.0 V

2.2. BTnut: Lightweight Software Support

Based on the requirements outlined in the introduction, the software support for the BTnodes is designed for functionality and versatility across different applications and catering to the needs of different developers. Therefore, simple and intuitive programming without the need for special development tools, languages or compilers are of primary importance.

The C-based BTnut system software is built on top of a threaded OS core for embedded systems, the Opensource Ethernut Nut/OS (see Figure 3). The basic support of this

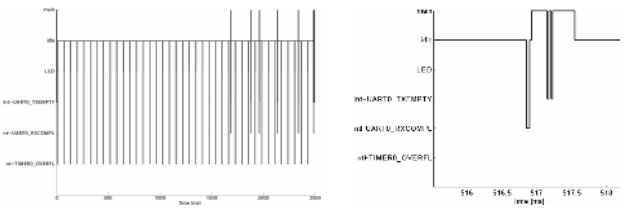

Figure 4. The BTnut OS tracer allows to track critical real-time issues (interrupts and thread switches shown here) on a target device without minimal interference.

OS core are primitives for scheduling multiple threads, basic memory management, events, synchronization, streaming IO and device drivers that allows an extremely fast jump-start, even on complex applications.

Compared to the popular TinyOS operating system [6], the BTnut system software does not require to install and learn new languages and tools (nesC) but uses plain C based programming and is based on standard operating systems concepts that are familiar to most developers. In combination with a developer kit and accompanying tutorial as well as community support through a Wiki based project web page and an archived mailing list this ensures a quick jump-start and accelerated learning curve.

2.3. Debugging with Event Traces

Event traces are a versatile and uncomplicated way to debugging and profile applications at different levels of abstraction. A separate library can be included to a BTnut application under investigation. It allows to detect and log arbitrary events, e.g. interrupts, context switches, critical sections, application context etc. All events are simply time-stamped and recorded in an internal buffer from where they are retrieved for offline analysis (see Figure 4). This strategy allows maximum transparency for profiling and tracing while only minimally altering the timing behavior of an application and so allows to debug complex, interactive applications between multiple communicating nodes.

2.4. In-situ Power Profiling

The switchable power supply on the BTnode rev3 (see Figure 2 offers direct current access for in-situ profiling of the power consumption of both the radio systems and the microcontroller core under real-life operating conditions. This can be used for detailed performance analysis and the tuning of operating mode and parameters, e.g. of communication protocols [11] (see Figure 5).

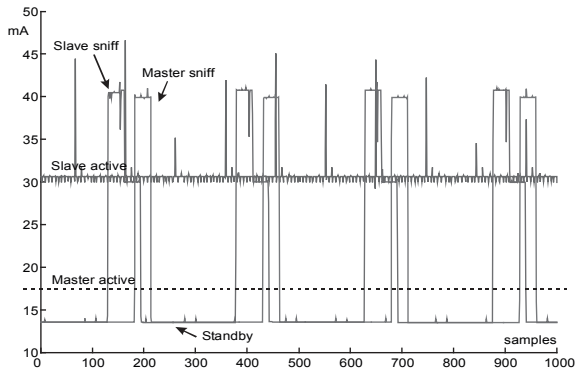

Figure 5. Detailed power profile of the Bluetooth radio at different operating modes.

3. State of the Art Platforms Compared

When comparing other state of the art platforms for sensor networks (Crossbow Mica2 and Mica2Dot Motes [5], Moteiv Tmote Sky [12] and Intel Imote [10]) a considerable bias towards specific niche application requirements becomes apparent (see Figure 6). The investigation of the properties of the radio systems used shows that there are two different paradigms followed by the competitors platforms; either a low-level "modem-like" bit-stream oriented radio (Chipcon CC1000) or a high-level, packet-oriented radio (Chipcon CC2420 or Bluetooth) is used. The features of the system core reveal in part considerably high processing capabilities and the ability to store complex programs while there is an apparent lack of program memory (SRAM) and thus a lack of flexibility for the application developer. Platforms allowing for $\geq$128 Kbyte program sizes but only supporting a few kilobytes of program memory are clearly limited in versatility and functionality not only during the development but also in a production phase.

The BTnode rev3 (shown on the left in Figure 6) offers a more balanced set of system resources with ample program memory and draws the best of both worlds with the two radio systems either operating in a tradeoff according to the performance requirements or simultaneously.

4. Next-Generation Deployment-Support for Sensor Networks

Classic approaches to develop and deploy wireless sensor networks use serial or ethernet cables for program download, control and monitoring [18]. Although successful in lab setups, this approach is limited due to scalability issues and completely infeasible for deployment in the field. Distributing firmware updates within a sensor network [7] requires nodes to be equipped with buffering and

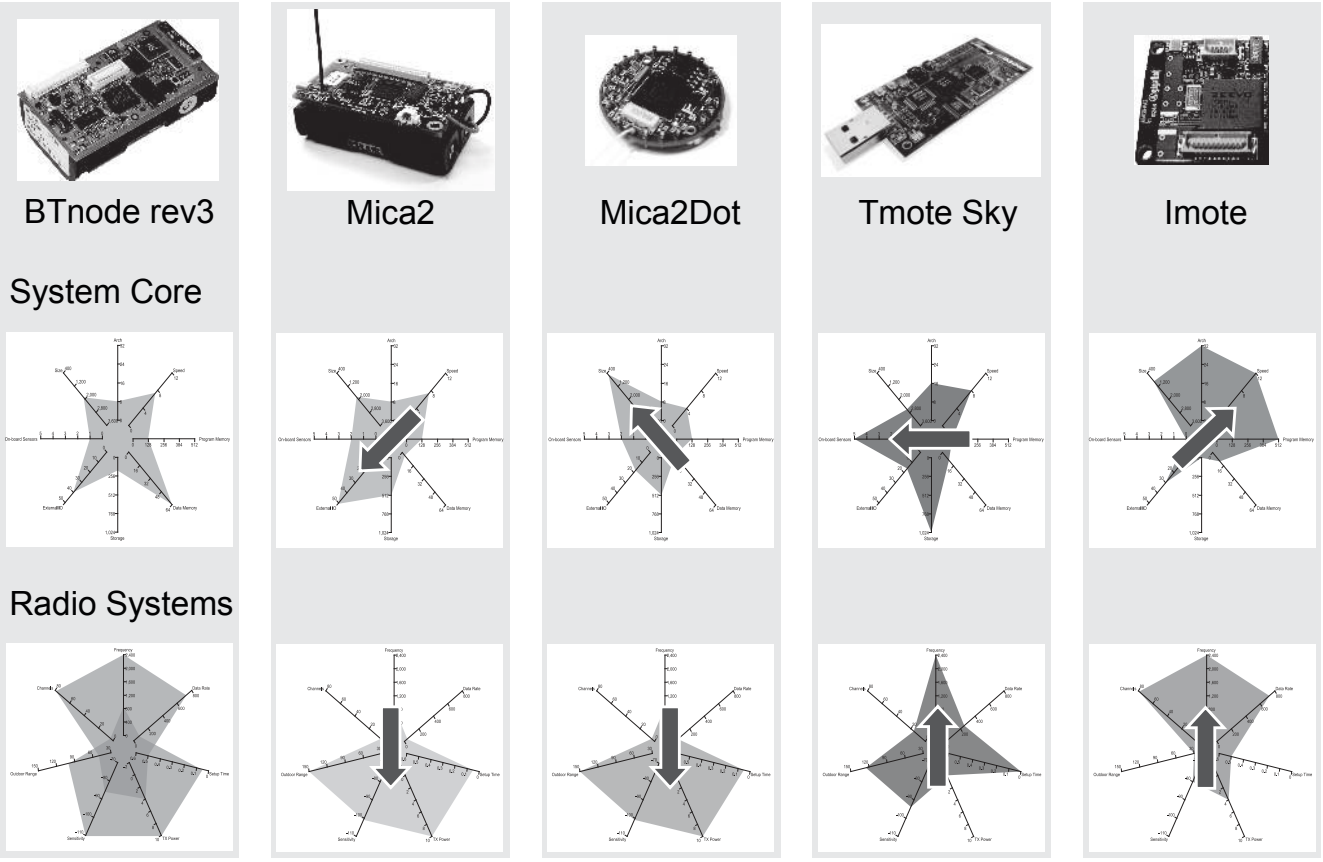

Figure 6. The BTnode rev3 offers a balanced resource mix with ample memory resources and two radio interfaces whereas competing platforms are more biased towards a specific application.

self-reprogramming support and often exhibit an excessive burden on the network itself, with heavy traffic compared to the average network operation and long latencies due to low power duty-cycling.

4.1. Deployment-Support Network

The deployment-support network (DSN) (see Figure 7) is a new methodology for the development, test, deployment, and validation of wireless sensor networks [2]. A DSN is a robust, wireless cable replacement offering reliable and transparent connections to arbitrary sensor network target devices. DSN nodes are battery powered nodes that are temporarily attached to some or all target nodes in a sensor network deployment under test. A target adapter on the DSN node is responsible for target control, (re-) programming and logging while a small monitor running on the target sensor node is responsible to output events and status information to the DSN node where it is logged and timestamped. Examples of such logged context are packet arrivals, sensor values as references for calibration, interrupts on the target node or error codes for debugging. Compared to traditional serial-cable approaches, this approach results in enhanced scalability and flexibility with respect to node location, density, and mobility. This makes the coordinated deployment and monitoring of sensor networks possible.

4.2. Sensor Network Maintenance Toolkit

In order to employ deployment-support network for the development and deployment of a sensor network application, the sensor network maintenance (SNM) toolkit has been devised as a set of sophisticate services that can be easily adapted and customized according to the maintenance and monitoring requirements. The SNM toolkit contains services for:

- Target Control
- Remote Programming

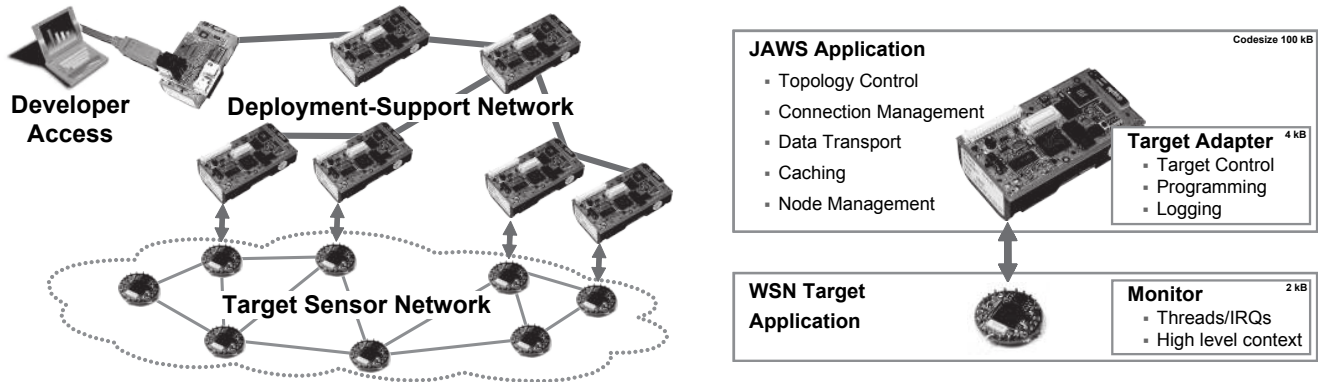

Figure 7. A deployment-support network is temporarily attached to as experimental target network and facilitates long-term surveillance and maintenance using the sensor network maintenance toolkit. Developers can access the DSN resources using the Bluetooth backbone network.

- Generic DSN Access
- Remote Logging and Event Detection
- Long Term Logging and Data Analysis

The current reference implementation of a deployment-support network is called JAWS and runs on 30 BTnode rev3 devices in a permanent installation at ETH Zurich.

4.3. BTnode Platform Success

The BTnode platform has been successfully used for both research and engineering education. To date, the hardware has been used by 30+ research groups leading to many successful student projects, courses and labs (e.g. a graduate lab in embedded systems design with 120 participants) research demos and publications (40+ scientific publications at ETH Zurich alone). The BTnode rev3 has been made commercially available through an industrial partner that is responsible for the logistics, manufacturing and testing of the hardware.

Due to it's reliability and versatility the BTnode platform has been especially popular in ubiquitous computing experiments and long-term ad hoc networking deployments. Although the early Bluetooth hardware has been arguably not optimal in terms of power consumption, the available Bluetooth devices have matured considerably and have proven a good and legitimate choice for the design goals of the BTnode platform: (i) The high-level abstraction of the event-based interface makes the radio easy to use with relaxed real-time constraints when compared to a "modem-like" bitstream interface on the CC1000 radio where the microcontroller is responsible both for the control-flow of the application and the (low-level) protocol processing. (ii) When transferring larger amounts of data in a duty-cycled fashion, the increased throughput of the Bluetooth radio adds favorably to the power-performance figures when regarded from a system and application perspective. Moreover the Bluetooth interface relieves the application developer from many low-level implementation issues and offers reliable, buffered link-layer data transfers with advanced features such as error correction and authentication.

In retrospective, the BTnode platform is living up to the initial design requirements and sets standards within the wireless sensor network community in terms of functionality and a sound mix of resources that allow for timely development, easy debugging and reliable function. In combination with the deployment-support network and the sensor network maintenance toolkit it forms a powerful and effective suite of fast-prototyping and validation tools offering full life-cycle support for sensor network applications.

Acknowledgments

I would like to acknowledge the invaluable hard work and tireless debugging of the BTnode core team, it's many student contributors as well as Luca Negri (power profiling) and Philipp Blum (event tracer) that have made the BTnode platform into a success.

References

[1] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald. Next-generation prototyping of sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 291–292. ACM Press, New York, Nov. 2004.

[2] J. Beutel, M. Dyer, L. Meier, and L. Thiele. Scalable topology control for deployment-sensor networks. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 359–363. IEEE, Piscataway, NJ, Apr. 2005.

[3] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund, and L. Thiele. Prototyping wireless sensor network applications with BTnodes. In *Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*, volume 2920 of *Lecture Notes in Computer Science*, pages 323–338. Springer, Berlin, Jan. 2004.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. 5th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom '99)*, pages 263–270. ACM Press, New York, Aug. 1999.

[5] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. Wireless sensor networks: The platforms enabling wireless sensor networks. *Communications of the ACM*, 47(6):41–46, June 2004.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. 9th Int'l Conf. Architectural Support Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104. ACM Press, New York, Nov. 2000.

[7] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 81–94. ACM Press, New York, Nov. 2004.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. 10th Int'l Conf. Architectural Support Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107. ACM Press, New York, Oct. 2002.

[9] J. Kahn, R. Katz, and K. Pister. Next century challenges: Mobile networking for smart dust. In *Proc. 5th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom '99)*, pages 271–278. ACM Press, New York, Aug. 1999.

[10] L. Nachman, R. Kling, R. Adler, J. Huang, and V. Hummel. The Intel mote platform: A Bluetooth-based sensor network for industrial monitoring. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 437–442. IEEE, Piscataway, NJ, Apr. 2005.

[11] L. Negri, J. Beutel, and M. Dyer. The power consumption of Bluetooth scatternets. In *Proc. IEEE Consumer Communications and Networking Conference (CCNC 2006)*, page to appear. IEEE, Piscataway, NJ, Jan. 2006.

[12] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 364–369. IEEE, Piscataway, NJ, Apr. 2005.

[13] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proc. 6th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom 2001)*, pages 160–171. ACM Press, New York, Sept. 2002.

[14] G. Simon, G. Balogh, G. Pap, M. Maróti, B. Kusy, J. Sallai, Á. Lédeczi, A. Nádas, and K. Frampton. Sensor network-based countersniper system. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 1–12. ACM Press, New York, Nov. 2004.

[15] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, June 2004.

[16] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*, volume 2920 of *Lecture Notes in Computer Science*, pages 307–322. Springer, Berlin, Jan. 2004.

[17] 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 *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, pages 51–63. ACM Press, New York, 2005.

[18] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A wireless sensor network testbed. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 483–488. IEEE, Piscataway, NJ, Apr. 2005.