A Structured Hardware/Software Architecture for Embedded Sensor Nodes

Geoff V. Merrett*, Alex S. Weddell, Nick R. Harris, Bashir M. Al-Hashimi, and Neil M. White
Pervasive Systems Centre, ECS, University of Southampton, SO17 1BJ, UK
{gvm, asw05r, nrh, bmah, nmw}@ecs.soton.ac.uk

***Abstract*—Owing to the limited requirement for sensor processing in early networked sensor nodes, embedded software was generally built around the communication stack. Modern sensor nodes have evolved to contain significant on-board functionality in addition to communications, including sensor processing, energy management, actuation and locationing. The embedded software for this functionality, however, is often implemented in the application layer of the communications stack, resulting in an unstructured, top-heavy and complex stack. In this paper, we propose an embedded system architecture to formally specify multiple interfaces on a sensor node. This architecture differs from existing solutions by providing a sensor node with multiple stacks (each stack implements a separate node function), all linked by a shared application layer. This establishes a structured platform for the formal design, specification and implementation of modern sensor and wireless sensor nodes. We describe a practical prototype of an intelligent sensing, energy-aware, sensor node that has been developed using this architecture, implementing stacks for communications, sensing and energy management. The structure and operation of the intelligent sensing and energy management stacks are described in detail. The proposed architecture promotes structured and modular design, allowing for efficient code reuse and being suitable for future generations of sensor nodes featuring interchangeable components.**

***Index Terms*—sensor nodes, embedded software, protocol stacks**

I. Introduction

EMBEDDED sensor nodes sense parameters in their surrounding environment, and communicate them over a network, normally to a central data 'sink'. Nodes may communicate through a wired or, as is becoming more common, wireless medium. Conventional wired embedded sensor nodes normally have a separate, dedicated hardware module for network communications. This is commonly known as the Network Capable Application Processor (NCAP), which provides a bridge between intelligent sensors and the network.

Wireless sensor nodes are inherently resource-constrained as they are usually powered by batteries or harvested energy. They must manage their resources carefully, often adapting their behavior dependent on their energy status. They must also interface with transducers and autonomously process this information into a form suitable for transmission over the network. The resource-constrained nature of wireless sensor nodes, and the fact that they are developed around radio transceivers, has given prominence to the communications software stack. Other functions, such as sensing and energy management, are inelegantly appended to the communications software stack. This issue is discussed in detail in section II, but in short it has meant that the top (application) layer of the software stack has become large and complex.

Furthermore, moves towards integrating wireless sensors into system-on-chip devices has resulted in products such as the Texas Instruments CC2431 [1], which features a radio transceiver and enhanced 8051-based microcontroller in a single surface-mount package. It has multiple reconfigurable I/O pins and onboard voltage converters. This means that the chip incorporates communications, energy management, and potentially sensor processing operations in a single integrated circuit. So, whilst hardware is integrating and becoming more capable, the embedded software on such devices remains dominated by the communications software stack (with its resultant problems).

The TinyOS operating system [2] (used in many wireless sensor nodes, such as the Crossbow motes [3]) has a layered software stack, where data from the sensors (and any other hardware) is directly processed in the application layer [4]. In energy harvesting nodes such as Heliomote [5] and Prometheus [6] the application must monitor voltage levels and incorporate cursory temperature compensation. In these nodes, such energy management operations occur in the application layer. These examples demonstrate that, while communication gets formally specified in the majority of implementations, other interfaces (such as sensing and energy management) are relatively unstructured.

In this paper, we propose an embedded system architecture to formally specify multiple interfaces on a sensor node. The architecture provides a sensor node with multiple stacks (each implementing distinct functionality), which are linked by a shared application layer. These linked stacks form a 'unified' stack that manages the complete functionality of the node.

Manuscript received March 9, 2008. This work was supported in part by the Engineering and Physical Science Research Council (EPSRC) under grant number EP/D042917/1.

*Corresponding Author, tel +44 (0)2380 594996, fax +44 (0)2380 593709

This establishes a structured platform for the formal design, specification and implementation of modern sensor, and wireless sensor, nodes. Use of the proposed architecture enables the management of all major node functions with the benefits of a structured, layered model. These benefits include:

- Structuring the node as a system of its component parts, and promoting modular design.
- Permitting efficient code reuse, reducing the design time of future projects.
- Promoting the development and standardization of interchangeable protocols.

This paper is structured as follows. Section II provides an overview of communication protocols models, and section III describes how these concepts are used and extended in the architecture proposed in this paper. Section IV describes an energy-aware intelligent sensing node that has been implemented using this architecture; a detailed overview of the energy-management stack (section V) and intelligent sensing stack (section VI) are given. Section VII draws conclusions and details the future directions of this research.

II. Communication Protocol Models

Communication protocol models have been utilized for decades: aiming to formalize, structure, and provide interoperability between the tasks of the networking system. The OSI Basic Reference Model (OSI-BRM) [7] was introduced in the mid-1970s, proposing a basic layered architecture for communication protocols – as shown in Fig. 1. Many widely-used communication stacks and models (for example the Internet Reference Model [8], shown in Fig. 1) have added, removed and merged layers from the OSI-BRM to tailor the model to their requirements. However, the OSI-BRM remains an important building block for modern communication protocols. The OSI-BRM was originally designed to represent only the communications functionality of a networked node. This was no longer found to be true in the majority of the models and stacks investigated.

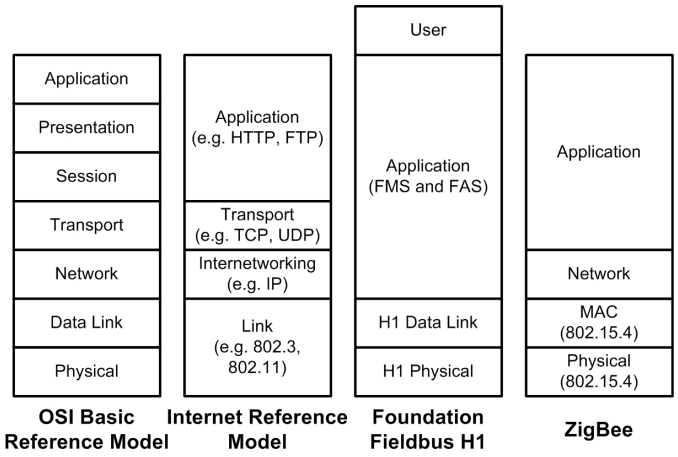

Fig. 1. The OSI-BRM [7], IRM [8], Foundation Fieldbus H1 [9] and ZigBee [10] communication stacks.

The ZigBee specification [10] defines a low-cost, low-power wireless communication standard, particularly suited to wireless sensor networks. The ZigBee protocol stack (shown in Fig. 1) uses the Physical (PHY) and Medium Access Control (MAC) layers from the IEEE 802.15.4 standard [11]. The transport layer is omitted, with the relevant functionality being absorbed by neighboring layers. ZigBee uses the communications stack to contain software for the entire device, placed in a large and subdivided application layer.

Fieldbus H1 [9] was designed as a network architecture for process control applications such as sensor networks. It can be seen from Fig. 1 that the majority of higher layers have been removed, as their functionality (for example packet routing, flow control and connection management) is not required by networks. Fieldbus adds a User layer to the top of the stack, allowing additional functionality to be provided, such as a user interface. Fieldbus networks place the program in the application or user layer, or alternatively use 'off-chip' software that does not promote reuse or interoperability.

Recently, cross-layer protocols and algorithms (which claim to improve efficiency through the merging of some or all layers of the OSI-BRM) have received considerable research interest [12]. The incorporation of additional stacks in the embedded software of a sensor node, as proposed here, introduces minor overheads in terms of space, computational time, and power consumption. We believe that the modularity (often lost through cross-layer design) that is gained for all interfaces on a node through our proposed architecture is worth the minor loss in efficiency. Cross-layer optimization techniques could still be applied to the multiple stacks of the architecture as part of the protocol development phase.

III. The Proposed Architecture

In the previous sections it has been shown that, while the communications interface is formally structured, other interfaces (such as sensor processing and energy management) are generally unspecified and often integrated into a large and complex application layer. This is not ideal for intelligent sensing and energy management, which are arguably as important as communications to the functionality of a node.

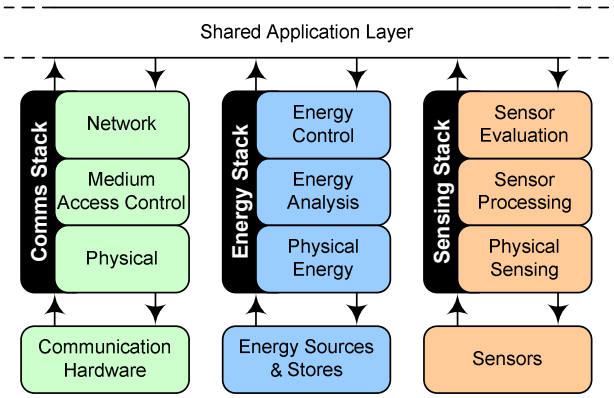

Fig. 2. A 'unified' stack incorporating communication, energy management and intelligent sensing interfaces.

We propose an embedded architecture, which specifies the structure of multiple interfaces on sensor nodes. The architecture uses a basic template stack, based upon the principles of the OSI-BRM, from which different interface stacks can be derived. A number of these stacks, for distinct tasks, are combined via a shared application layer. This forms a 'unified' stack, such as that shown in Fig. 2.

The proposed architecture's basic template stack (from which individual stacks are derived) is shown in Fig. 3. The layer boundaries are placed to accommodate a wide range of hardware interfaces, and to allow protocols to be interchanged without redefining surrounding layers.

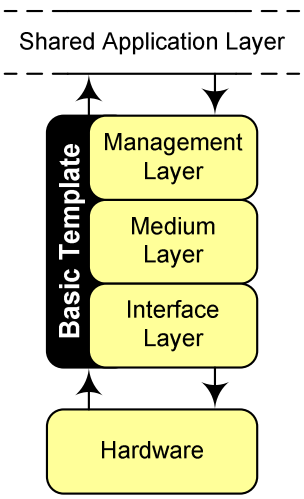

Fig. 3. The basic template stack defined by the proposed architecture.

The functions of the basic template stack's layers are:

- **Interface Layer:** directly interfaces with the relevant hardware, and hides the complexity of the circuitry from the higher layers. For communications, this layer represents the physical layer of the OSI-BRM.
- **Medium Layer:** provides low-level processing, and hides the complexity of interfacing with the medium (for example, the wireless channel or the sensed phenomenon) from the higher layers. For communications, this layer represents the data link layer of the OSI-BRM.
- **Management Layer:** provides high-level functions, and hides the complexity of groups of objects (for example a network of nodes, or number of different energy sources) from the higher layers. For communications, this represents both the network and transport layers of the OSI-BRM.
- **Shared Application Layer:** contains cooperative functionality that passes data between individual stacks, and also contains the end user's application. In conventional communication stacks, this layer represents the application layer in the OSI-BRM.

In order to show how the proposed architecture can be used, the remainder of this paper describes a developed energy-aware intelligent sensing system using this architecture.

IV. Development of an Energy-Aware Intelligent Sensing System

To demonstrate the process of creating stacks from the basic template stack, an energy-aware intelligent sensing node has been developed using the proposed architecture – shown in Fig. 4. The system consists of a Texas Instruments CC2431 evaluation module [1], which is powered from two 1.0F Panasonic electric double layer capacitors (also known as supercapacitors) [13]. The batteries shown attached to the node in Fig. 4 are for programming purposes only, and are isolated when the node is connected to the energy harvesting circuit. These supercapacitors are charged efficiently from a Schott amorphous silicon solar cell [14].

The node senses temperature (using the CC2431's onboard temperature sensor) and, based upon the sensed data and the node's energy state, reports this to another node in the network using its IEEE 802.15.4 [11] compliant wireless transceiver. Further details of the hardware in this system are outside the scope of this paper. It is however worth mentioning that, while this prototype is reasonably large (observable in Fig. 4), it will be significantly reduced in size once prototyping is complete. The power conditioning circuitry will be transferred from protoboard to a single PCB, which will enable it to interface directly with the CC2431 evaluation module (rather than going through the battery board as shown in the figure). It is envisaged that the final footprint of the node will be no larger than that of the solar cell, with its thickness dictated by the dimensions of the supercapacitors.

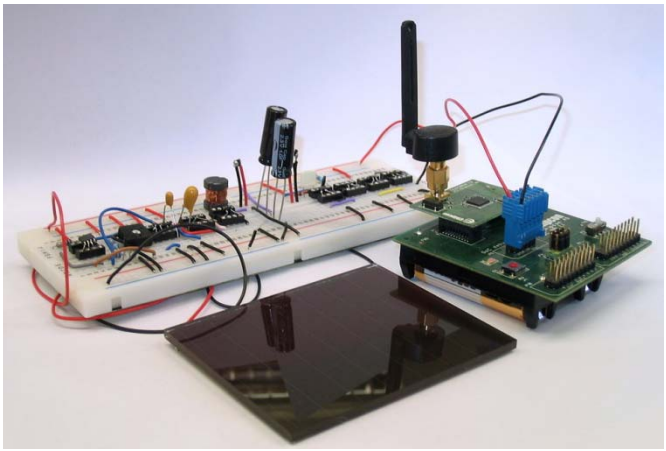

Fig. 4. The prototype energy-aware intelligent sensing node developed using the proposed architecture.

The embedded software in the developed system differs from existing solutions by devolving communications, intelligent sensing, and energy management to individual stacks; these are fundamental parts of almost all modern sensor nodes. The resulting 'unified' stack that has been developed in order to implement this is shown in Fig. 2.

The communication stack in the development is similar to that of ZigBee [10]. The physical layer interacts directly with the communication hardware, and hides the complexities of the communication circuitry from the MAC layer. The MAC

layer provides features including channel access, frame management, and low-level error detection, hiding the complexities of the communication medium from the network layer. The network layer controls message routing and subnet formation and maintenance, thus providing the shared application layer with a view of the network as a single entity.

The shared application layer handles collaboration between the individual stacks – passing high-level data between them. For example, a packet may be received via the communication stack requesting a measurement. The shared application layer would then request this data from the sensor stack and pass the result back to the communication stack for transmission.

The energy management and intelligent sensing stacks are investigated in detail in the following two sections.

V. The Energy Management Stack

The nodes in a wireless sensor network are typically resource-constrained, relying on batteries or energy harvesting to supply their energy. Where multiple energy sources are used, the management of the energy subsystem can become a complex process, with the flow of charge between sources requiring intricate control.

By partitioning the energy management process into distinct categories, the implementation can be migrated from the program area to a separate stack, removing much of the complexity from the application layer and providing a more defined structure.

A. Structure of the Energy Management Stack

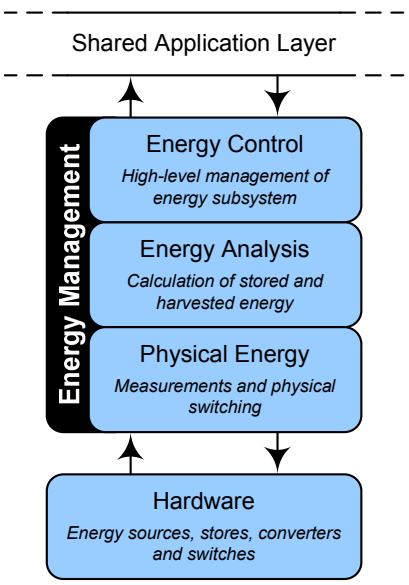

Fig. 5. A stack to implement energy management and control functionality, designed using the proposed architecture.

A unified stack, such as that discussed in section IV, can include an energy management stack (as shown in Fig. 5). Here, the energy management process is divided into three layers:

- **Physical Energy Layer (PYE):** interfaces with *physical* energy hardware in order to monitor and control the provision and consumption of energy. This includes monitoring energy sources (for example solar harvesting and mains electricity) and stores (for example batteries and supercapacitors), and controlling physical energy switching in order to direct the flow of energy. Fundamentally, this layer masks the complexities of interfacing with the physical hardware from the EAN.
- **Energy Analysis Layer (EAN):** *analyses* data obtained from the PYE, in order to provide the ECO with high-level information on the state of the energy components. This includes using energy models to convert voltage and current readings from the PYE into meaningful values, such as the residual energy in each energy store, and the yields obtained from energy sources. Fundamentally, this layer masks the physical voltage and current readings from the ECO.
- **Energy Control Layer (ECO):** takes a high-level view of the energy subsystem, *controlling* the node's energy aware operation. This includes making decisions about how the node should operate (based upon the state of its collective energy resources), and controlling the flow of energy between energy components (viewed from a high level).

These layer boundaries are placed in order to modularize distinct levels of abstraction in the energy-management process. The process of using this energy stack in the system described in section IV is detailed below.

B. Implementing an Energy Management Stack

In the developed system, the PYE obtains a reading of the voltage across the supercapacitor energy store using the CC2431's onboard ADC, and provides this to the EAN. In the future, it is envisaged that a secondary (rechargeable) battery will be included on the node to enable operation during unusually lengthy periods of darkness, or a rapid increase in the number of packets being communicated; this dual store architecture is similar to that proposed by Jiang *et al.* [6]. In this situation, the PYE will have to monitor the voltages across both energy stores, and also implement physical switching to control the flow of energy between the two. Furthermore, it is planned that the solar cell will be monitored by the PYE in order to calculate the energy that it is yielding, to allow intelligent algorithms elsewhere in the energy stack to predict the availability of future harvested energy.

The EAN converts the received supercapacitor voltage reading into an estimation of the 'residual' energy in the store (the number of Joules that are estimated to be remaining in the supercapacitor), by using an energy store model. For the supercapacitor, a derivative of the simple $E = 1/2\,CV\text{^}2$ relationship can be used. Using this estimation, the EAN can provide the ECO with a prediction of the 'remaining usable energy' (this is not necessarily equal to the residual energy),

and the 'remaining lifetime'. It is anticipated that future implementations of the EAN will use information from the PYE concerning energy source yields in order to make predictions on the rate and frequency of energy generation. Additionally, if a battery was added, a complex energy store model to relate the battery voltage to residual energy would be required [15].

The ECO evaluates the 'remaining usable energy' estimation provided by the EAN to assign an Energy Priority (EP) value to the node [16]. The EP is a function of the node's residual energy, and directly influences the operation of the node (a node with more energy will participate more in performing the tasks of the network). In the implementation, the EP is calculated linearly from the 'remaining usable energy'. It is anticipated that in the future, this calculation will also take into account the rate of energy generation and consumption, and the future availability of energy (using predictive techniques). Additionally, it is envisaged that this layer will make decisions about the components of the energy subsystem, controlling the transfer of energy between different sources and stores. It is important to control this flow, as rechargeable batteries are generally limited to only 300-500 recharge cycles; this means that energy transfer operations must be minimized to prolong the lifetime of the platform [6].

Obviously, the tasks performed by each layer (and the future tasks it is anticipated that they will perform) are only those desired for this implementation; there is considerable scope for implementing various energy monitoring, analysis and control techniques into all layers of the energy stack.

VI. The Intelligent Sensing Stack

In a modern sensor node, the scope for on-board sensor processing is considerable. Instead of the node simply reporting every sampled value, an intelligent sensing system can also infer estimations of error and uncertainty, incorporate localized data fusion, and provide event detection.

By devolving the intelligent sensing tasks into a separate multi-layer stack, the functionality can be structured at logical levels of abstraction. The details of this layer structure are provided below.

A. Structure of the Intelligent Sensing Stack

The sensor processing stack (shown in Fig. 6) is split into three layers:

- **Physical Sensing Layer (PYS):** interfaces with *physical* sensor hardware by activating sensors and obtaining readings via the node's onboard ADC. The PYS is also responsible for controlling any configuration inputs of intelligent sensors, under the abstract control of the higher layers. Fundamentally, this layer masks the complexities of interfacing with the physical hardware from the SPR.
- **Sensor Processing Layer (SPR):** *processes* sensor data obtained from the PYS, in order to provide the SEV with high-level and meaningful sensor readings. This involves converting the voltage, current or digital reading obtained from the PYS into a meaningful physical value through the use of models, and calculates estimates of error and uncertainty (which are of considerable use in a multi-sensor fusion context [17]). Fundamentally, this layer masks the physical sensor readings from the SEV.
- **Sensor Evaluation Layer (SEV):** interfaces with the SPR to *evaluate* the significance of the sensed data and identify faults in the sensor hardware. This includes using event detection techniques and information quantification algorithms to evaluate data, and high level sensor models to identify faults. The SEV provides an interface with the shared application layer, notifying it of detected data and responding to queries regarding the status of the sensors and their most recent values.

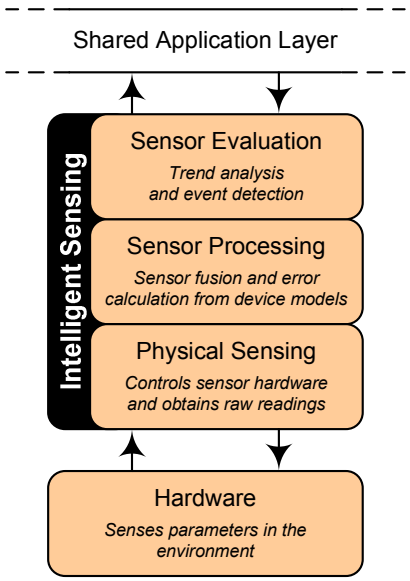

Fig. 6. A stack to implement sensor processing functionality, designed using the proposed architecture.

The process of using this intelligent sensing stack in the system described in section IV is described below.

B. Implementing an Intelligent Sensing Stack

In the developed system, the PYS activates a temperature sensor and ADC, and obtains a voltage reading from the sensor. The PYS layer will, as future work to this research, manage activation and data acquisition of a number of different sensors. It will also be responsible for controlling interfaces with intelligent sensors; for example, in simulation the PYS layer has been used to control a hardware switch in a light sensing circuit to vary the range and sensitivity. These control instructions are received from the SPR, in response to the data that the PYS provides to it (detailed below).

The SPR scales and offsets the raw sensor reading in order to convert the measured voltage into a temperature reading. To increase the accuracy of the reading, it is also necessary to take an average over a number of samples, and this is controlled by and performed in the SPR [18]. The SPR also

provides an indication of the possible error (currently simply taken from the datasheet) to the SEV. It is envisaged that future developments to the SPR will introduce localized data fusion; for example fusing temperature and pressure data, in order to provide temperature compensation. The SPR should also make sure that each sensor is managed (where available) in order to obtain the best reading; for example, in simulation the SPR layer has been used to monitor the measured light level (using a photodiode circuit that is sampled via the PYS) and, when it becomes saturated or quantized, instructs the PYS to adjust the range and sensitivity of the sensing hardware. Additionally, the simulated SPR layer incorporates a model for signal error, which takes into account the sensor reading and ambient temperature.

The SEV applies Rule Managed Reporting (RMR) [16] to the sensed parameters provided by the SPR. This provides the shared application layer with only sensed data that are perceived to be 'events', and also attaches an importance level (referred to in RMR as a Packet Priority). The shared application layer is also able to query the SEV for the maximum, minimum and most recent sensed values. It is planned that future extensions to this layer will also use trend analysis and high-level data models in order to identify sensor faults to the shared application layer. Additionally, other event detection techniques could be used, and algorithms which modify the sampling rate of the sensors based upon the expected variation of the parameter added.

VII. Conclusions

While the communications interface is usually formally structured in embedded sensor nodes, other interfaces are generally unspecified, and often integrated into a large and complex application layer. This is not ideal for sensor processing and energy management, which are arguably as important as communications to the functionality of a node.

This paper has proposed an embedded system architecture to formally specify and structure the multiple interfaces on a sensor node. Individual stacks (derived from a basic template stack) are used to implement distinct functionality, which communicate via a shared application layer to form a 'unified' stack. The proposed architecture establishes a structured platform for the formal design, specification and implementation of modern embedded sensor nodes. This allows for efficient code reuse and encourages the standardization of interchangeable protocols. To demonstrate the use of the proposed architecture, we have presented an embedded system that has been developed, that uses a 'unified' stack to structure communication, intelligent sensing, and energy management.

While only communication, energy management, and intelligent sensing have been considered in this paper, the architecture enables additional node functions to be easily devolved into distinct stacks. Such additional stacks could implement actuation (permitting actuation commands to be expressed via the shared application layer as high-level commands) and locationing (which has already been suggested as lending itself to being modularized into a separate stack [19]).

Future research directions include the extension of the energy-management and intelligent sensing stacks (the details of which have been described in the this paper) in order to progress the presented prototype into a usable embedded sensor node. Additionally, the use of the proposed architecture to extend a 'unified' stack to represent additional node functions will be investigated.

References

[1] Texas Instruments, TI/Chipcon CC2431, "Wireless Sensor Network Zigbee/IEEE 802.15.4 SoC RF Solution with Location Engine", 2006.

[2] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," presented at 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys'03), Los Angeles, CA, 2003.

[3] Crossbow Technology Inc., San Jose, CA, "http://www.xbow.com/", last accessed Mar. 2008.

[4] D. Patnode, J. Dunne, A. Malinowski, and D. Schertz, "WISENET - TinyOS based wireless network of sensors," presented at Conf. of the IEEE Industrial Electronics Society (IECON'03), Roanoke, VA, USA, 2003.

[5] A. Kansal, D. Potter, and M. B. Srivastava, "Performance aware tasking for environmentally powered sensor networks," presented at SIGMETRICS'04/Performance: Joint Int'l Conf. Measurement and Modeling of Computer Systems, New York, NY, 2004.

[6] X. Jiang, J. Polastre, and D. Culler, "Perpetual Environmentally Powered Sensor Networks," presented at 4th Int'l Conf. Information Processing in Sensor Networks (IPSN'05), Los Angeles, CA, 2005.

[7] ITU-T X.200-199407-I, "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model", 1994.

[8] D. Meyer and G. Zobrist, "TCP/IP versus OSI," *IEEE Potentials*, vol. 9, pp. 16-19, 1990.

[9] S. Kolla, D. Border, and E. Mayer, "Fieldbus networks for control system implementations," presented at Electrical Insulation Conf. & Electrical Manufacturing & Coil Winding Technology Conf., 2003.

[10] ZigBee Alliance, ZigBee Specification, "Document 053474r17", 2007.

[11] IEEE, 802.15.4™-2006, "IEEE Standard for Information Technology - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", 2006.

[12] T. Melodia, M. C. Vuran, and D. Pompili, "The state of the art in cross-layer design for wireless sensor networks," presented at Wireless Systems and Network Architectures in Next Generation Internet, Villa Vigoni, Italy, 2006.

[13] Panasonic, "Gold Capacitors Technical Guide", 2005.

[14] Schott Solar GmbH, "ASI-OEM Solarmodules for Indoor", 2006.

[15] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-time battery models for system-level low-power design," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 630-640, 2001.

[16] G. V. Merrett, N. R. Harris, B. M. Al-Hashimi, and N. M. White, "Energy Managed Reporting for Wireless Sensor Networks," *Sensors and Actuators A: Physical*, vol. 142, pp. 379-389, 2008.

[17] P. J. Boltryk, C. J. Harris, and N. M. White, "Intelligent sensors-a generic software approach," *Journal of Physics: Conference Series*, vol. 15, pp. 155-60, 2005.

[18] Texas Instruments, Design Note DN102, "SoC Temperature Sensor", 2006.

[19] J. Hightower, B. Brumitt, and G. Borriello, "The location stack: a layered model for location in ubiquitous computing," presented at Proceedings Fourth IEEE Workshop on Mobile Computing Systems and Applications, 20-21 June 2002, Callicoon, NY, USA, 2002.