Encyclopedia of Healthcare Information Systems

Nilmini Wickramasinghe
Illinois Institute of Technology, USA

Eliezer Geisler
Illinois Institute of Technology, USA

Volume III
MRI–Z

Medical Information Science REFERENCE

MEDICAL INFORMATION SCIENCE REFERENCE

Hershey · New York

Acquisitions Editor: Kristin Klinger
Development Editor: Kristin Roth
Senior Managing Editor: Jennifer Neidig
Managing Editor: Jamie Snavely
Assistant Managing Editor: Carole Coulson
Copy Editor: Laura Kochanowski, Jeannie Porter, and Sue Vander Hook
Typesetter: Jeff Ash and Sean Woznicki
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com/reference

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Encyclopedia of healthcare information systems / Nilmini Wickramasinghe and Eliezer Geisler, editors.
p. ; cm.
Includes bibliographical references.
Summary: "This book provides an extensive and rich compilation of international research, discussing the use, adoption, design, and diffusion of information communication technologies (ICTs) in healthcare, including the role of ICTs in the future of healthcare delivery; access, quality, and value of healthcare; nature and evaluation of medical technologies; ethics and social implications; and medical information management"--Provided by publisher.
ISBN 978-1-59904-889-5 (h/c)
1. Medical informatics--Encyclopedias. I. Wickramasinghe, Nilmini. II. Geisler, Eliezer, 1942-
[DNLM: 1. Information Systems--Encyclopedias--English. W 13 E5523 2008]
R858.E52 2008
651.5'04261--dc22
2007047456

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Programming Body Sensor Networks

Talles Marcelo Gonçalves de Andrade Barbosa
Catholic University of Goiás, Brazil
University of Brasília, Brazil

Hervaldo Sampaio Carvalho
University of Brasília, Brazil

Iwens Gervásio Sene Jr.
Catholic University of Goiás, Brazil
University of Brasília, Brazil

Francisco A. de O. Nascimento
University of Brasília, Brazil

João Luiz Azevedo de Carvalho
University of Southern California, USA

Adson Ferreira da Rocha
University of Brasília, Brazil

INTRODUCTION

The miniaturization and cost reduction of microelectronic devices have been leading to the development of new technologies. Wireless sensor networks (WSNs) are one example of these new technologies. A WSN is a distributed system that is composed of autonomous units with sensing capabilities (sensor nodes), interconnected by wireless communication. WSNs have been successfully applied in the monitoring of the human body. This development led to a new concept: Body Sensor Networks (BSNs).

According to Yang (2006, pp. 5), BSNs have specific requisites, when compared to WSNs. Among these requisites, we highlight the following:

- High level of security, in order to guarantee the confidentiality of information;
- Biocompatibility and biodegradability;
- Higher sensitivity to data loss, and the consequent need for mechanisms to ensure a minimum Quality of Service (QoS);
- Need for context awareness, as the physiological variations are strongly related to the changes in the context in which the user is in;
- Low number of sensor nodes, which should, however, be usually more precise than sensors for other applications of WSNs;
- Requirement of nodes with the capability of running multiple tasks.

Pervasive monitoring demands great adaptation capability from the BSN. Moreover, cases in which the decision made by the system can be different from the decision that would be made by a healthcare professional are frequent. Therefore, besides intelligent algorithms that allow autonomous operation, BSNs need mechanisms that allow changes in their behavior in order to become a clinically useful tool. According to Baldus, Klabunde, and Müsch (2004), "the BSN has to work automatically, but has also to be always under explicit control of any clinician."

Thus, specific models that include programmability as a functional requisite are important in a software architecture designed for BSNs. However, the greatest challenge is to allow the modifications to be made not only to the structure of the software, but also in its behavior, without excluding the capability for autonomous operation of the system.

According to Barbosa, Sene, Carvalho, da Rocha, Nascimento, and Camapum (2006), two important

concepts related to programmability in BSNs are: (i) deployment-time programmability, and (ii) run-time set-up. The deployment-time programmability refers to the definition of software artifacts and algorithms that are embedded in the sensor node. In BSNs, the inclusion of this functionality requires a programming interface that is suitable for healthcare personnel, as well as intelligent compilers. Intelligent compilers should be capable of handling implicit functional and nonfunctional requisites of a program. As an example, the inclusion of mechanisms and policies for energy saving could be treated by these structures.

The run-time set-up refers to the capability for adjustments in run-time. The BSN should provide interactivity between the healthcare professional (the BSN manager) and the system. As a requisite, sensor nodes need mechanisms that allow a better control of the tasks that are being run. A possible solution is the use of data structures that allow preemptive multitasking.

The goal of this article is to present the current state of the art, regarding programmability in BSNs. Moreover, we want to present potential benefits of a paradigm shift in which healthcare professionals become the actual programmers and maintainers of the BSNs. With that in mind, we briefly present a software architecture that has been developed with the goal of allowing programmability at network and sensor node levels.

BACKGROUND

Many issues related to software for BSNs have not been discussed yet. Among them, we can mention: (i) the development of graphical user interfaces directed to healthcare personnel; (ii) the hardware abstraction layers (HALs); (iii) the standardization of services and information structuring (BSN ontology); and (iv) programmability at sensor and sensor node levels. The solutions to these problems can lead to improvement of the effectiveness of these systems.

Currently, the software used in BSNs has the following characteristics:

- They are composed of proprietary systems built based on a specific architecture (hardware), and designed for handling a single application. They have little or no modularization, and they are not committed to software development methodologies. In general, they do not employ multiprogramming. Sensor nodes are usually viewed just as sources of data, and all processing is performed in a gateway—an element that interconnects the BSN to other systems—or in a Local Processing Unit (LPU), which is an element to where the data are transmitted. Some examples of such systems are presented in Asada, Shaltis, Reiner, Sokwoo, and Hutchinson (2003), Valdastri, Menciassi, Arena, Caccamo, and Dario (2004), Linz, Kallmayer, Aschenbrenner, and Reichl (2006), Kara, Kemaloglu, and Kirbas (2006), and Chakravorty (2006).
- These systems are usually based on a generic, general purpose model. It is usually based on the NesC programming language (Gay, Levis, von Behren, Welsh, Brewer, & Culler, 2003), on the TinyOS Operating system (Hill, 2003), and on a network programming system, the Deluge (Hui & Culler, 2004). All of these systems are free of charge, and they were developed by the University of California, in Berkeley. CodeBlue (Welsh, 2006), WHMS (Jovanov, 2006), and UbiMon (ICL, 2006) are examples of designs that use the TinyOS framework. Other examples are presented in (Bauldus et al., 2004) and in (Farshchi, Nuyujukian, Pesterev, Mody, & Judy, 2006).

Regarding programmability, systems built based on TinyOS have the following limitations when applied to BSNs:

- The NesC programming language imposes a peculiar syntax, based on concepts emerged from software engineering. Without the knowledge of programming logic and the expertise in managing software components, it is virtually impossible for a nonspecialized user to use this system.
- The multiprogramming model used in TinyOS is not interactive enough. It offers little control over the activities (tasks) run by the sensor node, because there is no context switch. Tasks cannot be immediately interrupted or replaced in order to answer a policy established by the application, or to answer to a command issued by the user. Moreover, according to Han, Bhatti, Carlson, Dai, Deng, Rose, Sheth, Shucker, Gruenwald,

and Torgerson (2005), the execution of more complex tasks can be limited by the occurrence of deadlocks.

BSNs must be designed to operate in an autonomous manner. On the other hand, they should offer mechanisms that yield the control to healthcare personnel, since all the functionalities of the network should reflect their judgment and their clinical evaluation. Thus, a great challenge in BSN software design is to increase, in a transparent way, the access to the internal configurations of the network and its sensor nodes without damaging its capability for autonomous operation.

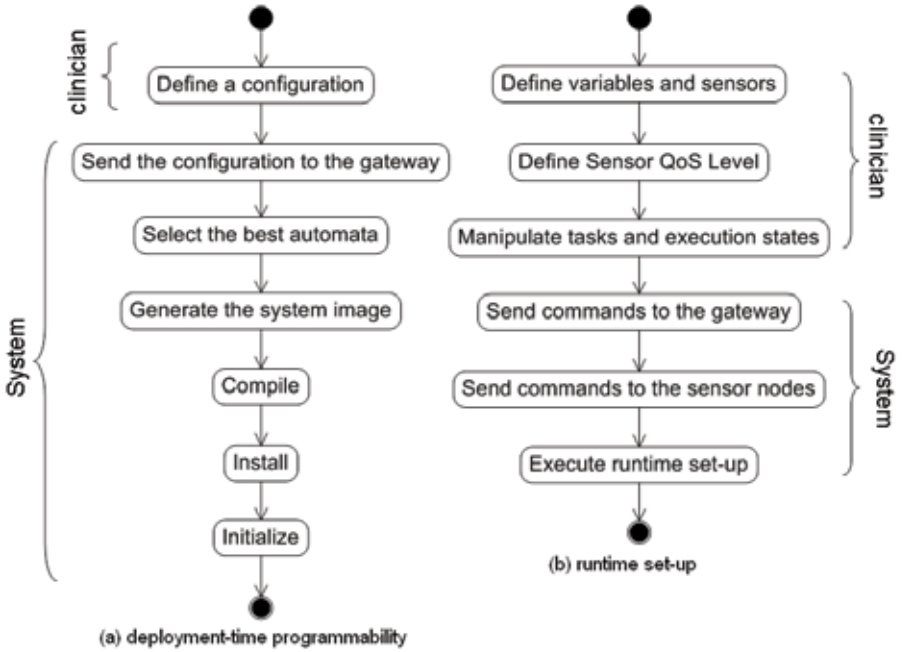
PROGRAMMING BODY SENSOR NETWORKS

Software Architecture for Body-worn Sensor Networks (SOAB) is a system that is being developed at the University of Brasilia to support BSN-based applications (Barbosa et al., 2006). SOAB incorporates the concept of programmability, presented in the introduction of this article, as functionality (service) offered by the BSN. Figure 1 presents a diagram illustrating the activities related to the method for programming BSNs proposed by SOAB. Part of these activities should be run by healthcare professionals (clinicians), and the remaining activities are run by the system itself.

Figure 1a refers to programming at deployment time. These activities are responsible for the definition of mechanisms and policies that are included in the sensor nodes to support a specific application. Figure 1b refers to the run-time set-up process, responsible for the definition and/or adjustments of applications and data sources. The detailed definition of these activities will be presented later, during the description of the layers of the proposed architecture.

SOAB is a modular system, divided into four independent functional layers: (i) a graphical interface, developed especially for healthcare personnel; (ii) middleware for interconnection of the BSN with the Internet; (iii) a server for handling the services related to the services requested by the programmers; and, finally, (iv) an operating system with support for preemptive multitasking, which will be installed in

Figure 1. Activity diagrams to describe the SOAB programming methodology



the sensor nodes. This operating system was named MedOS, and one of its main features is that it tries to increase the lifetime of the sensor nodes by means of proper scheduling of tasks, based on policies that are adapted to biomedical applications. In order to systematize these policies, an automata-based model has been used. Moreover, MedOS allows BSN managers to manipulate the states of the running tasks. Figure 2 presents an overview of SOAB.

In the first layer of SOAB (Figure 2), we have the BWSNET Configuration Tool. This graphical interface is responsible for providing means by which healthcare professionals can describe their algorithms in a way that is more intuitive, less time-consuming, and less error-prone. This interface also allows the possibility of including new sensors that were not initially specified, without the need to recompile the source code. An interface based on Java Reflection (Java Tutorials, 2006) was used to achieve this feature. The BWSNET Configuration Tool has also a simulator that acts as a virtual debugger for estimating the sensor node lifetime and other performance metrics. As a sample of the BWSNET Configuration Tool, Figure 3a shows a deployment-time programming interface for the acquisition of the electrocardiogram (ECG), while Figure 3b shows the run-time set-up interface.

Figure 3b also shows how an application is defined, and how it can be modified. As an example, the application "monitoring user stress" was defined by four variables: blood pressure, ECG, heart rate, and Blood O2. Each variable can also be defined by other variables of data sources, as described by Carvalho, Perillo, Heinzelman, and Murphy (2004). The "ECG" variable, in this case, has been defined by three data sources: ECG 1Lead 100 Hz, ECG 3Leads 500 Hz, and ECG 6Leads 1000 Hz. In order to support these functionalities, the tasks and the software artifacts have also been included at deployment-time (see Figure 3a).

In order to increase the lifetime of the network and, consequently, of the application, the system is responsible for determining how much time each data source should stay active, and how it should operate (the required precision). In order to do so, the system uses information regarding the application in order to execute policies that promote, for instance, energy saving. These policies act on the scheduling of the running tasks and on the adjustment of the bandwidth, and of the transmission power of the communication interface of the sensor nodes (Sensor QoS Level). By using the run-time set-up interface, the healthcare professional can alter the configuration that is usually chosen by the system, excluding and/or adding new data sources and variables, and can manipulate the Sensor QoS Level.

Figure 2. SOAB's four layer system

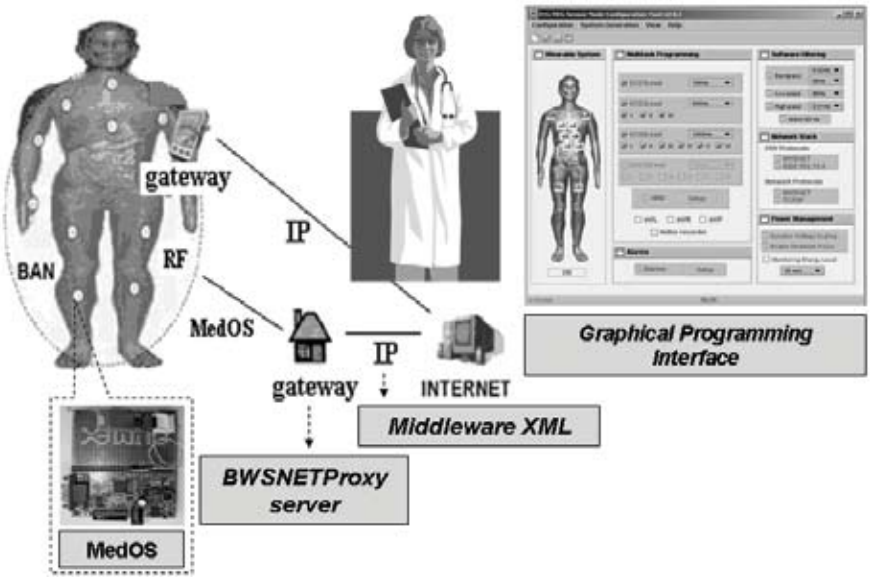
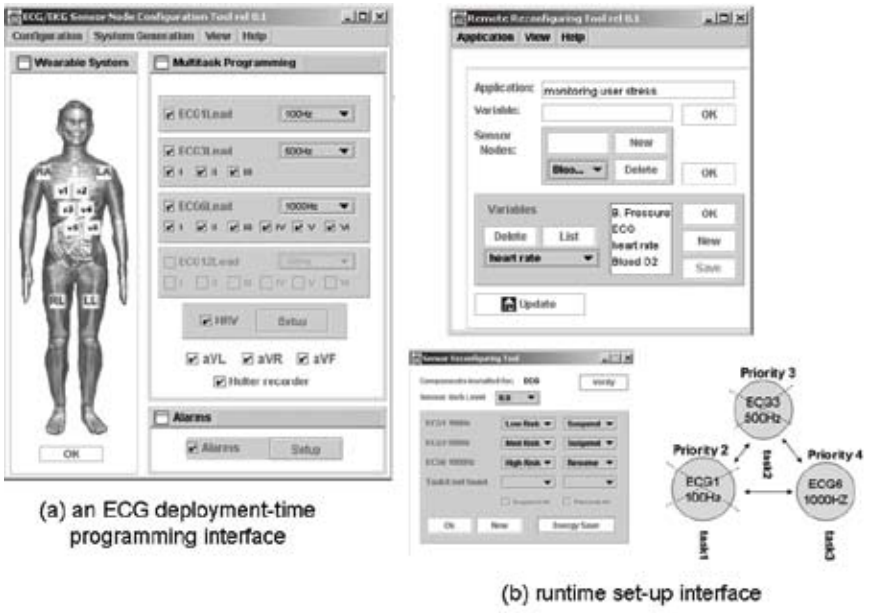


Figure 3. Examples of SOAB graphical interfaces



The user can also change the current values attributed to the priorities associated with the tasks, and also suspend and restart tasks.

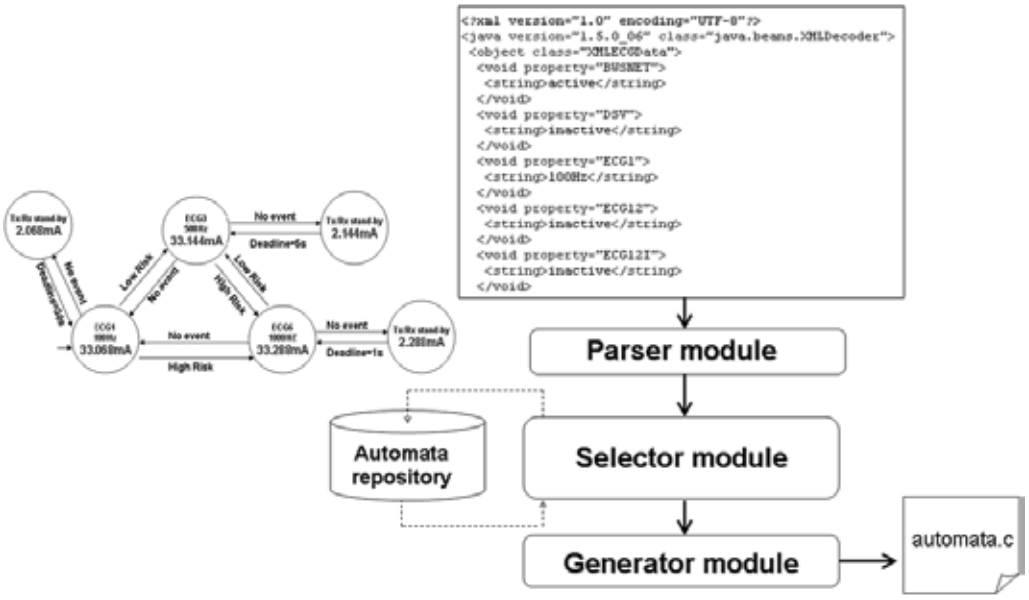
In order to provide support for the programming and reconfiguration of the sensor nodes from the Internet, a group of remote procedure calls (RPCs) has been implemented in compliance with the recommendations of the W3C Web Services (http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/; http://www.w3.org/TR/wsdl20/). These RPCs represent the XML middleware layer in Figure 2.

In the third layer, we have the *BWSNET Proxy*. This software entity has been projected to operate in the gateway devices. It is responsible for the translation of the requisitions of the SOAP XML format (http://www.w3.org/TR/2003/REC-soap12-part1-20030624/) to commands that are understandable to the underlying layer. Besides, it has the responsibility of creating the real image of the system from the BSN manager's program and the correspondent automata, the selfadjustable binary code that will be executed by the sensor nodes. Another functionality of the BWSNET Proxy server that is equally important, is that it promotes the integration of the BSN with an external healthcare information system (HIS).

Figure 4 shows the module that is responsible for selecting the best automatum for a given configuration defined at deployment time. The automata generator is an intermediate-level intelligent compiler. It is composed of three modules. The "Parser" module is responsible for extraction relevant information for the "Selector" module. In practice, these informations are related to parameters, such as the objective of the application, the number of tasks, the types of tasks, and the selected alarms, among others. The values of these parameters are used for filling a table of parameters that will be used by the selector module. The "Selector" module chooses the automatum that maximizes the potential of each application, guaranteeing a greater lifetime for the system. In practice, the selector module is a decision tree (Cormem, Leiserson, & Rivest, 1997, pp. 173) that is responsible for the choice of the best automatum from a set of possible automata that are deposited in an automata repository. This choice is made based on the table of values that is made available by the "Parser" module. The definition of this three (selection algorithm) is based on the medical knowledge that influences the values of each parameter in the table.

After the choosing the best automatum, the "Generator" module can generate the "automato.c" file (see Figure 4), which will represent the program that will be embedded into the sensor node. For this task, a description of the necessary code libraries and necessary system calls should be used. This description is stored in the automata repository, along with the functional description of each automatum. During the automata

Figure 4 . The automata generator module



selection, this configuration file is used by the "Generator" module for assembling the "automato.c" file.

In the last layer of SOAB, there is an application-oriented operating system called MedOS. The main objective of MedOS is to provide support for the behavioral adjustment of the sensor-node at run-time, changing the priority values that can be associated with the tasks provided by the sensor-node. Figure 5 shows the main artifact of MedOS and their dependencies. The functionalities are represented by tasks created together using FreeRTOS libraries (Barry, 2006).

Besides the preemptive multitasking functionality, MedOS implements a set of device drivers, a set of library functions that are used by the automata generator module, and a command interpreter. The command interpreter allows programmers to alter the scheduling of tasks that is currently adopted by the system. This functionality may or may not be included during the generation of the image of the system. If this functionality is disabled, artifacts related to the library "interpreter.h" (Figure 5) will not be included. Thus, only artifacts related to the "policy.h" file (which represent the behavior of the automatum that has been chosen by the application) will be included in the "automata.h" file, which represents the image of the system that will be installed in the sensor nodes.

In addition to the code libraries of FreeRTOS, the MSPGCC code libraries (http://mspgcc.sourceforge.net/) have also been used for the implementation of MedOS.

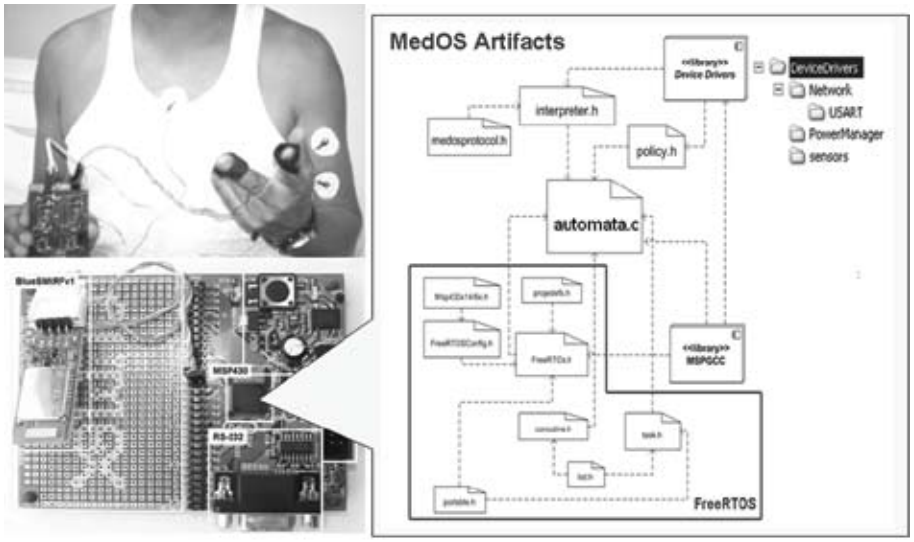
SOAB has been tested in the hardware platform presented in Figure 5. The sensor-nodes were built using the Olimex MSP-P149 kit (http://www.olimex.com/dev/msp-p149.html), with a bluetooth radio BlueSRMIFv1 (http://www.sparkfun.com/commerce/product_info.php?products_id=582). This system allows the monitoring of electrocardiographic and electromyographic signals, as well as skin temperature, arterial pressure, and galvanic skin resistance.

FUTURE TRENDS

The SOAB includes three main innovative features to the BSN research area: (i) a graphical interface for programming and reconfiguration of the sensor nodes; (ii) an application of the concept of preemptive multitasking to BSNs; and (iii) an intelligent compiler (which is also an automata generator). The authors believe that these three features will be pursued by future efforts in this area. Some other potential future trends may be highlighted:

- *The study of man-machine interfaces, and its repercussions on programming and reconfiguration models of BSNs*. A graphical user interface is

Figure 5. MedOS artifacts and sensor node architecture



an attractive and effective alternative, especially for remote access using the Internet. However, in some situations and environments, such as in an emergency room, other devices (hardware) could be used in a more effective way. Baudus et al. (2004) discuss the use of a infrared set-up pen, which works as a remote control, allowing the physicians and nurses to connect the data sources needed for patient monitoring. These data sources are implemented with sensor nodes that can be interconnected to form a BSN that is suitable for the desired application.

- *Smart compilers* that are capable of embedding, during the programming at deployment time of the sensor nodes, mechanisms and policies that increase the effectiveness of the application should also be targeted by new developments. These compilers must be based on a specialist's knowledge, since the medical expert is the one who should know how to schedule the resources according to each application, and should operate in a transparent way to the user.
- *Smart and adaptive algorithms* for task scheduling that take into account the dynamics of the applications should also be pursued. For instance, the health condition of the patient and the capability of the sensor node and of the network should be taken into account in the formulation of these algorithms. In this context, preemptive multitasking allows the computer system to guarantee for each process a regular "slice" of operating time in a more reliable way. It also allows the system to rapidly deal with important external events, such as incoming data, which might require the immediate attention of one or another process.
- *Stantardization of the software features* and of the process related to the development of sytem for BSNs will also be an important issue. In this subject, we can highlight: (i) the development of methodologies for evaluation of these systems regarding their usability and effectiveness; and (ii) the standardization of the interfaces and services as a tool for promoting the independency of hardware and for facilitating the integration of information with other networks and systems. For example, the definition of a BSN Sensor Ontology could establish a hierarchy that expresses the semantics associated to each type of sensor, group or relationship. This approach can increase the effectiveness in the use of BSNs facilitating the gathering of information, which could also be used for programming and reconfiguration of the network.

CONCLUSION

By using a modular software architecture that is designed according to the requirements of possible applications, it is possible to increase the capability

for autonomous operation of the BSN, and still offer tools for programming and reconfiguration of these systems that can be used for people with little grasp on programming languages.

This text presented a proposal for a paradigm shift for the programming and reconfiguration of the BSNs. By making more suitable tools available—such as programming interfaces, intelligent compilers, and application-oriented algorithms—the goal is to allow healthcare workers to become the actual programmers and maintainers of this technology. The possibility of developing and/or testing new applications without the need of specific technical knowledge on programming languages and computational models can facilitate the popularization of this technology.

SOAB integrates concepts such as reflection, Web services, automata, intelligent compilers, and preemptive multitasking, with the goal of becoming a solution that is functional, portable, usable, effective, and easy to maintain. Also, this architecture can be extended into four levels, allowing its improvement and the implantation of other systems (services). This solution establishes a concept: application-oriented software architecture.

REFERENCES

Asada, H., Shaltis, P., Reiner, A., Sokwoo, R., & Hutchinson, R. C. (2003, May–June). Mobile monitoring with wearable photoplethysmographic biosensors. *IEEE Engineering in Medicine and Biology Magazine, 22*(3), 28–40.

Baldus, D., Klabunde, K., & Müsch, G. (2004). Reliable set-up of medical body-sensor networks. *Lecture notes in computer science, 2920/2004,* 353–363.

Barbosa, T. M. G. de A., Sene, I. G. Jr., Carvalho, H. S., da Rocha, A. F., Nascimento, F. A. O., & Camapum, J. F. (2006). Application-oriented programming model for sensor networks embedded in the human body. In *Proceedings of the 28th Annual International Conference IEEE Engineering in Medicine and Biology Society (EMBC)* (Vol. 1, pp. 6037–6040). IEEE EMBC Press.

Barry, R. (2006). *FreeRTOS*. Retrieved February 6, 2008 from, http://www.freertos.org/

Carvalho, S. H., Perillo, M., Heinzelman, W., & Murphy, A. (2004, January–February). Middleware to support sensor network applications. *IEEE Network Magazine Special Issue*, *18*(1), 6–14.

Chakravorty, R. A. (2006). Programmable service architecture for mobile medical care. In *Proceedings of the Fourth IEEE International Conference on Pervasive Computing and Communications WorkShops (PERCOMW´06)*. IEEE Computer Society Press.

Cormem, T. H., Leiserson, C. E., & Rivest, R. L. (1997). *Algorithms* (1st ed.). Cambrige, Massachusetts: MIT Press.

Farschchi, S., Nuyujukian, H. P., Pesterev, A., Mody, I., & Judy, W. J. A. (2006, July). TinyOS-enabled MICA2-based wireless neural interface. *IEEE Transactions on Biomedical Engineering*, *53*(7), 1416–1424.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., & Culler, D. (2003). The nesC language: A holistic approach to network embedded systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)* (Vol. 38, No. 5, pp. 1–11). ACM Press.

Han, R., Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., & Torgerson, A. (2005, August). MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms. *ACM/Kluwer Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks, 10*(4), 563–579.

Hill, J. L. (2003). *System architecture for wireless sensor networks*. Unpublished doctoral dissertation, Computer Science Department, University of California, Berkeley.

Hui, J., & Culler, D. (2004). The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (Vol. 1, pp. 81–94). ACM Press.

ICL - Imperial College London. (2006). *Ubiquitous monitoring environment for wearable and implantable sensors*. Imperial College London University Web site. Retrieved February 6, 2008, from http://www.doc.ic.ac.uk/vip/ubimon/partners/index.html

Java Tutorials. (2006). *Tutorials trail: The reflection AP*. Retrieved February 6, 2008, from, http://java.sun.com/docs/books/tutorial/reflect/

Jovanov, E. (2006). *WHMS - wearable health monitoring system*. The University of Alabama in Huntsville, Electrical and Computer Engineering department Web site. Retrieved February 6, 2008, from http://www.ece.uah.edu/~jovanov/whrms/

Kara, S., Kemaloglu, S., & Kirbas, S. (2006). Low-cost compact ECG wih graphic LCD and phonocardiogram system design. *Springer Jornal of Medical Systems, 30*, 205–209.

Linz, T., Kallmayer, C., Aschenbrenner, R., & Reichl, H. (2006). Fully integrated EKG shirt based on embroidered electrical interconnections with conductive yarn and miniaturized flexible electronics. In *Proceedings of International Workshop on Wearable and Implanted Body Sensor Networks (BSN'06)* (Vol. 1, pp. 23–26). IEEE Computer Society Press.

Valdastri, P., Menciassi, A., Arena, A., Caccamo, C., & Dario, P. (2004, September). An implantable telemetry platform system for in vivo monitoring of physiological parameters. *IEEE Transactions on Information Technology in Biomedicine, 8*(3), 271–277.

Welsh, M. (2006). *CodeBlue: Wireless sensor networks for medical care.* Harvard University, Division of Engineering and Applied Sciences Web site. Retrieved February 6, 2008, from http://www.eecs.harvard.edu/~mdw/proj/codeblue/

Yang, G. (2006). *Body sensor networks*. London: Springer-Verlag.

KEY TERMS

Application-Oriented Operating System: An operating system that is designed or customized for fulfilling the interests of an application or of a set of similar applications that share a significant part of the requirements. This kind of system must include specific mechanisms and policies, in order to effectively serve the interest of the applications.

Automata or State Machines: A mathematical model that can be used for representing the behavior of a computational system. A state diagram is used for its graphical representation. Basically, an automatum is composed of states that can be graphically represented by circles and edges, which indicate the actions that are responsible by changes of state. In WSNs, usually the states are associated with the tasks, and the edges are directly related to the events that are responsible by the changes in states.

Context Awareness: The capability of computational systems of determining circumstances and scenarios in which its users may be inserted, or simply determining information that is of interest to the system. To do so, the system uses rules (intelligent algorithms) and data that can be supplied by the user itself, or obtained by sensors. These data can also lead to probabilistic information. In this case, the context awareness is not exact, but instead, a hypothesis with an associated probability value.

Multiprogramming: The approach that allows several processes to run simultaneously in a system. In a multiprogrammed sensor node, several programs are simultaneously maintained and managed in the memory by resident software. These programs are organized as tasks. The execution of these tasks follows a policy that is defined during the design of the system, and it is coordinated by a mechanism called task scheduler. In WSNs, there are two kinds of multiprogramming: (i) event-driven run-to-completion single thread approach, and (ii) preemptively time-sliced multithreading model.

Ontology: A data model that represents the entities that are defined and evaluated by its own attributes, and organized according to a hierarchy and a semantic. Ontologies are used for representing knowledge on the whole of a specific domain or on of it. In WSNs, ontologies are included in software architectures as a way of facilitating the search for relevant information. For example, in a BSN an ontology could be used for sensing a specific context (context awareess).

Pervasive Monitoring: The capability of a system to keep itself operating under any condition. These conditions can be related not only to the location (space), but also to the time. They can also be related to technical conditions, such as connectivity, safety, and the amount of energy stored in the batteries.

Preemptive Multitasking: An approach of multiprogramming. The operating system can interrupt

the execution of a task and initiate the execution of another, in order to satisfy the restrictions that are defined by the policy that is being used for the scheduling of tasks. It also allows the execution of a task to be suspended or interrupted by a command received from the application.

Quality of Service (QoS): The capability of the system of adjusting itself or of offering mechanisms that allow its adjustment in order to fulfill the requisites that are defined for each application. In WSNs the parameters that are commonly related to the quality of the service that is offered by this kind of system are application lifetime (which is related to energy consumption), connectivity, confidentiality, reliability, bandwidth, and transmission power.

Reflection: A concept related to software: programming languages, operating systems, middleware, and Graphical User Interface (GUI). A reflexive system offers mechanisms that allow its data structures to be inspected and/or modified during execution (at run-time). For this, the system must keep its selfrepresentation that is commonly organized as metadata. In nonreflexive systems, the metadata usually are lost or discarded during compilation, typically when the low-level codes (assembly language) are generated.

Web Services: Modular, independent, selfdescriptive programs that are designed to guarantee interoperability among systems that are developed with different technologies and that interact in a computer network. Typically, Web services are described by using the WSDL (Web Services Description Language), and they use SOAP (Simple Object Access Protocol) for message exchange.