# Using UPnP services with an Intelligent Sensor Network Node

RADU DOBRESCU, MATEI DOBRESCU, MAXIMILIAN NICOLAE, DAN POPESCU

POLITEHNICA University of Bucharest, ROMANIA
Splaiul Independentei 313, Faculty of Control & Computers

*Abstract:* - The goal of the equipment which is proposed for research and developed in an experimental version is to associate a typical WSN (wireless sensor networks) node architecture with the UPnP services architecture, in order use TCP/IP and WEB technologies to manage sensor networks without a specific configuration. The sensor management software architecture uses UPnP and WSN technologies and allows interactions between UPnP Control Points and wireless sensors networked in the ambient environment. Even if the sensor devices are embedded systems with limited resources, the proposed software package allows direct interactions with UPnP services.

*Key-Words:* - system prototype, middleware support, networking services.

## 1 Introduction

The wireless sensor networking is one of the most essential technologies for implementation of ubiquitous computing. The sensor nodes are usually scatted in a sensor field and data are routed back to the sink by multi-hop. These sensor networks usually share the same communication channel. Sensor nodes have limited in power, computational capacities, memory and short-range radio communication ability. The limited battery life of sensor nodes raises the efficient energy consumption as a key issue in wireless sensor networks. There are four major sources of energy waste: collision, overhearing, control packet overhead and idle listening. References as [1], [2], [3], [4], [5] discuss different procedures to optimize the power consumption. The aim of this paper is to describe the implementation of a wireless sensor with specific facilities for integration in a sensor network. The design offer a microcontroller based hardware architecture, using TinyOS operation system [6] and propose a solution for integration of this devices in UPnP Services Network [7]. Its performances: low power consumption, reduced error bit rate, high computational capacities are fair enough to recommend it for use as a node in a WSN and justify its acronym: ISNN – Intelligent Sensor Network Node. In the same time, the device proposed in this paper is part of a highly performing communication system that aims to combine message processing procedures with signal processing procedures and multi-point transmission organisation.

## 2 Architecture of the Intelligent Sensor Network Node

The specific requirements for an Intelligent Sensor Network Node (ISSN) particular hardware and software architecture can differ depending on the application:

- A large-area low-density sensor network deployment will require a more powerful radio than a short range indoor or on-the-body sensor application.
- Applications where high data rates and complex signal processing functions are required will benefit from a more powerful signal processor.
- Sensors and associated sensor electronics will vary from application to application.
- For some applications the power can be provided by an appropriate primary battery, whereas others call for a complete power management system that can scavenge energy from the environment.

### 2.1 Hardware architecture

ISNN is realized on the principle of co-design [8]. It has 6 input channels, each channel having a DAC on 10 bits. Data are processed by Atmel Atmega 128 microcontroller with 128 KB Flash memory. The data transmission/receiving is realized with Chipcon CC1000 radio module. The device has also an external flash memory of 512KB utilized to store the data when the connection with the base station is not available. The data acquisition is realized with

MTS510CA device. The ISNN module can be connected at a PC by a dedicate MIB510CA serial interface, so realizing a bridge between this PC and the sensors placed in the environment. The block scheme of the hardware structure is shown in Fig. 1.

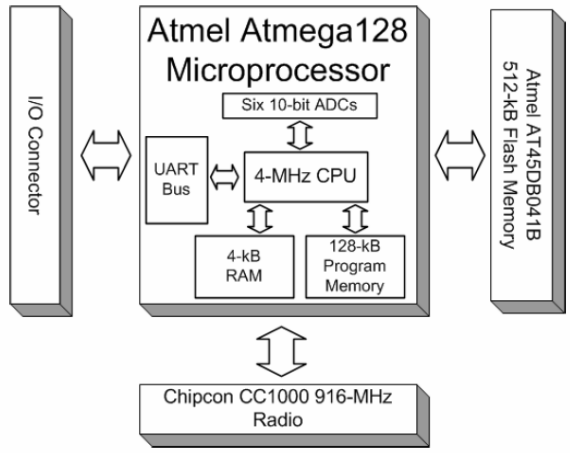


Figure1. ISSN hardware architecture block scheme

## 2.2 Software architecture

The ISNN software architecture responds to the requirements of the software framework of the whole network. The components of the framework provide the functionality of single sensors, sensor nodes, and the whole sensor network. According to these components, applications are classified into *sensor applications*, *node applications* and *network applications* [9]. The software implemented on ISNN corresponds to the first two levels (see Fig.2).

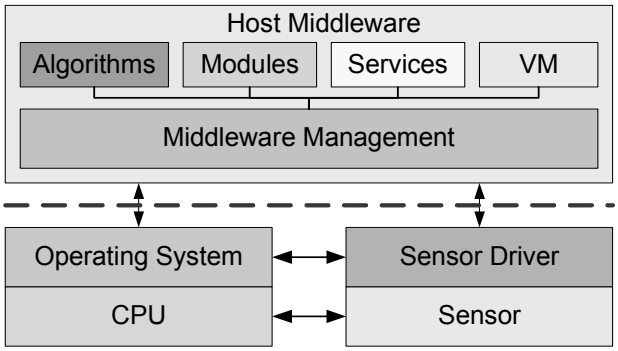


Figure2. ISSN software architecture block scheme

A *sensor application* contains the readout of a sensor as well as the local storage of data. It has full access to the hardware and is able to access the operating system directly. The *node application* contains all application specific tasks and functions of the middleware to build up and maintain the network e.g., routing, looking for nodes, discovering services, and self localization. The term *middleware* refers to the software layer between operating system and sensor application on the one hand and the distributed application which interacts over the network on the other hand [10]. Primary objective of the middleware layer is to hide the complexity of the network environment by isolating the application from protocol handling, memory management, network functionality and parallelism. A middleware for sensor networks has to be· scalable, generic, adaptive and reflective. Resource constraints (memory, processing speed, bandwidth) of available node hardware require an optimization of every node application. Thereby, the application is reduced to all essential components and data types and interfaces are customized (*scalable middleware*). The components of the middleware require a generic interface in order to minimize customization effort for other applications or nodes (*generic middleware)*. The mobility of nodes and changes of infrastructure require adaptations of the middleware at runtime depending on the sensor network application. The middleware must be able to dynamically exchange and run components (*adaptive middleware*). Reflection covers the ability of a system to understand and influence itself. A reflective system is able to present its own behaviour. Thereby, two essential mechanisms are distinguished – the inspection and the adaptation of the own behaviour (*reflective middleware).*

## 2.3 TinyOS design

In TinyOS, we have chosen an event model so that high levels of concurrency can be handled in a very small amount of space. Tiny-OS provides mechanisms (events and components) to statically define linking between layers. The predefinition of needed instances at compile time prevents from dynamical memory allocation at runtime. Tiny-OS supports the execution of multiple threads and provides a variety of additional extensions like the database TinyDB for cooperative data acquisition.

The complete system configuration consists of a tiny scheduler and a graph of components. A component has four interrelated parts: a set of command handlers, a set of event handlers, an encapsulated fixed-size frame, and a bundle of simple tasks. Tasks, commands, and handlers execute in the context of the frame and operate on its state. Commands are non-blocking requests made to lower level components. Tasks allow us to simulate concurrency within each component, since they execute asynchronously with respect to events.

## 3 UPnP Services for WSN

UPnP is a dedicated architecture for point to point interconnection of intelligent devices, wireless devices or different kind of computers. UPnP define a set of protocols that allow the access of the devices in the network without a special configuration procedure. The UPnP stack has 6 levels : (1) Device Addressing, (2) Device Discovery, (3) Device Description, (4) Action Invocation, (5) Event Messaging and (6) Presentation (see figure 3).

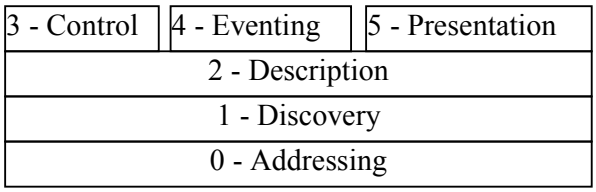


Figure3. Block scheme of the UPnP multilevel architecture

The levels 0,1 and 2 are mandatory, while the levels 3, 4 and 5 are optional.

Once a device is added in the network, its description becomes visible for all the control points, together with the available services. The information is presented in a XML document. The control points can invoke an action on the device by sending a message towards an URL contained in the device description. As one can see the UPnP network has two components. The first component is the UPnP device point having the role of a server; the second component is the UPnP Control Point which can control all the devices interconnected in the UPnP network. The following features characterize the UPnP stack levels.

Addressing. Each device must have a client DHCP (Dynamic Host Configuration Protocol), which search for a DHCP server when the device is introduced for the first time in the network. If there is no available DHCP server, the device must use Auto IP to obtain an IP address from a ser of reserved addresses.

Discovery. When an UPnP device is added in the network, the UPnP protocol allows publishing the services that it can offer to the control point. In a similar way, the UPnP protocol allows to a new added control point to search the UPnP networked devices.

Description. For each service, the description contains a list off the commands (actions) specified by the list of the arguments and their type and also the list of variables, specified by type, value domain and event characteristics.

Control. The control messages are expressed in XML using the Simple Object Access Protocol (SOAP) protocol. As a response to such a message, the service will return a value that defines the invoked action.

Eventing. Each service will publish the changes of the state variables, while the control points who have subscribed will receive the event messages contawining the variables name and their current values. These messages are expressed in XML format using the General Event Notification Architecture (GENA) protocol.

Presentation. If the device has a presentation URL, then the control point will obtain the page from this browser and according to the page capabilities, will visualize or control the status of the device.

## 4 A Case Study – Design of a Server C# Application

In This application written in C# language is a bridge between the sensors from a wireless network and the UPnP control points. The software architecture of the application, showed in figure 4, is a modular one.

The communication module ensures a serial or TCP (Serial Forwarder) sensor connection. The module is implemented in a PortConnection class and has two constructers. The first constructer is used to initialize a serial connection, the second to open a TCP/IP connection.

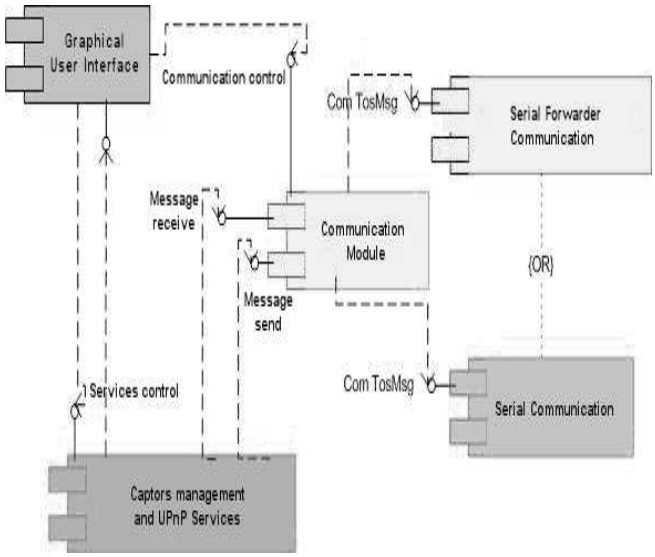


Figure 4. Software architecture of the C# application.

Figure 5 shows how to invoke a remote command on a wireless sensor. The action invoked

by a Control Point is changed in a TosMsg message send to the destination sensor. The commands *get* (to obtain a parameter value) and set (to modify a parameter value) are called from the set of defined calls, which are blocking calls.

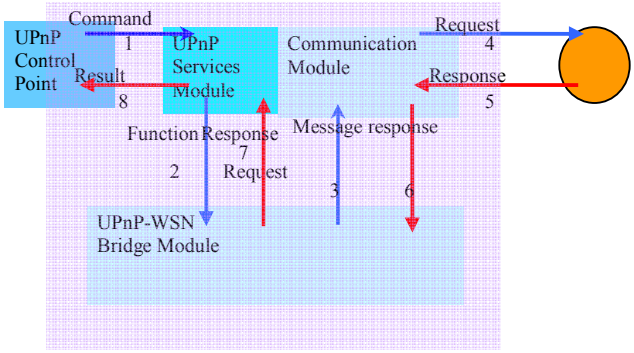


Figure 5 Interactions between sensors and UPnP Control Points.

## 5 Conclusions

The device proposed in this paper is part of a highly performing communication system that aims to combine message processing procedures with signal processing procedures and multi-point transmission organisation. The sensor management software architecture uses UPnP and WSN technologies and allows interactions between UPnP Control Points (in software applications supported by PCs, PDAs or Smart Phones) and wireless sensors networked in the ambient environment. Even if the sensor devices are embedded systems with limited resources, the proposed software package allows direct interactions with UPnP services.

The proposed ISNN architecture differs from previous work in being based explicitly on a hardware/software co-design approach supporting the deployment of novel programming language constructs directly onto the hardware in order to improve optimization. We are currently completing feasibility studies on the components of our proposed architecture, prior to initial development work. Our immediate research challenges are to determine appropriate abstractions for the construction and deployment of the embedded systems architecture from hardware and software perspectives. We intend to evaluate our work against a range of applications in order to check the qualities of individual dedicated solutions.

*ACKNOWLEDGEMENTS*

This work was partially supported by the Romanian Ministry of Education and Research under Grants No. 1467A/2005 and No. 434 TD/2006

*References:*

[1] G. Pottie. and W. Kaiser, "Wireless integrated network sensors" *Communication of the ACM*, **43**, no. 5, 2000, pp. 51-58.

[2] S. Olariu, L. Wilson, M. Eltoweissy and K. Jones, "Training a wireless sensor network", *Mobile Networks and Appl.*, 10, 2005, pp. 151-167

[3] F. Akyildiz, W. Su, Y. Sankarasubramanian and E. Cayirci, „Wireless sensor networks: A survey", *Comp. Networks*, **38** (4), 2002, pp. 393–422.

[4] T.van Dam and K.Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *SenSys*, pp.171-180, 2003

[5] J. Cortes, S. Martinez, T. Karatas and Francesco Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243-255, 2004

[6] C.L. Fok, "TinyOs tutorial", CS521, www.princeton.edu/~wolf/EECS579/imotes/tos_tutorial.pdf

[7] H. Song, D. Kim, K. Lee and J. Sung, "UPnP-based Sensor Network Management Architecture and Implementation", *Second International Conference on Mobile Computing and Ubiquitous Networking*, 2005, Osaka

[8] A. Nisbet and S. Dobson, "A systems architecture for sensor networks based on hardware/software co-design", In Mikhail Smirnov, editor, *Proceedings of the 1st IFIP Workshop on Autonomic Communications*, Springer Verlag, 2004.

[9] R. Dobrescu, M. Nicolae, F. Stoica and R. Varbanescu - Design of an Intelligent Sensor Network Node, Proceedings of the*10th International Conference on Optimization of Electrical and Electronic Equipments OPTIM'06*,Brasov, 2006, p. 71-78

[10] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks", *Information Processing in Sensor Networks*, 2005, pp. 246 – 253