Embedding wireless sensors in UPnP services networks

Radu Dobrescu, Matei Dobrescu, Maximilian Nicolae and Dan Popescu

***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..

***Keywords*—**About embedded architecture, middleware support, service discovery, UPnP, WSN.

I. INTRODUCTION

THE document wireless sensor networking is one of the most essential technologies about acquiring and processing information. The recently research of wireless sensor networks touch upon the technology of service discovery. Service discovery protocols enable service providers to advertise capabilities to potential clients, while also providing to clients and service providers a means for entering into relationship. Service Location Protocol (SLP), Bluetooth's Service Discovery Protocol (SDP) or Universal Plug and Play (UPnP) are protocols addressing service discovery [1]. This paper discusses two main aspects regarding the wireless sensor networking technology: first, the implementation of a wireless with embedded architecture using TinyOS operation system [2] that offers specific facilities for integration in a sensor network as an Intelligent Sensor Network Node and second, the analysis of the mechanisms for service and resource discovery, using UPnP protocol. UPnP developed by Microsoft [3], is designed to bring easy-to-use standards-based connectivity to ad-hoc or unmanaged networks whether at home, in a small business, public spaces, or attached to the Internet. UPnP supports invisible networking and automatic discovery for a breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, advertise its capabilities and learn about the presence and capabilities of other devices. UPnP technology enables data communication between any two devices under the command of any control device on the network and aims to provide secure connections. So, 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. It can represent the best solution for closely coupling enterprise systems and tiny wireless sensor nodes that are embedded into physical items.

Manuscript received October 9, 2006. This work was partially supported by the Romanian Ministry of Education and Research under Grants No. 72CEEX/2006 and No. 11040PC/2007 . Revised received May 3, 2007

The authors are with the POLITEHNICA University of Bucharest, Faculty of Control and Computers, Splaiul Independentei 313, Bucharest 7100, Romania (telephone: +40214029105, e-mail: radud@isis.pub.ro).

II. ARCHITECTURE OF THE INTELLIGENT SENSOR NETWORK NODE

A. Requirements

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.

B. Hardware architecture

ISNN is realized on the principle of co-design [4]. 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.

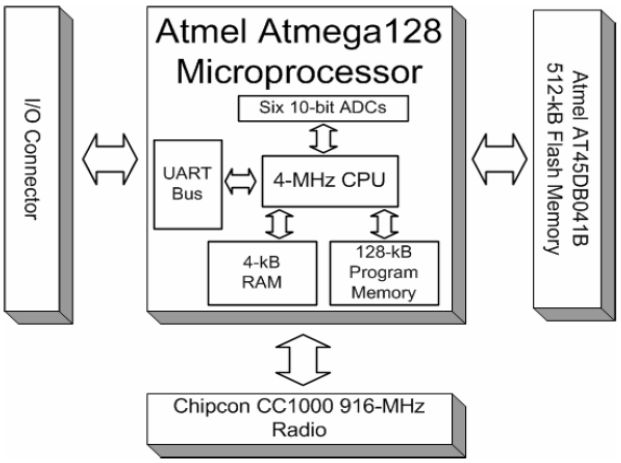

Fig.1. ISSN hardware architecture block scheme

C. 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* [5]. The software implemented on ISNN corresponds to the first two levels (see Fig.2).

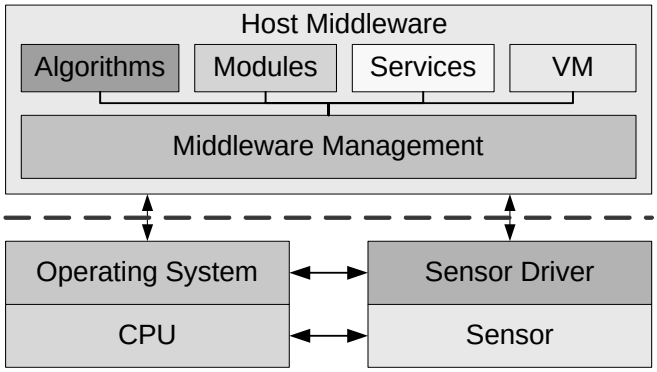

Fig.2. ISSN software architecture block scheme

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 [6]. 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)*.

D. 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 [7]. 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.

E. Dedicated UPnP-Based Sensor Network Management Architecture

In UPnP technology, the control point is the entity in the network that works with the services provided by UPnP devices [8]. In our approach, the control point is a device to control and manage a sensor network using the services provided by an ISSN, which can be considered also an UPnP device that has enough powerful resources to compute UPnP protocol. This base node uses bridge architecture for communication between the sensor network and control point. Finally, each sensor node is a non-UPnP device which has sensing capabilities. If the current sensor device does not have

sufficient resources to be able to process the UPnP protocol, the control point and ISSN communicate with each other using UPnP protocol. On the other hand, non-UPnP sensor devices and ISSN use sensor network specific protocol for communication. Consequently, ISSN must have the ability to know the list of services of the sensors through communicating with sensor nodes using sensor network proprietary protocol. The Service Manager has the following work flow: if a new ordinary (non-UPnP) sensor is added in the sensor network, it sends a service advertisement message to the service table manager. The service table manager adds the entry into the service table with a received message from the non-UPnP sensor devices. At this time, the control point receives an event notification message about the device addition through UPnP eventing. The control point can call the *GetSensorDescription()* action, one of the discovery services provided by ISSN, in order to get the description message based on XML related with the sensors. In addition, when a control point is added to the networks, the control point can request the description document for the nodes in the sensor network in which it has participated using the discovery service.

III. UPnP Services for WSN

A. Service discovery in wireless sensor networks

Please The wireless sensor networks have several characteristics that set them apart from the classical wired and wireless networks. These differences make the large body of research in Service Discovery (SD) protocols, not directly applicable to WSN. One of the basic properties of the wireless sensor networks is their application specific nature. The nodes in a WSN cooperate together in order to jointly solve one or more clearly defined tasks or applications. This has a significant impact on the SD functionality that traditionally has to create a binding between the requested service/resource description and the Identification (ID) of the nodes in the network that provides such a service[9]. The semantic routing protocols that operate without globally unique IDs make such a direct binding unnecessary. Here the main task of the SD component would be to support the reconfiguration capability of the network and to build repositories of the available semantic attributes. These repositories will facilitate the cooperation between the nodes in the network and will support the interaction with the human operator and the interfacing with the external networks. Of course, there are some particular tasks in a WSN that require a centralized ID communication. Regardless of the ID-centric nature, the SD implementation has to be streamlined and should contribute to the overall energy-efficiency of the protocol stack. The number of generated messages is the main factor determining the scalability of the protocol. Thus, every attempt should be made to reduce it. The tight integration between the routing and the SD functionality has been suggested as a promising approach to this end. The sharing of the available state information and the lowered control overhead can significantly increase the scalability of the implementation making it appropriate for the resource constrained WSN setting.

B. Features of the UPnP architecture

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 table 1).

TABLE I
THE UPNP MULTILEVEL ARCHITECTURE

3 - Control	4 - Eventing	5 - Presentation
2 - Description		
1 - Discovery		
0 - Addressing		

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 set 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, it 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, 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.

C. Implementing services from an enterprise system on the wireless sensor node

In the last years, we have seen an increased engagement to electronically support industrial processes that involve physical items. In many cases it is necessary to relocate process tasks from the back-end enterprise system (BES) onto the items. The execution of these tasks is delegated to physically embedded systems (PES), implemented as tiny, embedded devices, which are directly attached to the items. The following benefits can be obtained [10]: *scaling*, because the relocation of process tasks will unload the BES from processing data streams coming from each single item; *real-time activities* (for example, alerts can be raised locally at the point of incidence where immediate action is required; increased degree of freedom, because the process of relocation is designed, initiated and controlled by the BES. With wireless sensor network platforms, sufficient PESs are already available.

Figure 3 outlines a service-based scenario in WSN. We assume the software running on each network node is organized as a set of services. Middleware components create so called service proxies, which are run-time representations of the embedded services that the BES can access. Three possible ways of using data from service proxies are shown: real-time data to support the execution of one unique process task (U1), parts of a process that can be relocated to the PES (U2) or a choice of which process task to execute next if more than one option is available (U3).

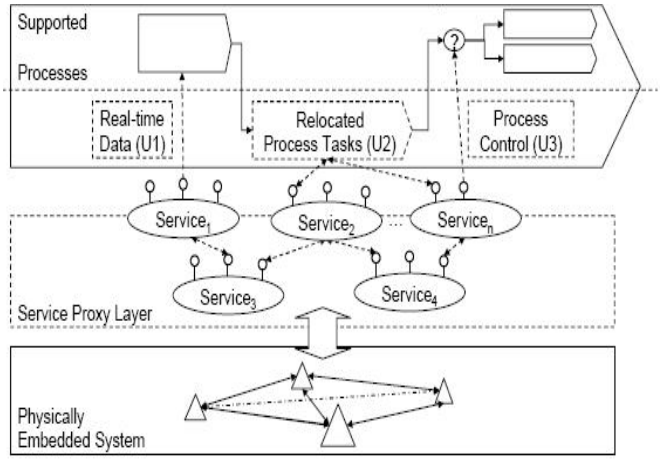

Fig. 3. Coupling BES with PES using relocated process tasks

Service proxies on platform gateways [11] couple PES and BES utilizing UPnP interfaces. UPnP is standardized and is specifically designed to enable a service oriented architecture for embedded systems. The UPnP model of state variables supports the communication with the sensor nodes by generic primitives like state queries. Events from the sensor network, e.g. an alert due to a hazardous situation, can be directly supported by UPnP's GENA event system through subscriptions and notification on the state variables.

IV. A Case Study

A. Design of a Server C# Application

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.

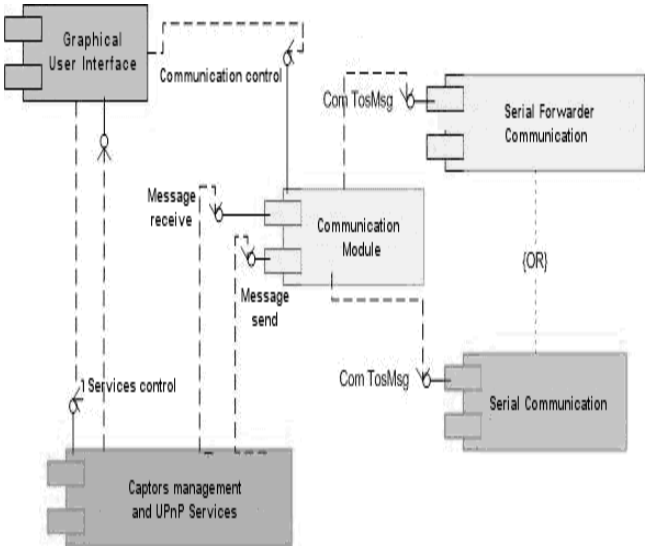

Fig. 4. Software architecture of the C# application.

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

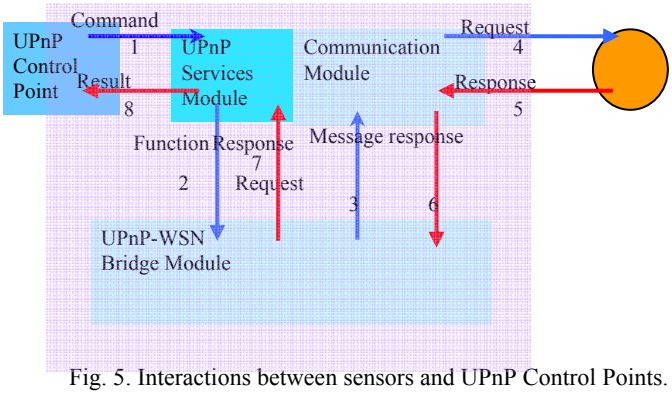

Fig. 5. Interactions between sensors and UPnP Control Points.

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.

B. Design of algorithms for service discovery and binding

Discovery protocols are needed to operate in an environment where no static configuration at any level exists. We propose a full protocol stack addressing (PSA) architecture related to the construction and maintenance of an ad-hoc network specifically formed to access a named service. Each node establishes its local service directory dynamically by collecting information from its data link neighbors. No intermediary centralized lookup services are needed. The model consists of a set of mobiles acting either as client or server, and a set of valid services on which clients can make requests. PSA architecture is a protocol stack operating at three interacting layers to accommodate access to general purpose network in a highly mobile dynamic ad-hoc environment. The lifetime of the ad-hoc network is the duration of the client-server interaction. The network is terminated after the service is used. Mobile nodes dynamically establish routes among themselves to form a multihop network. Service binding is delayed until the service request is made. The protocol stack supports network layer addressing and routing as well as service lookup and binding using descriptive names with the assumption that no static component at any level exist. PSA is composed of the following layers:

- Data link layer; a standard wireless data link such as 802.11.
- PSA layer; designed above a standard data link to establish and maintain session networks between client and server nodes. Three interacting software components are operating together in this layer. *Session Manager (SM)* provides a descriptive naming interface to the application; *Service Agent (SA)* handles service related issues. It deals with service tracking and delayed binding. *Routing Agent (RA)* routes packets to an appropriate next hop using a route maintenance and optimization algorithm embedded in the service description model. All PSA packets are forwarded to the *Communication Agent (CA)* which passes them to the wireless link interface.
- Application Layer; mobile client and server applications developed using PSA Interface for named services.

Two types of services are defined by the architecture: *non interactive services* which only require best effort delivery from the network for the client data to be delivered to the node running the service and *interactive services*, which require reliable delivery of the packets and therefore a retransmission scheme is needed. In the architecture, services are defined and represented as eXtensible Markup Language (*XML*) instances that conform to an *XML Schema*, designed to specify structure and content of the instances. XML provides a flexible, easy to parse, structured text-only format to access data efficiently.

V. 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. Future research addresses in particular appropriate error handling and resolving strategies within the real time applications for critical technical problems occurring in the embedded sensor network. Finally, our goal is to build efficient tools for monitoring, management, and support for the whole service lifecycle from modeling and deployment to termination and removal.

REFERENCES

[1] X-l. Zhou and M. Wu, "Service Discovery Protocol in Wireless Sensor Networks", in *Proceedings of the Second International Conference on Semantics, Knowledge and Grid* ,p.101-105, 2006

[2] C.L. Fok, "TinyOs tutorial", CS521, Available: www.princeton.edu/~wolf/EECS579/tutorial.pdf

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

[4] 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.

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

[6] 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

[7] S.Madden, M.Franklin, J.Hellerstein and W.Hong, "TinyDB: An Acqusitional Query Processing System for Sensor Networks", *ACM TODS*, 2005

[8] Y. Gsottberger, X. Shi, G. Stromberg, T. Sturm and W. Weber, "Embedding Low-Cost Wireless Sensors into Universal Plug and Play

Environments", In *Lecture Notes in Computer Sciences*, Springer Verlag, 2004, p.291–306.

[9] S. Baek; E. Choi and J. Huh, **"**Sensing information managing mechanism for context aware service in ubiquitous home network", *Consumer Electronics*, 2006, p. 51 - 52

[10] C. Decker, P. Spiess, L. Moreira, M. Beigl and Z. Nochta: "Coupling Enterprise Systems with Wireless Sensor Nodes: Analysis, Implementation, Experiences and Guidelines", *Pervasive Technology Applied*, 2006, Dublin

[11] M. Kuorilehto, M. Hannikainen and T. Hamalainen, "A Survey of Application Distribution in Wireless Sensor Networks", *EURASIP Journal on Wireless Communications and Networking*, 2005, p. 774-788.

Radu N. Dobrescu. Born in 11.01.1946, in Braila, Romania. Dipl.Eng. degree in Automatic Control from the Faculty of Control and Computers of the Polytehnical Institute of Bucharest, in 1968. Ph.D. degree in Electrical Engineering from the Polytehnical Institute of Bucharest, Romania, in 1976.

Currently Professor in the Department of Automation and Industrial Informatics of the Faculty of Control and Computers, POLITECHNICA University of Bucharest, head of the laboratory on Data Transmission and Industrial Communication. From 1992 Ph.D. adviser in the field of Automatic Control. Several scientific works in three main domains: Modern structures for the numerical control of machine tools and manufacturing flexible systems; Data acquisition, processing and transmission; Local area networks and industrial communication.

Prof. Dobrescu is a pioneer in fractal theory applications for medical image processing and biological systems modelling and simulation. Organiser of the three International Symposiums on Interdisciplinary Approaches in Fractal Applications (IAFA 2003, IAFA 2005 and IAFA 2007). IEEE Member since 1991, IEEE Senior Member since 2005.

Matei R.N. Dobrescu. Born in 27.11.1976 in Bucharest, Romania. Dipl.Eng. degree in Automatic Control from the Faculty of Control and Computers of the Polytehnical Institute of Bucharest, in 2000. Ph.D. degree in Systems Engineering from the "Politehnica" University of Bucharest, in 2004.

Currently Researcher in the Department of Automation and Industrial Informatics of the Faculty of Control and Computers, POLITEHNICA University of Bucharest, engaged in a post-doctoral research stage. Several scientific works, especially multimedia applications with parallel and distributed processing:

Dr. Dobrescu has received in 2005 the Excellence Award Werner von Siemens for the best doctoral thesis of the year in the field of IT&C. IEEE Member since 2000.

Maximilian C. Nicolae. Born in 16.10.1980 in Bucharest, Romania. Dipl.Eng. degree in Automatic Control, in 2004 and Master degree in Open Systems Architecture, in 2005, both from the Faculty of Control and Computers of the POLITEHNICA University of Bucharest. Ph.D. student in Systems Engineering from 2006 at the "Politehnica" University of Bucharest.

Currently Asisstant Professor in the Department of Automation and Industrial Informatics of the Faculty of Control and Computers, POLITEHNICA University of Bucharest, specialized in data transmission, wireless communication and sensor networks.

Mr. Nicolae has obtained in 2007 a research grant for two years in order to finish the Ph.D. thesis on Mobile Wireless Sensor Networks.

Dan Popescu. Born in 18.07.1950, in Bucharest, Romania. Dipl.Eng. degree in Automatic Control from the Faculty of Control and Computers of the Polytehnical Institute of Bucharest, in 1974. Licence degree in Mathematics of the University of Bucharest, in 1980. Ph.D. degree in Control Engineering from the Polytehnical Institute of Bucharest, Romania, in 1986.

Currently Professor in the Department of Automation and Industrial Informatics of the Faculty of Control and Computers, POLITEHNICA University of Bucharest, head of the laboratory of Techniques for Experimental Data Processing. The scientific research activity is concretized on the following domains: acquisition and processing of signals of image type, signals acquisition from sensors and experimental data processing..

Prof. Popescu published 14 books (8 of them as unique author) and more than 100 papers in several specialty journals or in volumes from international conferences.