On-Demand Customizable Wireless Sensor Network

Article *in* Procedia Computer Science · December 2015

DOI: 10.1016/j.procs.2015.05.089

6 authors, including:

Toshiaki Miyazaki
The University of Aizu
126 PUBLICATIONS **426** CITATIONS
SEE PROFILE

Song Guo
The Hong Kong Polytechnic University
316 PUBLICATIONS **2,135** CITATIONS
SEE PROFILE

Takafumi Hayashi
Niigata University
122 PUBLICATIONS **968** CITATIONS
SEE PROFILE

Tsuneo Tsukahara
The University of Aizu
82 PUBLICATIONS **706** CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 302 – 309

Procedia Computer Science

The 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015)

On-Demand Customizable Wireless Sensor Network

Toshiaki Miyazaki[a,]*, Peng Li[a], Song Guo[a], Junji Kitamichi[a], Takafumi Hayashi[a], Tsuneo Tsukahara[a]

[a]*The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan*

Abstract

In this paper, we propose a wireless sensor network (WSN) whose behavior can be dynamically customized by injecting programs or roles specified by the user. To enable easy specification of the roles, a role-generation mechanism is also proposed. To realize the WSN, we introduce a reconfigurable wireless sensor node that has an ultra-low-power field-programmable gate array (FPGA) as well as a low-power microcontroller unit (MCU). By injecting several different roles into the sensor nodes, we confirmed that the behavior of the WSN can be changed on demand.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Customization; Reconfigurable sensor node; FPGA

1. Introduction

Wireless sensor network (WSN) technology has become popular for the monitoring of environmental phenomena, and it has become a key technology for ubiquitous living. However, almost all WSNs that are currently available focus on specific sensing missions. Thus, even to monitor applications in the same field, if we want to detect different environmental phenomena, additional WSNs must be deployed. This introduces redundancy and is wasteful. One idea to solve this problem is to deploy into the target field a customizable WSN that can change its behavior on demand based on users' requests and environmental situations. To realize WSN customization, some interesting works[1,2,3] have been proposed. Frank and Romer[1] proposed an idea to customize the WSN behavior by injecting into the sensor nodes different roles that are specified by the user. However, their "role description" is limited, and has

* Corresponding author. Tel.: +81-242-37-2572; fax: +81-242-37-2598.
E-mail address: miyazaki@u-aizu.ac.jp

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.089

not advanced beyond evaluations that are based on simulations, leading to doubts about its feasibility. In[2], the authors propose a dynamic reconfigurable sensor network by providing a virtual machine in each sensor node that provides a real-time evaluation of tasks that are received from the outside of the sensor node. It provides a task-specification language that is based on a famous formal specification method named Communication Sequence Processes (CSP), which was proposed by C. Hoare. Thus, the validity of the sensor node behavior, which is defined using the specification language, can be checked formally. This is very useful to verify the task dependencies among the sensor nodes. However, it is still difficult to check real-time conditions such as the sensor's data sampling rate even if the mechanism is introduced. Reference[3] also proposes a reconfigurable WSN, but its reconfigurability is limited and only some parameters, e.g., the sensitivity of accelerometers, can be changed remotely to enable the monitoring of the health of structures such as bridges.

We believe that the following features are mandatory to specify a practical sensor node behavior.

(1) Condition recognition: Self-condition and neighboring condition,
(2) Sensor control: Switching on/off, and setting the sampling rate, and
(3) Data handling: Signal processing, data aggregation, packet transfer, etc.

To the best of our knowledge, there has been no related work that considers all of the above-mentioned features. In this paper, we introduce an on-demand customizable WSN (ODCWSN) whose behavior can be changed dynamically by injecting a program or "role" into each sensor node. In addition, a C-like specification language is provided to specify the role, and it considers all three features.

The sensor node is key to realizing the ODCWSN. Many wireless sensor nodes have microcontroller units (MCUs), and their behavior can be changed by downloading new programs to the sensor nodes[8, 9]. However, to realize more flexible dynamic reconfiguration and high-performance processing, some original sensor nodes have been proposed[4, 5]. These nodes use field-programmable gate arrays (FPGAs) to execute computationally intensive tasks such as sensor data processing, which cannot be done with simple MCUs in the sensor node. Reference[4] showed a method to reconfigure the remote hardware in a sensor node. However, the Xilinx Spartan-3 FPGA that was used consumes much more electric power compared to the MCU in the sensor node. Thus, their implementation is not practical for battery-powered sensor nodes. Reference[5] successfully realized a power-efficient signal processing method using a non-volatile low-power FPGA, but the proposed approach does not enable the remote reconfiguration of the FPGA. In this paper, we propose a hardware/software (HW/SW) remotely-reconfigurable sensor node to prove the feasibility of the ODCWSN.

The main contributions of this paper are summarized as follows:

- To present the three mandatory features that specify the sensor node behavior,
- To present the concept of the ODCWSN, and
- To realize a system to prove the feasibility of the ODCWSN concept.

The remainder of this paper is organized as follows. The ODCWSN is overviewed in Section 2. In Section 3, the HW/SE reconfigurable wireless sensor node is described. Then, some experimental results are shown in Section 4. Finally, we conclude this paper in Section 5.

2. On-demand customizable wireless sensor network architecture

Fig 1 shows a schematic overview of the proposed ODCWSN system. In addition to the main body of the WSN system, a role-generation mechanism is proposed. In the role-generation mechanism, a scenario compiler generates roles based on a scenario description that specifies the functions or behaviors to be activated in various sensor nodes; the roles are then downloaded to the appropriate sensor nodes via a wireless link. By changing the scenario description and delivering the generated roles to the appropriate sensor nodes via the wireless link, the behavior of sensor nodes can be customized even after the nodes have been deployed. To generate specific tasks, a scenario compiler is used to refer to a function library comprising basic functions such as sensed data transmissions, data relays, and data collection. These functions can be invoked as roles and performed by individual sensor nodes. In

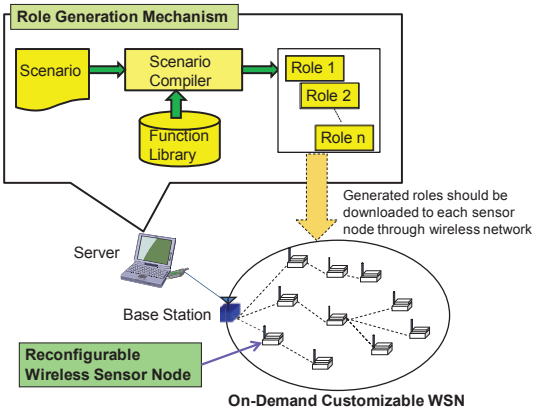

Fig. 1. An overview of the on-demand customizable wireless sensor network system.

order to download the generated roles to the appropriate sensor nodes via the wireless link, an over-the-air programming (OTAP) protocol based on the attributes of the sensor nodes (instead of the sensor node IDs) is used. The overall role generation and delivery mechanism is implemented by a server PC connected to a base station (BS) through either a wireless or wired network.

The ODCWSN consists of many reconfigurable sensor nodes, a BS, and a user terminal. Sensed data obtained by each sensor node are forwarded to the user terminal via the BS. The WSN supports a multi-hop wireless communication infrastructure. Each sensor node has a mechanism to know neighboring sensor node condition as well as its own one easily. Thus, we can change the sensor node behavior dynamically using the condition information. Because it is difficult to incorporate such capability into existing commercially available sensor nodes, we developed an original sensor node to implement this concept. In order to handle sensed data and communicate effectively using wireless transmissions with a low-power MCU, this novel sensor node requires both hardware and software reconfigurability.

2.1. Property Table

To enable each sensor node to know its own condition and the conditions of neighboring sensor nodes, it has a "property table" shown in Fig 2. The property table contains information related to itself and to neighboring sensor nodes, such as the remaining battery power, the equipped sensors and their on/off status, and the hop-count between itself and corresponding sensor nodes. The contents of the property table are periodically updated by exchanging information with other sensor nodes using the wireless link. Thus, even without the need to sense query packets, a

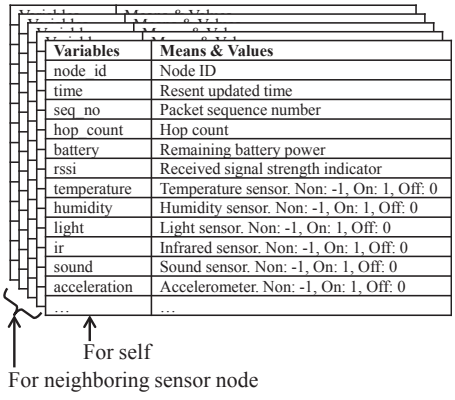

Fig. 2. The property table.

sensor node can easily know the conditions of its neighboring sensor nodes. This enables it to know the environmental status, and to assume an appropriate role without the risk of packet collisions caused by the transmission of query packets, as opposed to an on-demand-based condition query mechanism.

2.2. The Role-Generation Mechanism

Using the "coverage problem" as an example, we now explain the role generation and delivery mechanisms. The coverage problem involves finding the optimum distribution of sensing roles to cover a monitored area using a minimal number of sensor nodes. Fig 3 shows the description of a scenario used to solve the coverage problem. The scenario description language has C-like syntax. The scenario description is actually translated to C description internally by the scenario complier, and the roles running on the target senor nodes are generated from the internally-generated C description. In the scenario, it is assumed that each sensor node has both a temperature and an infrared (IR) sensor, and we can articulate role assignments that activate either of the sensors in a specific sensor node depending on the states of the neighboring nodes as follows.

Temperature sensing must commence if the sensor node has a sufficiently high battery level (greater than the threshold), and no node sensing temperature within the area representing the "sensing_range" is present within a one-hop communication distance. Otherwise, the IR sensor must be activated instead of the temperature sensor. In this example, the sampling rates of the temperature and IR sensors are preset to 0.1 (i.e., 10 Hz) and 0.2 (i.e., 5 Hz), respectively (see lines 15 and 19, respectively, in Fig 3). The packet sending timing and the method for packetizing sensed data are defined using a function that is invoked periodically by means of interval timers. The "signal()" function is used to set up the interval timer and to bind the function to the timer (lines 16 and 20, respectively), while "send_temp()" and "send_ir()" are interrupt functions that are utilized for sending temperature and IR sensor data, respectively. The "send_temp()" function directs the transmission of an averaged temperature reading every second, and "send_ir()" causes sampled raw IR sensor data to be sent out from the node to base station BS1 every 3 s. Unlike the method proposed in[1], which cannot define a detailed scenario that includes sampling and packet sending rates, our proposed environment can. It considers the three features described in Section 1, and requires that we specify the detailed control scheme of the sensor nodes in order to properly manage a real WSN. Note that the scenario description can be simplified as in[1] by providing some macro or library functions in which parameters such

```
1   main{  // main scenario
2     wp = 0; // wireless communication port id
3     temp = 0; // temperature sensor id
4     ir= 0; // IR sensor id
5     comlist = getCom(); // get communication port
6     foreach com comlist{
7       If(com.kind() == "Wireless"){
8         wp = com; // assign com as the wireless comm. port
9       }
10    }
11    stlist = getSensor(); // get sensor names and kinds
12    foreach st stlist{
13      if(st.kind() == "Temperature"){
14        temp = st; // assign a temperature sensor to temp
// set sampling_rate = 0.1 s to temperature sensor
15        setProperty(temp,sampling_rate,0.1);
16        signal(send_temp,1); // call the packet sending function every 1 s
17      }else if(st.kind()=="IR"){
18        ir = st; //assign an IR sensor to ir
// set sampling_rate = 0.2 s to IR sensor
19        setProperty(ir,sampling_rate,0.2);
20        signal(send_ir,3); // call the packet sending function every 3 s
21      }
22    }
23    if(ir != 0 && temp != 0){ // this node has both temp and IR sensor
24      if(getProperty("Battery") > threshold){ // enough battery level
// list up 1-hop neighbor nodes activating temperature sensing
25        snlist = listUp(1,"Tsens");
26        i = 0;
27        foreach sn snlist {
28          if(dist(my.Position, sn.Position) < sensing_range){ // check distance
29            i++;
30          }
31        }
32        if(i<1){ // if no 1-hop sensor node sensing temperature
33          setProperty(temp,state,"ON"); // Temperature sensing ON
34          setProperty(ir,state,"OFF"); // IR sensing OFF
35        }else{
36          setProperty(ir,state,"ON"); // IR sensing ON
37          setProperty(temp,state,"OFF"); // Temperature sensing OFF
38        }
39      }
40    }
41  } // end main scenario

// interrupt function for sending temperature data.
// an averaged temperature value will be sent to base station BS1
42    send_temp(){
43      value = average(getData(temp));
44      send("BS1",wp, value);
45    }

// interrupt function for sending IR data.
// a raw IR value will be sent to base station BS1
46    send_ir(){
47      value = raw(getData(ir));
48      send("BS1",wp, value);
49    }
```

Fig. 3. Scenario description for solving the coverage problem.

as the sensor data's sampling rate are pre-fixed, if desired by the user.

In addition to the "signal()" function, functions containing bold characters in their names, such as "getSensor()" and "listUp(),"are provided in the function library, and they are linked to roles that are generated by the scenario compiler that can be downloaded to target sensor nodes using an OTAP protocol. In addition to node IDs, the OTAP protocol used here supports attribute-based target specification, and uses broadcasting to deliver roles to multiple sensor nodes. Based on this capability, effective role deliberation can be performed regardless of the number of sensor nodes.

3. The hardware/software reconfigurable wireless sensor node

Fig 4 shows our sensor node architecture. It comprises a main board, a wireless module, and sensors. The main board has a main MCU and an FPGA. More detailed information of the main components used to realize the proposed sensor node is as follows:

MCU： TI MSP430 was selected as the MCU because in our sensor mode, sensor control and sensed data handling

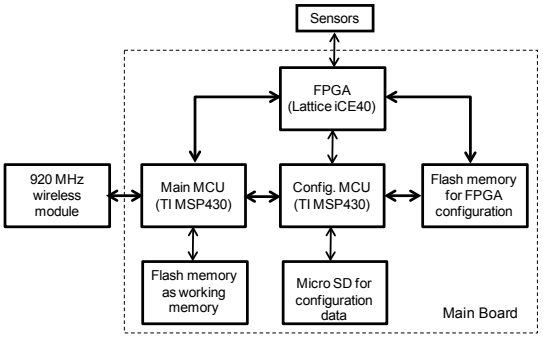

Fig. 4. A block diagram of HW/SW reconfigurable wireless sensor node.

Fig. 5. The HW/SW reconfigurable wireless sensor node.

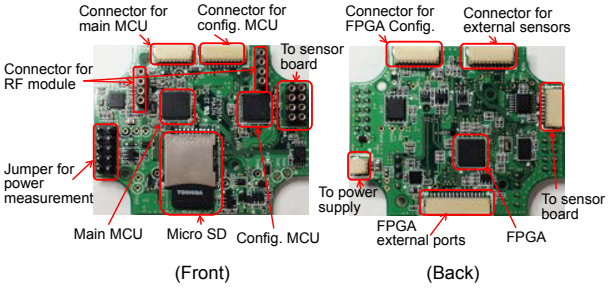

Fig. 6. The main board of the HW/SW reconfigurable wireless sensor node.

must be performed in the FPGA. Therefore, the DMA controller (not the controller in Atmel AVR, which is popular and is used in IRIS MOTE[6]) must be able to transfer the sensed data from the FPGA to local memory, as is possible with the MSP430 DMA controller. An additional TI MSP430 was also used as the configuration MCU.

FPGA: Lattice iCE40LP1K FPGA was selected, which is designed for battery-driven embedded systems such as smartphones. The typical current consumption of the FPGA is only 100 μA, which is less than that of the MCU.

RF Module： We selected AR’S Co. Ltd. “Markhor,” which uses a TI CC1101 RF chip, as the RF module. Markhor uses the 920 MHz band, which has a data transfer speed of 250 kbps, and consumption currents of 30 and 20 mA for transmission and reception, respectively. Unlike the 2.4 GHz ISM band used by Atmel RF230, which is the RF chip in the IRIS MOTE, the 920 MHz band is not yet popular, which allows for the avoidance of packet collisions with other radio systems.

Sensors: We introduced a number of sensor types, including a temperature/humidity sensor, an IR sensor, a sound sensor, a light sensor, and a 3-axis accelerometer. In addition, we added three light-emitting diodes (LEDs) and a buzzer as actuators. All of the sensors and actuators were installed on a sensor board and connected to the FPGA through either an analog-to-digital converter (ADC) or a digital-to-analog converter (DAC).

Moreover, a micro SD card and two flash memories were introduced to temporarily store the configuration and other data.

The power supply is a 5 V 850-mAh lithium-ion rechargeable battery. With this battery, the aforementioned hardware can run for 19 hours with the sensor node sending a packet every second.

Fig 5 is a snapshot of our HW/SW reconfigurable sensor node. All components, including an antenna and the battery, are packed into a small plastic case. Fig 6 is the main board. All of the main parts are mounted on both sides of an originally designed printed circuit board (PCB). Besides the main board, the sensors and actuators are mounted on another PCB, and attached to the plastic case. Two PCBs and the RF module are connected to each other using connectors and flat cables.

3.1. The FPGA Roles

The FPGA can perform several roles to reduce the load of the MCU. Typical uses of the FPGA are as follows:

(1) Co-processing: The goal of co-processing is to offload heavy tasks from the MCU to the FPGA.
(2) Sensor Interface: Although MCUs have popular interfaces that allow direct connections to sensors, some intelligent sensors require relatively complicated interfaces and command sequences in order to be accessed. These can be accommodated by configuring a dedicated interface and a command sequence generator to obtain sensed data.
(3) Data Aggregating: If a sensor node has more than one sensor, the sensed data obtained from different sensors must be placed in packets. However, the data sampling rates may vary by sensor, and in some cases, post-

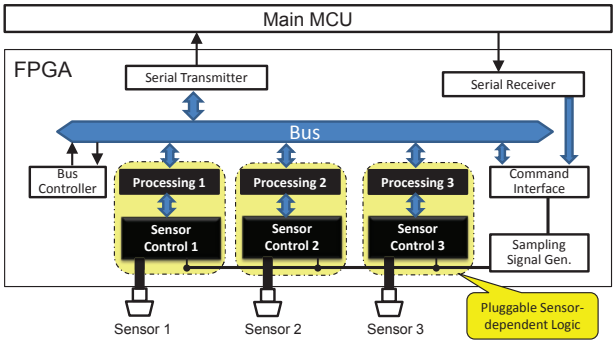

Fig. 7. Template circuit for the FPGA.

processing must be performed on raw sensed data prior to data packing. If the data aggregation task is performed in the FPGA, the sensor node can send a sensed data packet using the MCU to forward the aggregated data to the RF module.

The user can realize any functions including the abovementioned ones in the FPGA. However, some hardware design skills are needed to implement a circuit for the FPGA, and the FPGA's hardware resources are limited. Thus, to support easy hardware customization, we provided the template circuit shown in Fig 7. The circuit has a bidirectional serial link to the main MCU, and sensor control and data processing circuits are provided for each sensor. The sensor-dependent parts are pluggable. Thus, a new sensor is easily supported by providing the sensor control and processing parts without modifying other parts. The sampling signal generator sets up the sensor data sampling rate for each sensor. Using the dedicated processing part, we can obtain processed data such as the averaged and min/max value within a specific time period. All processed data are sent to the main MCU through a bus and the serial link. To recognize which sensed data are at the main MCU, a simple header is attached to the processed datum in the processing parts. The header includes the sensor ID and the data attribute indicating one of {"row," "averaged," "min," "max," "other"}. The sensing start/stop and sampling rate setting can be performed by the main MCU. To do so, we provided a set of commands. By sending the commands though the serial link from the main MCU, we can control all of the circuits shown in Fig 7.

4. Experimental results

We conducted experiments to verify the functionality of the HW/SW reconfigurable sensor node and the ODCWSN system. First, we provided a HW role for the FPGA using the template circuit shown in Fig 7, which activates all sensors and sends the sensed row data directly to the main MCU. The SW role running on the main

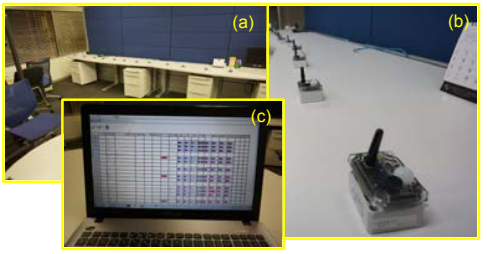

Fig. 8. Snapshots of the basic function test.
(a) and (b): Linearly located sensor nodes.; (c): A user terminal showing received sensor data.

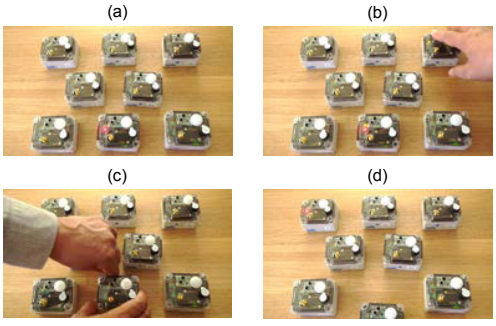

Fig. 9. Realization of the cluster-head election.
(a): The center-bottom node initially becomes a cluster head after all sensor nodes are switched on this time; (b): When the light sensor on the right-top node is covered with a finger, its yellow LED will blink. At the same time, the event will be sent to the current cluster head (the center-bottom one), and the yellow LED of the cluster head will blink; (c): The center-bottom node disappears, i.e., it is switched off; (d): Another node will become a new cluster head. In this case, it is the left-top node.

MCU just sends the sensed data received from the FPGA to a base station via the RF module. As shown in Fig 8(a) and (b), twelve sensor nodes are linearly placed on a table. The user terminal is connected to the base station and displays the received sensed data on the screen in real-time (see Fig 8(c)). Under this condition, a person passed near to the sensor nodes and shook some of the sensor nodes. We confirmed that some corresponding sensors accurately detected the human actions, and the events could actually be monitored at the user terminal. The measured power consumptions of the FPGA and the MCU were 21.2 mJ and 34.8 mJ on average, respectively. This proves that the FPGA can realize high-performance computing, which the MCU cannot, with low power consumption.

Next, we injected into the sensor node the role to solve the coverage problem shown in Fig 3, and the results confirmed that the role works well. Furthermore, we provided a role to solve a cluster-head election problem. The definition of the cluster-head election problem is as follows. If a sensor node does not have any one-hop neighboring sensor node that has a cluster-head mission, and its own node ID is the smallest among the one-hop neighboring sensor nodes, then the sensor node should be a new cluster head. In addition, if the current cluster-head sensor node finds other cluster-head sensor nodes in one-hop neighboring nodes, and its own node ID is larger than that of any one of them, the sensor node should resign as the cluster head. By invoking this role periodically at each sensor node, a cluster-head election and its replacement from among the sensor nodes can be done autonomously. In addition to the cluster-head election task, we also implemented the following function to enable us to visually check the dynamic cluster-head election role. If a sensor node becomes a cluster head, its red LED will be turned on. Moreover, if the light sensor in each sensor node detects darkness, its yellow LED should blink and the event should be transferred to the current cluster head. At the cluster head, if the darkness event arrives, the yellow LED of the cluster head should also blink. Fig 9 shows some snapshots proving that the cluster-head election and the related function work properly.

5. Conclusion

In this paper, we proposed an on-demand customizable wireless sensor network (ODCWSN) that can change the behavior of a WSN by injecting new roles into the sensor nodes. A role-generation mechanism was also introduced to realize easy specification of new sensor node behaviors. Furthermore, we developed a HW/SW dynamically reconfigurable sensor node, and the feasibility of the ODCWSN was demonstrated using the sensor node.

In future, we plan to implement a middleware for the HW/SW reconfigurable wireless sensor node. The middleware must support a light-weight role invocation by delivering only parts that are different from the running role on the sensor node, and by introducing a dynamic object-linking technique. This will be realized using information provided in[7].

Acknowledgements

This work is partly supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE No. 121802001).

References

1. Frank, C. and Romer, K. 2005. Algorithms for Generic Role Assignment in Wireless Sensor Networks. Proc. of ACM SenSys'05 (Nov. 2005), 230-242.
2. Jaskó, S. and Simon, G. 2010. Reconfigurable sensor network architecture for distributed measurement systems. Proc. of IEEE Instrum. and Meas. Technol. Conf. (3-6 May 2010). I2MTC2010. 198-203.
3. Bocca, M., Cosar E. I., Salminen J., and Eriksson L. M. 2009. A Reconfigurable Wireless Sensor Network for Structural Health Monitoring. Proc. of 4th Int. Conf. on Struct. Health Monit. of Intell. Infrastruct. (Zurich, Switzerland, 22-24 July 2009). SHMII-4, 2-9.
4. Krasteva, Y. E., Portilla, J., Carnicer, J.M., de la Torre, E., and Riesgo, T. 2008. Remote HW-SW reconfigurable Wireless Sensor Nodes. Proc. of IEEE 34th Annual Conf. of Ind. Electron. (10-13 Nov. 2008). IECON2008, 2483-2488.
5. Engel, A., Liebig, B., and Koch, A. 2012. Energy-efficient heterogeneous reconfigurable sensor node for distributed structural health monitoring. Proc. of Conf. on Des. and Archit. for Signal and Image Process. (23-25 Oct. 2012). DASIP2012, 1-8.
6. MEMSIC, Inc. IRIS MOTE datasheet. http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf.
7. Taherkordi A., Loiret F., Rouvoy R., and Eliassen F. 2013. Optimizing sensor network reprogramming via in situ reconfigurable components. ACM Trans. on Sensor Network. 9, 2 (March 2013), Article 14, DOI= http://doi.acm.org/10.1145/2422966.2422971.
8. Libelium. Waspmote. http://www.libelium.com/
9. Senceive. FlatMesh. http://www.senceive.com/index.php/flatmesh/