METROLOGY AND MEASUREMENT SYSTEMS
Index 330930, ISSN 0860-8229
www.metrology.pg.gda.pl

TRANSMISSION PROTOCOL SIMULATION FRAMEWORK FOR THE RESOURCE-CONSTRAINED WIRELESS SENSOR NETWORK

Marek Wójcikowski

Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
(✉ wujek@ue.eti.pg.gda.pl, +48 58 347 1974)

Abstract

In this paper the prototype framework for simulation of the wireless sensor network and its protocols are presented. The framework simulates the operation of the sensor network with data transmission, which enables the simultaneous development of the sensor network software, the sensor network hardware and the protocols for the wireless data transmission. The advantage of using the framework is the convergence of the simulation with the real software. Instead of creating the model of the sensor network node, the same software is used in real sensor network nodes and in the simulation framework. The operation of the framework is illustrated with examples of the simulations of the selected transactions in the sensor network.

Keywords: sensor network, computer simulations, sensor network protocols.

1. Introduction

Simulation of non-standard complex systems consisting of hardware and software is not an easy task. An example of such a complex system is a wireless sensor network (WSN). A WSN is a distributed set of autonomous devices, called nodes, collecting data and exchanging it among other nodes using wireless short-range transceivers. Sensor networks are very useful in collecting data from large areas, many potential examples of sensor networks may be found in the literature [1], [2], [3], [4], [5], [6]. The design of the network is usually prepared in three main steps:

1) Hardware design of the sensor network node.
2) The design of transmission protocol or adaptation of existing standard protocol.
3) Development of the software of the node.

Sensor networks usually have a very constrained hardware with limited resources such as power supply, processing capabilities or memory. Standard network protocols may not be sufficiently efficient; therefore the development of a custom protocol is often needed. Hardware and software design is a well-known process, but the designer of the sensor network must simultaneously develop hardware, software and protocols. For this non-standard approach, an accurate WSN simulator is essential.

There are many simulators that can be used for WSN simulation. Simulators may be divided into two classes: general-purpose network simulators and the frameworks dedicated for simulation or emulation of specific hardware and software. The well-known general-purpose network simulator ns-2 [7] is capable of simulating standard layered network protocols, but adding new custom protocols to ns-2 is very difficult due to the complex inter-module dependencies and it requires a profound knowledge of the internals of the simulator's object-oriented code. ns-3 [8] simulator is a continuation of ns-2, with a completely rewritten C++ code, but it is not reverse compatible with ns-2 and it lacks a support for WSNs. Simulators SSFNet [9] and GloMoSim [10] are capable of parallel simulations and they are similar in complexity to ns-2. The simulator SENSE [11] is a component-port model based simulator; therefore it is easy to extend by building custom components and simulation engines. It is excellent for modelling the general operation of the sensor network, without great knowledge of the internals of software implementation. Another simulator J-Sim [12][13], intended for WSN simulations, uses Java, therefore it is easy to extend. It also uses the detail models of the node and radio channels and can estimate the power consumptions in the sensor network. A popular component-based network simulation library and framework with powerful GUI interface OMNeT++ [14], written in C++, may also be used for wireless sensor networks.

TOSSIM [15] is an example of the dedicated simulator, specifically designed for WSNs using the TinyOS [16] operating system. ATEMU [17] is an emulator of an AVR processor for sensor networks built in C. The EmStar [18] framework is dedicated to develop sensor networks based on the *microserver* hardware platforms, which are more complex and powerful than typical WSN hardware. The detailed survey of simulation tools may be found in [19].

Most standard simulators require construction of the model of the sensor network node, which introduces discrepancy between the model and real hardware. At a later stage of the sensor network design process, complex debugging must be done, requiring great insight into the internal structure of the network node and internal state of the network for sophisticated bug tracing. An accurate simulator capable of tracking the transmitted packets is necessary for this purpose.

In this paper the simulation framework for the simulation of the custom protocol together with firmware and software of the sensor network nodes is described. The presented method has been used during the design of the prototype sensor network for urban traffic monitoring [20]. The proposed simulator was capable of debugging complex problems of data transmission between different sensor network nodes, at the same time being a very accurate model of the software operation in the real hardware.

2. The sensor network

The typical sensor network node consists of the microprocessor system with sensor subsystem, the transceiver and the power supply. In this example of the sensor network for traffic monitoring, the microprocessor system is a custom system on a chip (SoC) with 32-bit microprocessor and Wishbone buses. The sensing subsystem is rather complex: the video stream from the on-board camera is continuously analysed and segmented to detect moving vehicles. For the radio transmission, the ISM band 868MHz 500mW transceiver module is used. The power supply circuitry is designed to provide the power to the system and also to recharge the battery from the mains power supply and solar cells, using the maximum power point tracking (MPPT) method.

Each sensor network node is running the dedicated single-thread software written in C with the interrupt service routines (ISRs) written also in C and partly in the assembler. The cooperation between the nodes is based on the custom transmission protocol. The transceiver provides only one transmission channel, which must be shared by all the nodes using the time division multiple access (TDMA) technique. To provide acceptable transmission rates, the nodes must synchronise their transmissions in order not to jam each other and to prevent from "the hidden terminal" scenario. For this purpose, the transmission is divided into time slots, and each node selects its own transmission time slot. The nodes have local clocks and they track the existence and the transmission times of the neighbouring nodes. This data is kept in the local table of each node (referred to as NBR table). Moreover, the neighbours of the neighbours, (called indirect neighbours), are also detected by listening to the transmissions nearby; this information helps to avoid jamming owing to the hidden terminal problem. The transceiver must be turned on some time before the start of the transmission, due to the hardware warm-up time. This warm-up time is different for transmission and reception. The details of the transmission protocol are described in [20].

3. The simulation environment

The simulation of such a sensor network, being the mixture of sensor nodes' hardware, C-software and radio transmission protocol is a challenging task. The simulation is especially important, when applied to the development of the radio transmission protocol. The author of this paper has created the software framework environment, which enables one to embed the C software of the node. The framework is capable of emulation of any number of sensor network nodes (constrained by the host computer's memory and speed), performing the function of a multi-virtual machine for the sensor network nodes, while simultaneously simulating the radio transmissions in the area between the nodes.

The use of the framework enabled simulation of exactly the same software as the software used in the real hardware nodes, except for a few functions closely related to the hardware, such as the transceiver's drivers. The simulation of the radio transmission is based on a simple model with discrete transmission ranges with the artificially injected transmission jams at random moments. The concept of the simulation framework is presented in Fig. 1.

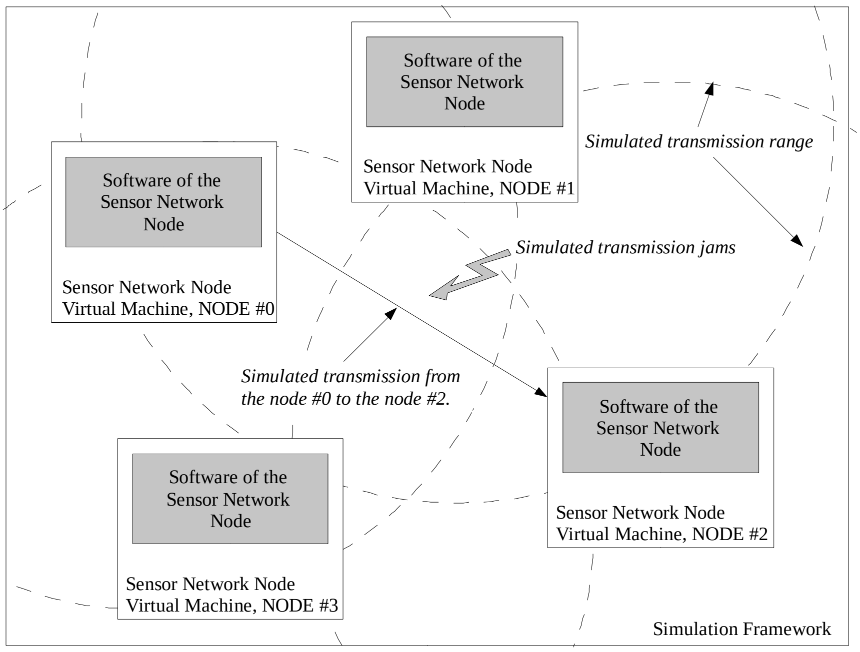

Fig. 1. The simulation framework for the sensor network.

The simulation framework has been written in C++ and Tcl/Tk. C++ code has been used for all calculations; Tcl/Tk scripts provide the graphical user interface (GUI) and data visualisation. The source C code from the single node is included in the C++ source code of the simulation framework and then compiled and executed on the host workstation. The main screen of the simulator is shown in Fig. 2.

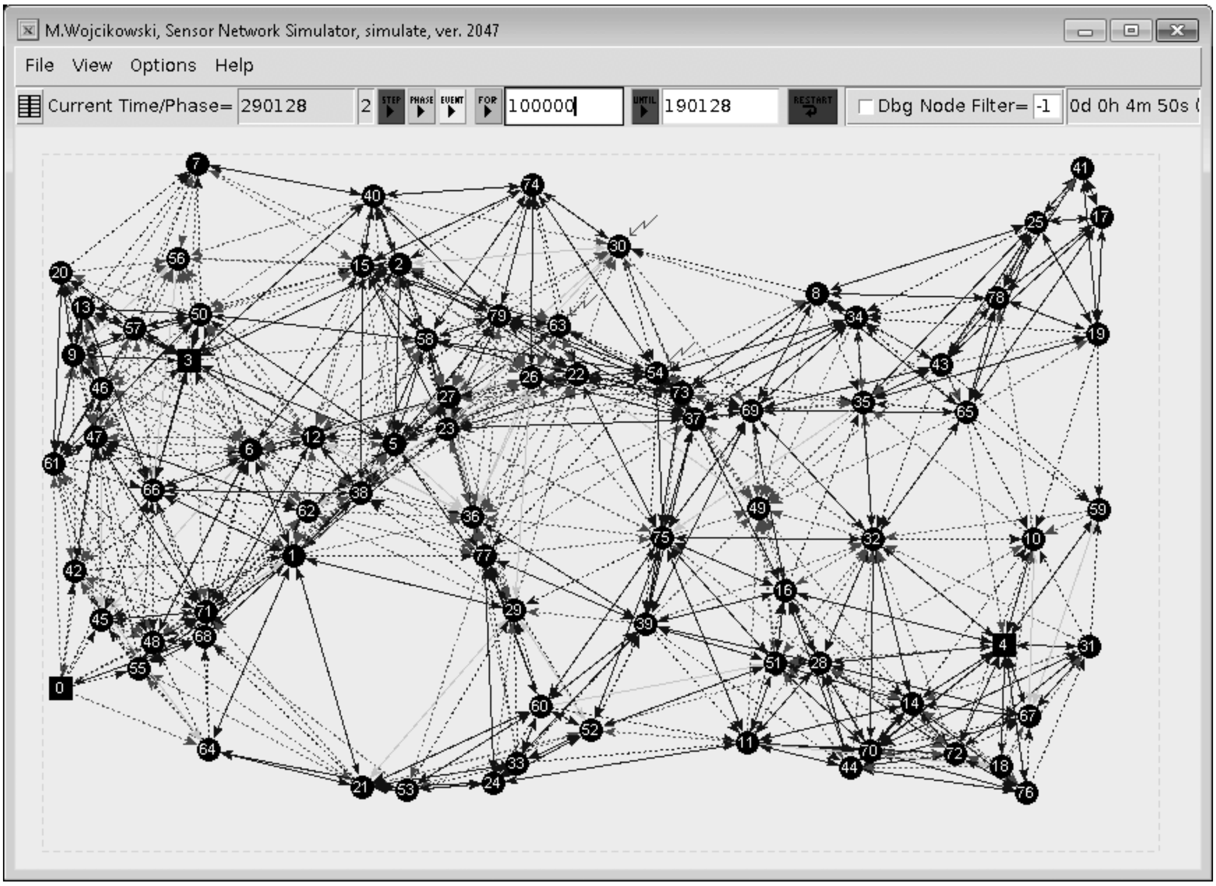

Fig. 2. The main screen of the sensor network simulator.

The GUI displays the geometrical placement of the nodes; the nodes may be moved using the mouse. The buttons enable control of the simulation: it is possible to run the simulation for the specified time (the button FOR=for the specified time, the button UNTIL=until the specified time), more precise control is possible using STEP, PHASE and EVENT buttons. STEP runs the simulation for one time unit. Each simulation time unit is divided into three phases. In phase 0 all the nodes are checked, if they are ready to start the data transmission or reception, i.e. they are turned on and their local state denotes the transmission or the reception mode. In phase 1 all the active transmissions are made by copying data from the transmitting node to the receiving node, provided that the appropriate node pairs are within their transmission range. Phase 2 is for finalising the transactions. It is also possible to run the simulation until the next event in the network. The event is defined here as the meaningful change of the state of the node, such as the start of the data reception or the start of the data transmission. The events may be filtered using the node number, providing simulation fine control.

To show the use of the simulator, one of the network transaction scenarios is presented in this paper. The proposed sensor network has auto-configuration property. Each node at regular intervals enters the new neighbour's discovery mode. The new nodes are recorded in the node's local NBR table. Those nodes that have not been detected for a long time are deleted from the table. The simulator provides the possibility of displaying the NBR table of each node. The process of establishing the connection between two nodes is shown in the example described below. For simplicity, only the 2-node network is considered in this example:

1. At the beginning both nodes #0 and #1 are unaware of each other. Their NBR tables are empty. The node #1 starts it discovery mode, the node #0 transmits regular beacon signal (Fig. 3)
2. The node #1 has received the beacon signal from the node #0. The node #1 writes an entry about the node #0 with "hear" attribute in its local NBR table (Fig. 4).
3. The node #1 quits its discovery mode; both nodes are regularly transmitting beacons. From this time, the beacon of the node #1 will contain information, that the node #1 can "hear" the transmissions from the node #0.
4. The node #0 enters discovery mode and receives the beacon from the node #1. The node #0 records the node #1 as "normal" node in its NBR table, because the beacon received from the node #1 has information, that the node #1 can hear the node #0. From this time, the beacon from the node #0 contains information, that the node #1 is a regular neighbour of node #0, i.e. the transmission in both directions has been confirmed (Fig. 5).
5. The node #0 quits the discovery mode and it starts regular transmissions of the beacons. The beacon contains information, that the node #0 can hear the node #1. Once the node #1 has received this beacon, both nodes #1 and #2 are aware of each other and they are considered as mutually connected (Fig. 6).

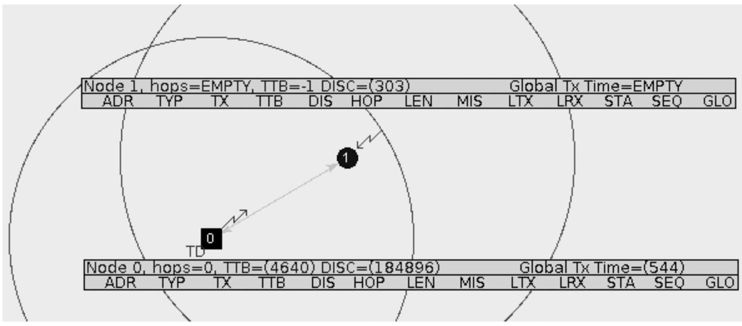

Fig. 3. Initial state of the simulation. Node #0 and #1 are presented together with their NBR tables; both nodes are not connected to each other yet (their NBR tables are empty). Grey arrow indicates that the nodes are in their mutual transmission ranges. Zigzag arrows denote transmission or reception currently occurring at the node, as indicated by the direction of the arrow. In this figure the node #0 is transmitting data; the node #1 is receiving it.

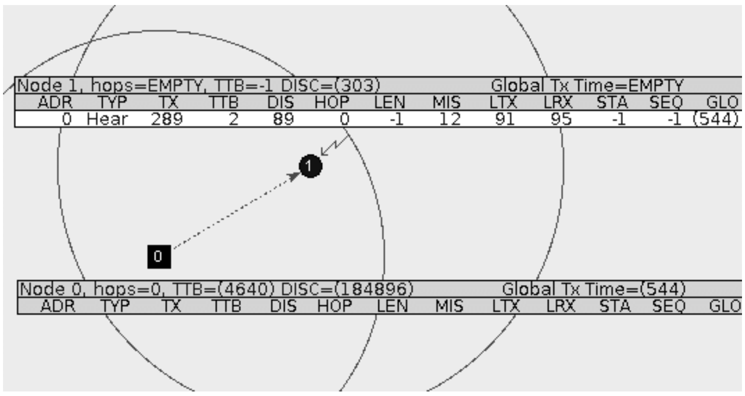

Fig. 4 The node #1 has received the beacon signal from the node #0. The NBR table of the node #1 now contains information about hearing the node #0.

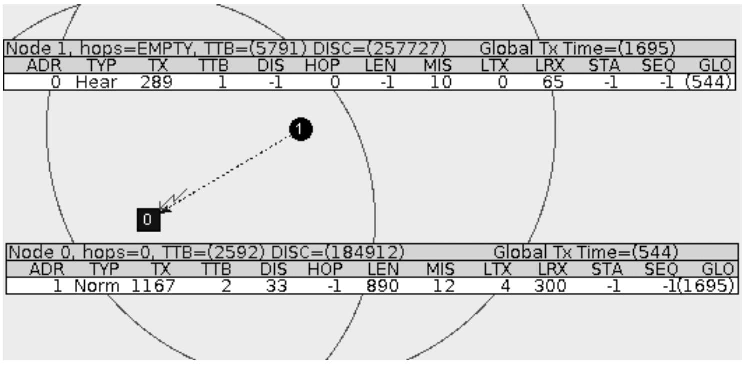

Fig. 5. The node #0 received beacon signal from the node #1.

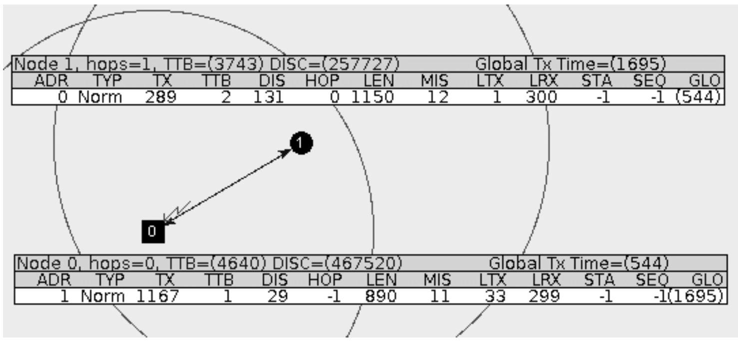

Fig. 6. Both nodes know about each other and they have exchanged information about each other.

During the transmission, the nodes also overhear the transmissions of the neighbouring nodes to detect the target of their transmissions. It is thus possible to detect the indirect neighbours. After the initial time, all possible links should have been established, as shown in 3-node example in Fig. 7. Please note that the nodes #0 and #2 know about each other, despite the fact that they are not within the transmission range.

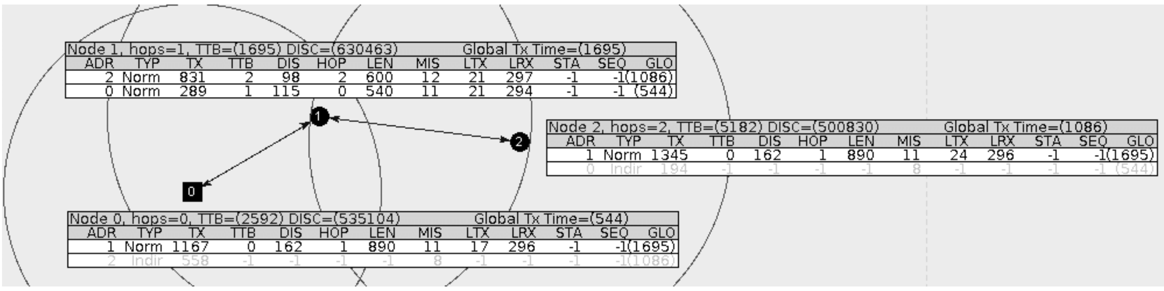

Fig. 7. The links established between 3 nodes.

As a result of establishing the bi-directional links between the nodes, each node knows the beacon transmission times of all the nodes from the NBR table. Due to this, the node turns on its receiver only when one of the neighbours is transmitting data, avoiding overhearing. Moreover, each node refrains from making transmissions concurrent with the transmissions of the indirect neighbours, which prevents the hidden terminal problem.

The simulation framework enables an easy inspection of the state of the sensor network and its nodes. It is possible to stop the simulation at any moment and track problematic situations. Each change in the code of the sensor network node is immediately reflected in the simulator, due to the fact that the software used in sensor network hardware is embedded without any changes in the simulation framework. The simulation speed as a function of network size is shown in Fig. 8.

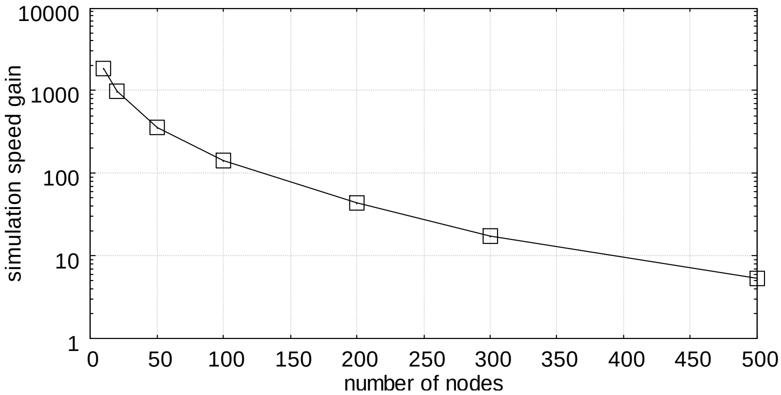

Fig. 8. The speed of the simulator as a function of the number of nodes in the sensor network. The simulation speed gain shows how the simulation runs faster in comparison to the time of operation of the real network, while performing the same tasks. The simulations were performed on a PC with Intel i7 3.07GHz processor and 8GB RAM.

4. Conclusions

The presented sensor network simulation framework has been designed and used by the author when designing the resource-constrained sensor network and its data transmission protocols. As has been shown in the presented examples, the simulation framework enables the development of the sensor network data transmission protocol simultaneously with the development of the software of the sensor network nodes. The framework enables one to track the network transactions and to observe the internal state of the nodes and their variables (i.e. NBR tables).

In comparison to most simulators known from the literature, the presented simulator framework does not require construction of the dedicated WSN model. Model-based simulators can achieve higher simulation speeds, but the use of the model simplifies the network's operation, which prevents the tracing of complex problems. The emulation of the hardware is used instead, thus the great advantage of the simulator is the possibility to use the same C code that has been implemented in the real hardware. This helps to maintain the convergence of the simulation with the real hardware operation. The simplified radio channel is sufficiently suitable to develop the transmission protocol but for more detailed development it would be necessary to implement more complex transmission channel models. The real wireless sensor network with its network protocol has been successfully designed using the presented simulation framework. The designed network has operated for more than one year, collecting data from the environment for research purposes.

References

[1] Akyildiz, I. F., Weilian Su, Sankarasubramaniam, Y., Cayirci, E. (August 2002). A survey on sensor networks. *Communications Magazine, IEEE*, 40(8), 102- 114.

[2] Karl, H., Willig, A. (2007). *Protocol and Architecture for Wireless Sensor Networks*. John Wiley and Sons, Ltd.

[3] Stojmenovic, I. (2005). *Handbook of Sensor Networks*. John Wiley and Sons, Inc.

[4] Yick, J., Mukherjee, B., Ghosal, D. (August 2008). Wireless sensor network survey. *Computer Networks*, 52(12), 2292-2330.

[5] Bojko, T. (2005). Smart Sensor Solutions for Mechanical Measurements and Diagnostics, Metrology and Measurement Systems, 12(1), 95-103.

[6] Pereira, D. J. M., Viegas, V., Postolache, O., Girao, P. S. (2013). A Smart and Distributed Measurement System to Acquire and Analyze Mechanical Motion Parameters, Metrology and Measurement Systems, 20(3), 465-478.

[7] The Network Simulator, ns–2. *http://www.isi.edu/nsnam/ns*

[8] ns-3. http://www.nsnam.org

[9] Scalable Simulation Framework (SSF). *http://www.ssfnet.org*

[10] Global Mobile Information Systems Simulation Library (GloMoSim). *http://pcl.cs.ucla.edu/projects/glomosim*

[11] Chen, G., Branch, J., Pflug, M., Zhu, L., Szymanski, B. (2005). Sense: A Wireless Sensor Network Simulator. *Advances in Pervasive Computing and Networking*, Springer US, 249-267.

[12] Sobeih, A., Chen, W.-P., Hou, J. C., Kung, L.-C., Li, N., Lim, H. , Tyan, H.-Y., Zhang, H. (Aug. 2006). J-Sim: a simulation and emulation environment for wireless sensor networks. *Wireless Communications, IEEE*, 13(4), 104,119.

[13] J-Sim. *http://j-sim.cs.uiuc.edu*

[14] OMNET++ discrete event simulator. http://www.omnetpp.org

[15] Levis, P., Lee, N., Welsg, M., Culler, D. (2003). TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In Proc. *1st ACM Int. Conf. Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, 126–137.

[16] TinyOS: Open-source operating system for wireless embedded sensor networks. *http://www.tinyos.net*

[17] Polley, J., Blazakis, D., McGee, J., Rusk, D., Baras, J. S., Karir, M. (October 2004). ATEMU: A Fine-grained Sensor Network Simulator. In Proc. *1st IEEE Int. Conf. Sensor and Ad-hoc Communications and Networks (SECON'04)*, Santa Clara, CA, 145-152.

[18] Girod, L., Elson, J., Cerpa, A., Stathopoulos, T., Ramanathan, N., Estrin, D. (2004). EmStar: A software Environment for Developing and Deploying Wireless Sensor Networks. In Proc. *2004 USENIX Technical Conference*, Boston, MA, 283–296.

[19] Egea-Lopez, E., Vales-Alonso, J., Martinez-Sala, A. S., Pavon-Marino, P., García-Haro, J. (July 2005). Simulation tools for wireless sensor networks. In Proc. *Summer Simulation Multiconference-SPECTS* (Vol. 2005).

[20] Wójcikowski, M., Żaglewski, R., Pankiewicz, B., Kłosowski, M., Szczepański, S. (Apr. 2013). Hardware-Software Implementation of a Sensor Network for City Traffic Monitoring Using the FPGA- and ASIC-Based Sensor Nodes. *Journal of Signal Processing Systems*, 71(1), 57-73.