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

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

Abstract

In this paper a prototype framework for simulation of wireless sensor network and its protocols are presented. The framework simulates operation of a sensor network with data transmission, which enables simultaneous development of the sensor network software, its hardware and the protocols for wireless data transmission. An advantage of using the framework is converging simulation with the real software. Instead of creating a model of the sensor network node, the same software is used in real sensor network nodes and in the simulation framework. Operation of the framework is illustrated with examples of simulations of 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 with 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 [1–6]. The network design usually consists of three main steps:

1. Design of the sensor network node hardware.
2. Design of a transmission protocol or adaptation of an existing standard protocol.
3. Development of the node software.

Sensor networks usually have a very constrained hardware, limited to such resources, as: a power supply, processing equipment, and memory. Standard network protocols may not be sufficiently efficient; therefore development of a customized protocol is often needed. Hardware and software design is a well-known process, but the designer of a 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 frameworks dedicated for simulation or emulation of specific hardware and software. A well-known general-purpose network simulator ns-2 [7] is capable of simulating standard layered network protocols, but adding new customized protocols to ns-2 is very difficult due to complex inter-module dependencies, and it requires a profound knowledge of the internals of the simulator's object-oriented code. The ns-3 [8] simulator is a continuation of ns-2, with a completely rewritten C++ code. However, it is not reverse compatible with ns-2 and lacks a support for WSNs. The SSFNet [9] and GloMoSim [10] simulators are capable of parallel simulations and their

Article history: received on Jul. 11, 2014; accepted on Jan. 27, 2015; available online on May 29, 2015; DOI: 10.1515/mms-2015-0019.

complexity is similar to that of ns-2. The SENSE simulator [11] is a component-port model based simulator; therefore it is easy to extend by building customized components and simulation engines. It is excellent for modelling general operation of a 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 detailed models of a node and radio channels, and can estimate the power consumption 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 a dedicated simulator, specifically designed for WSNs using the TinyOS [16] operating system. ATEMU [17] is an emulator of AVR processor for sensor networks built in C. The EmStar [18] framework is dedicated to develop sensor networks based on *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 a sensor network node model, 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 network node and internal state of network for sophisticated bug tracing. An accurate simulator capable of tracking the transmitted packets is necessary for this purpose.

In this paper a simulation framework for simulation of a customized protocol together with firmware and software of sensor network nodes is described. The presented method has been used during design of a 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 software operation in the real hardware.

2. The sensor network

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

Each sensor network node is running dedicated single-thread software written in C with interrupt service routines (ISRs) written also in C and partly in the assembler. The cooperation of the nodes is based on a customized 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. The data are 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 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

Simulation of such a sensor network, being a mixture of sensor nodes' hardware, C-software and radio transmission protocol, is a challenging task. The simulation is especially important, when applied to development of the radio transmission protocol. The author of this paper has created the software framework environment, which enables embedding the C software of the node. The framework is capable of emulation of any number of sensor network nodes (limited 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 among the nodes.

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

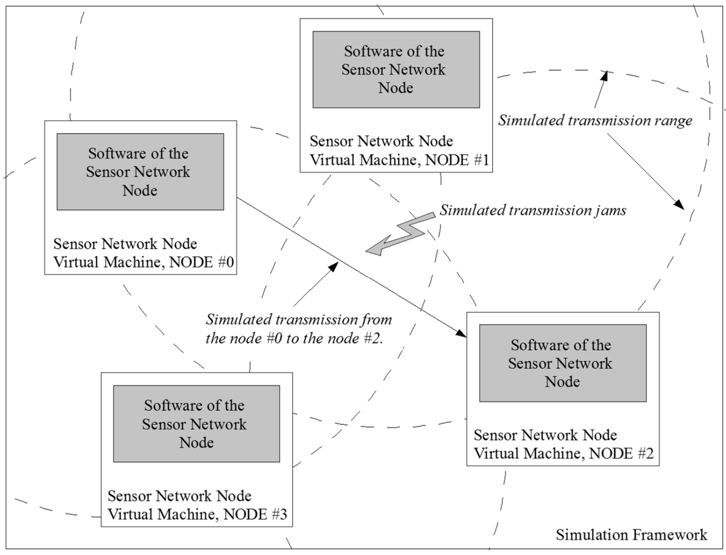

Software of the Sensor Network Node
Sensor Network Node Virtual Machine, NODE #1
Simulated transmission range
Software of the Sensor Network Node
Sensor Network Node Virtual Machine, NODE #0
Simulated transmission jams
Simulated transmission from the node #0 to the node #2.
Software of the Sensor Network Node
Sensor Network Node Virtual Machine, NODE #2
Software of the Sensor Network Node
Sensor Network Node Virtual Machine, NODE #3
Simulation Framework

Fig. 1. The simulation framework for a 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 a 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.

The GUI displays the geometrical arrangement 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 a specified time (the FOR button = for a specified time, the UNTIL button = until a 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, whether they are ready to start data transmission or reception, *i.e.*, they are turned on and their local states denote either the transmission or reception mode. In phase 1 all 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. An event is defined here as a meaningful change of the node state, such as the start of data reception or the start of data transmission. The events may be filtered using the node number, providing fine control of simulation.

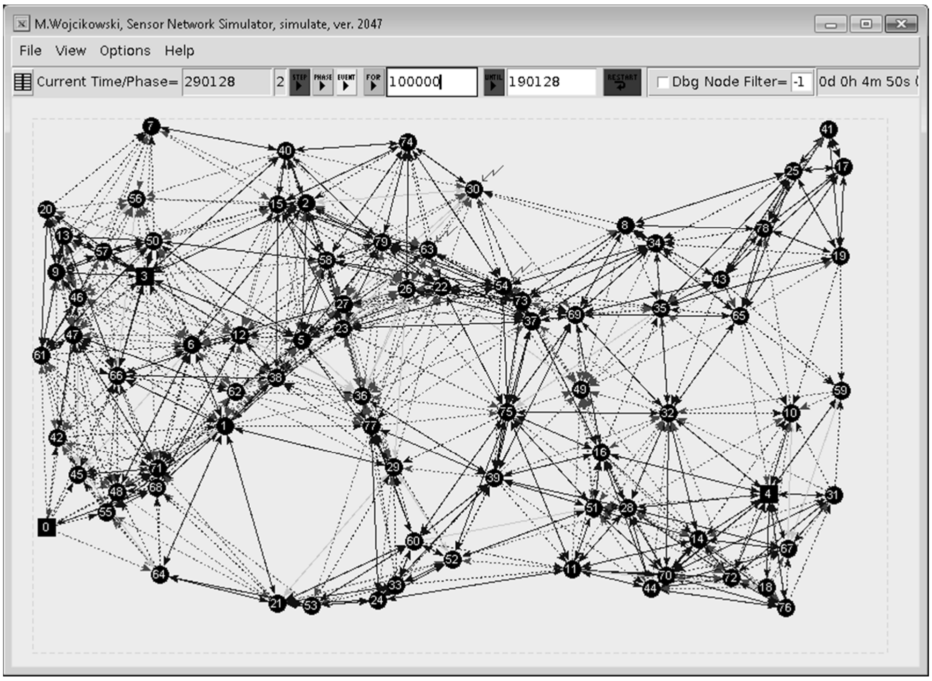
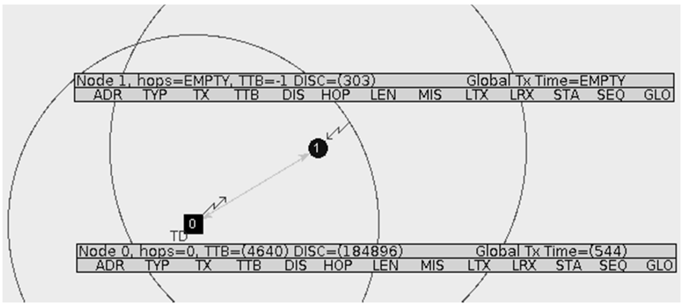


Fig. 2. The main screen of the sensor network simulator.

To show the use of simulator, one of the network transaction scenarios is presented in this paper. The proposed sensor network has the 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 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. Node #1 starts its discovery mode, node #0 transmits the regular beacon signal (Fig. 3).
2. Node #1 has received the beacon signal from node #0. Node #1 writes an entry about node #0 with "hear" attribute in its local NBR table (Fig. 4).
3. Node #1 quits its discovery mode; both nodes are regularly transmitting beacons. From this time, the beacon node #1 contains information, that node #1 can hear the transmission from node #0.

4. Node #0 enters its discovery mode and receives the beacon from node #1. Node #0 records node #1 as a "normal" node in its NBR table, because the beacon received from node #1 has information that node #1 can hear node #0. From this time, the beacon from node #0 contains information that node #1 is a regular neighbour of node #0, *i.e.*, the transmission in both directions has been confirmed (Fig. 5).
5. Node #0 quits the discovery mode and starts regular transmission of the beacons. The beacon contains information that node #0 can hear node #1. Once 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).



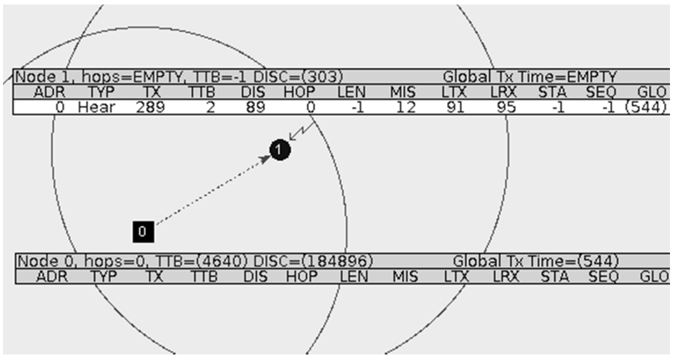
Node 1, hops=EMPTY, TTB=-1 DISC=(303) Global Tx Time=EMPTY

ADR	TYP	TX	TTB	DIS	HOP	LEN	MIS	LTX	LRX	STA	SEQ	GLO

Node 0, hops=0, TTB=(4640) DISC=(184896) Global Tx Time=(544)

ADR	TYP	TX	TTB	DIS	HOP	LEN	MIS	LTX	LRX	STA	SEQ	GLO

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



Node 1, hops=EMPTY, TTB=-1 DISC=(303) Global Tx Time=EMPTY

ADR	TYP	TX	TTB	DIS	HOP	LEN	MIS	LTX	LRX	STA	SEQ	GLO
0	Hear	289	2	89	0	-1	12	91	95	-1	-1	(544)

Node 0, hops=0, TTB=(4640) DISC=(184896) Global Tx Time=(544)

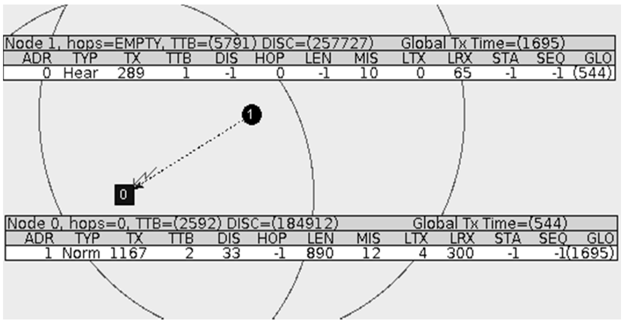
ADR	TYP	TX	TTB	DIS	HOP	LEN	MIS	LTX	LRX	STA	SEQ	GLO

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

During the transmission, the nodes also overhear 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 a 3-node example in Fig. 7. Please note, that nodes #0 and #2 know about each other, despite the fact that they are not within the transmission range.

As a result of establishing the bi-directional links between nodes, each node knows the beacon transmission times of all the nodes from its NBR table. Due to this, the node turns on its receiver only when one of its neighbours is transmitting data, avoiding overhearing.

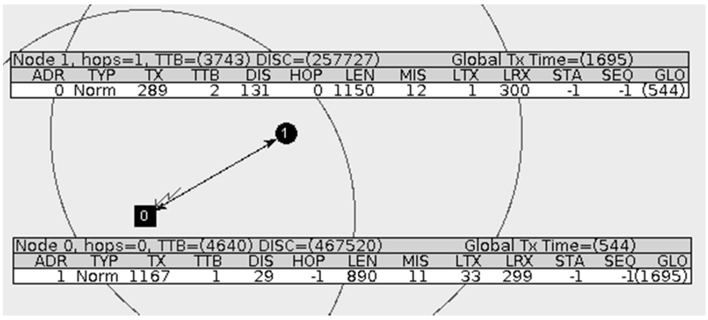
Moreover, each node refrains from making transmissions concurrent with transmissions of the indirect neighbours, which prevents the hidden terminal problem.



```
Node 1, hops=EMPTY, TTB=(5791) DISC=(257727)      Global Tx Time=(1695)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   0 Hear  289     1    -1    0   -1    10     0   65   -1   -1 (544)

Node 0, hops=0, TTB=(2592) DISC=(184912)          Global Tx Time=(544)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   1 Norm 1167     2    33   -1  890    12     4  300   -1  -1(1695)
```

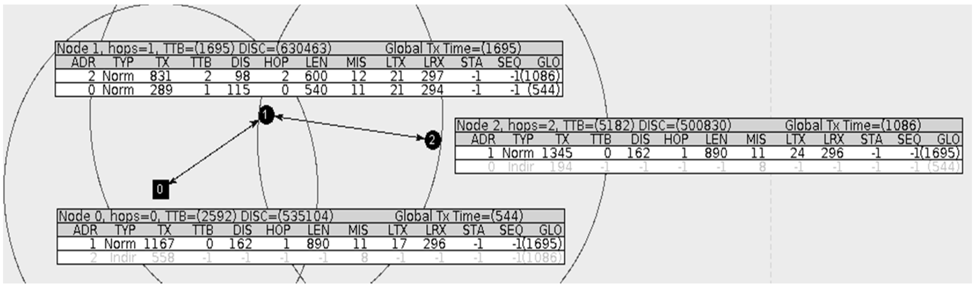
Fig. 5. Node #0 has received the beacon signal from node #1.



```
Node 1, hops=1, TTB=(3743) DISC=(257727)          Global Tx Time=(1695)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   0 Norm  289     2   131    0 1150    12     1  300   -1   -1 (544)

Node 0, hops=0, TTB=(4640) DISC=(467520)          Global Tx Time=(544)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   1 Norm 1167     1    29   -1  890    11    33  299   -1  -1(1695)
```

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



```
Node 1, hops=1, TTB=(1695) DISC=(630463)          Global Tx Time=(1695)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   2 Norm  831     2    98    2  600    12    21  297   -1  -1(1086)
   0 Norm  289     1   115    0  540    11    21  294   -1   -1 (544)

Node 2, hops=2, TTB=(5182) DISC=(500830)          Global Tx Time=(1086)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   1 Norm 1345     0   162    1  890    11    24  296   -1  -1(1695)
   0 Indir 194    -1    -1   -1   -1     8    -1   -1   -1   -1 (544)

Node 0, hops=0, TTB=(2592) DISC=(535104)          Global Tx Time=(544)
 ADR  TYP   TX   TTB   DIS  HOP  LEN   MIS   LTX  LRX  STA  SEQ   GLO
   1 Norm 1167     0   162    1  890    11    17  296   -1  -1(1695)
   2 Indir 558    -1    -1   -1   -1     8    -1   -1   -1  -1(1086)
```

Fig. 7. The links established between 3 nodes.

The simulation framework enables an easy inspection of the state of sensor network and its nodes. It is possible to stop the simulation at any moment and track problematic situations. Each change in the sensor network node code 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.

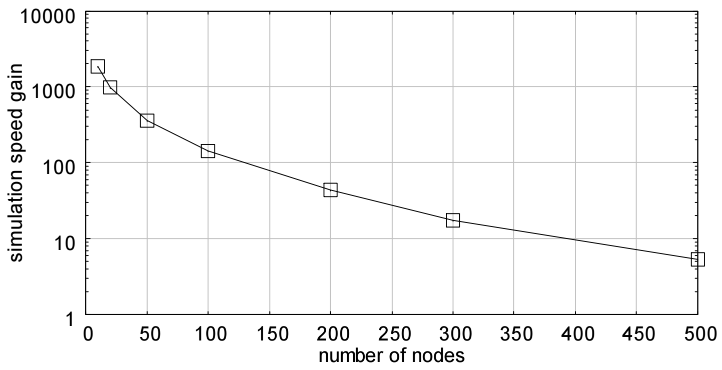


Fig. 8. The speed of simulator as a function of the number of nodes in the sensor network. The simulation speed gain shows how the simulation runs faster in comparison to the time of operation of the real network, while performing the same tasks. The simulations were performed on a PC with Intel i7 3.07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

4. Conclusions

The presented sensor network simulation framework has been designed and used by the author when designing a resource-constrained sensor network and its data transmission protocols. As it has been shown in the presented examples, the simulation framework enables development of a sensor network data transmission protocol simultaneously with development of the software of sensor network nodes. The framework enables tracking network transactions and observing the internal state of 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 a dedicated WSN model. Model-based simulators can achieve higher simulation speeds, but using the model simplifies the network's operation, which makes tracing of complex problems not feasible. The emulation of the hardware is used instead, thus a 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 a 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, S., Sankarasubramaniam, Y., Cayirci, E. (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. (Aug. 2008). Wireless sensor network survey. *Computer Networks*, 52(12), 2292–2330.

[5] Bojko, T. (2005). Smart Sensor Solutions for Mechanical Measurements and Diagnostics. *Metrol. Meas. Syst.*, 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. *Metrol. Meas. Syst.*, 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. *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. (Oct. 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. *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. (Jul. 2005). Simulation tools for wireless sensor networks. *Proc. Summer Simulation Multiconference-SPECTS* .

[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.