Implementation and validation of an energy-efficient MAC scheduler for WSNs by a test bed approach

*D. Alessandrelli, **L. Mainetti, **L. Patrono, *G. Pellerano, *M. Petracca, **M.L. Stefanizzi

*Scuola Superiore Sant'Anna
Real-Time Systems Laboratory
Via Moruzzi 1, Pisa, Italy

**Salento University
Department of Innovation Engineering
Via Monteroni, Lecce, Italy

E-mail: laura.stefanizzi@unisalento.it, giovanni.pellerano@evilaliv3.org

Abstract: **The paper presents an energy efficient MAC scheduler for wireless sensor networks, and its implementation in the Contiki Operating System. Simulations performed using the Contiki's simulations tools (i.e., Cooja and MPSim) show that the proposed scheme reduces the power consumption with respect to the ZigBee standard solution and the X-MAC protocol, already implemented in Contiki. Furthermore, the functionality and the portability of the implementation are validated using two test beds based on two different platforms.**

1. INTRODUCTION

A Wireless Sensor Network (WSN) is composed of a large number of spatially distributed sensor nodes that cooperate in order to monitor, collect, process and share data acquired from the environment in which they are located. The recent progress of wireless technology has enabled the development of low-cost, low-power, multifunctional sensor nodes that are characterized by ad hoc communications. These features make WSNs suitable to be used for a wide range of potential applications including environment monitoring, home and building automation, healthcare, and robotic exploration.

However, the realization of such applications requires the use of efficient power management techniques. Sensor nodes, in fact, are usually battery powered and deployed in large areas in which change or recharge batteries could result completely unfeasible. Therefore, energy consumption is a primary issue to be considered, and the use of effective solutions to increase the WSN nodes lifetime is fundamental in real applications.

Considering that radio communications consume the majority of mote energy [1], many research works have been focused on developing medium access control (MAC) protocols able to minimize the radio energy consumption. Since the most prevalent source of energy waste in radio communication is the idle listening (i.e., listening of the channel when there is no activity on it), a careful selection of the receiving time could significantly improve the network lifetime.

In literature several enhancements to minimize energy consumption in WSNs have been presented. Most of these works are validated only by means of mathematical or simulation approaches that often show significant limits. The use of a real test bed, on the contrary, is able to overcome theoretical limits, thus guaranteeing that a proposed protocol can be actually used on real devices. This approach also demonstrates not only the efficiency but also the feasibility of the proposed solution. In fact, a protocol able to ensure excellent performance in a simulated environment may present implementing problems on physical devices or may demonstrate highly degraded performance in a real environment. The main reason is due to the difficulty of simulating the behavior of a WSN, which depends on events that are not controllable in a mathematical or simulation model (e.g., software installed, atmospheric and environment conditions, etc.).

In this work, an energy efficient duty cycling mechanism based on an asynchronous scheduler is presented and validated by means of simulations and experiments performed in a real test bed. The scheduler is an enhancement of the solution reported in [2] that solves problems not previously considered (e.g., the hidden node problem) or not properly analyzed (e.g., the joining of a new node) and takes into account technical requirements related to physical boards and the Contiki Operating System (OS) [3]. The proposed solution is able to reduce energy consumption preventing unnecessary awakenings of the node. Each node communicates to its neighbors the interval time before the data transmission, so that they can set their wake up time accordingly. This way, every node has the list of the transmission times for all of its neighbors and knows in advance when it is supposed to be awake to receive data or can switch to the sleep mode to save energy. The solution is able to face topology changes since the information stored by each node is updated as soon as it is not able to listen a neighbors transmission or is aware of a new node presence.

In order to analyze the effectiveness of the proposed scheme, a performance comparison by using two different kinds of sensor nodes running the Contiki OS was carried out. We first compared the network power consumption of the defined protocol with the current ZigBee [4] standard solution and the

well-known X-MAC [5] protocol, a very energy-efficient MAC solution, by using the Contiki simulation environment. Then we demonstrated the actual functionality and portability of proposed enhancement by means of two different type of sensor node: MB851 [6] and Seed-eye board [7].

The rest of the paper is organized as follows. Section 2 summarizes the state-of-the-art for energy consumption minimization in WSNs and the use of test bed approaches. The proposed scheduler is described in Section 3. The test bed environment is given in Section 4, while in Section 5 numerical results are discussed. Conclusions are drawn in Section 6.

2. RELATED WORKS

Energy-efficient MAC protocols are a very active research area in wireless embedded and sensor networks. All protocols presented in literature fit into three main categories: *preamble-sampling*, *scheduling* and *hybrid* approaches.

Preamble-sampling MAC protocols, such as BMAC [8], WiseMAC [9] and X-MAC [5], exploit the Low-Power-Listening (LPL) [10] technique for sampling packet preambles. LPL minimizes the duty cycle when there are no packets to be exchanged, but during transmissions it needs a preamble longer than the wakeup interval to guarantee the receiver channel activity detection. WiseMAC reduces the length of the wakeup preamble by exploiting knowledge of the sensor nodes' sampling schedules. X-MAC solves the overhearing problem by using a strobed preamble that consists of a sequence of short preambles prior to data transmission.

S-MAC [11] and T-MAC [12] are scheduling MAC protocols. They synchronize the wakeup schedules of sensor nodes in a neighborhood by exchanging synchronization packets. Such message exchanges can cause high overhead and consume significant energy even when there is no traffic. S-MAC was the first duty cycling MAC protocol for WSNs and it developed a distributed coordination scheme to synchronize node sleep schedules in a multi-hop network. T-MAC improves S-MAC by reducing the wakeup duration controlled by an adaptive timer and introducing future request to send.

Hybrid approaches, such as SCP-MAC [13] and AS-MAC [14], combine preamble sampling with scheduling techniques. SCP-MAC minimizes the length of preamble by exploiting the synchronization of the wake up time of neighboring nodes. It reduces the periodic wakeup time by LPL. AS-MAC uses LPL to minimize the periodic wakeup time and it also asynchronously coordinates the wakeup times of neighboring nodes to reduce overhearing, contention and delay. One of the main disadvantages of this protocol is the inefficiency in broadcasting since AS-MAC has to transmit every packet once per each neighbor.

The proposed scheduler employs duty cycling like the previous schemes to avoid idle listening and asynchronously coordinates the wakeup times of neighboring nodes. This way, it reduces the number of synchronization packets typically used in state-of-the-art MAC schedulers. It also reduces the overhearing problem of preamble-sampling MAC protocol. The nodes, in fact, store the scheduled wakeup times of their neighbors, therefore, they do not need to add long preambles at the beginning of transmission nodes.

Many works describe new protocols validated by mathematical or simulation approach. These techniques often show an incomplete analysis because they do not take into account physical requirements of real boards. In order to bypass this problem a test bed approach is needed.

In [11], the authors use Motes as development platform and test bed to demonstrate the effectiveness of S-MAC. The motes were running TinyOS [15], an efficient event-driven operating system for tiny sensor nodes. The goal of the experimentation was to reveal main trade offs among energy, latency, and throughput. For this purpose, they used two different network topologies: two-hop network with two sources and two sinks, and a ten-hop linear network with one source and one sink. In [13] the average energy consumption of SCP-MAC was compared with respect to that of a simple LPL protocol by placing 10 nodes, all within range of each other, forming a single hop mesh. Each node periodically generated a 40 Bytes data message (not including preamble) to be sent in broadcast into the network at several transmission rates in order to study MAC performance.

3. PROPOSED MAC SCHEDULER

The basic idea of the proposed scheduler is that nodes wake up periodically to transmit packets, and at the scheduled times to receive messages from their neighbors. To solve the hidden node problem, a node validates the awakening times announced by its neighbors and alerts them of possible collisions. When the radio is on, a node either transmits data and receives acknowledgment messages (ACKs), or receives data from other nodes and sends ACKs. Before describing the scheduler, some defined parameters are introduced:

- T_0 is the time interval (in seconds) between two subsequent transmissions. It is the same for every node and it is preconfigured.
- *WakeTime* is the time interval (in seconds) in which a node can transmit the local buffered data or receive data from its neighbors.
- *Announce Packet* (Pkt_{ANN}) is a signaling packet used by each node to advertise its presence and it contains the time interval between the transmission time of the packet and the next awakening time for transmission chosen by the node.
- *Alert Packet* (Pkt_{ALERT}) is a signaling packet used by a node to alert a neighbor of a possible collision.
- *Full Packet* (Pkt_{FULL}) is a signaling packet used by a node to inform its neighbors that it is out of the network.
- *Wake-up Table* (W_{TBL}) is a table used by each node to store information about the awakening time for transmission of its neighbors. Each table entry is associated with one of the neighbors and contains the following information: (a) the ID of the neighbor, (b) the offset of awakening time and (c) the number of cycles of length T_0 during which no data have been received from the corresponding node.

In the following the start-up phase and the periodic listening and sleep phase are described.

3.1 Network Start-up

During the initialization phase of the network all nodes remain active for the period of time necessary to setup the network parameters and exchange information about their transmission time by sending a Pkt_{ANN}. On the reception of such a message from a new node, a node validates the information received: it checks if the time interval declared by the new node does not overlap with the transmission intervals already scheduled by its neighbors. In this case, the node updates its W_{TBL} by inserting a new entry. The offset stored in the entry is obtained by subtracting an appropriate time interval, which takes into account the processing, transmission and propagation time of the packet, from the time announced by the new node. The entries in the table are in increasing order of the offset. Otherwise, the node sends a Pkt_{ALERT} to the new node containing the transmission time that overlaps with the time announced. This packet has been introduced to greatly reduce the hidden node problem that is one of the main problems that afflict ad hoc networks. By this way, it is possible avoid collisions at a common neighbor by nodes two hops away. The new node inserts this information into its W_{TBL} and chooses a new transmission time. If the network is full and the new node cannot find a valid transmission time, it communicates this information to its neighbors by sending a Pkt_{FULL} and turns off the radio. On the reception of such a message, nodes delete the relative entry from their W_{TBL}.

Note that the time is divided into fixed slot time with duration T_0. Therefore, a node must choose its transmission time within this period. In particular, each node chooses its own transmission time as a random value in a proper interval, also taking into account the choices done by its neighbors. This time differentiation leads to a reduced channel access contention. If the W_{TBL} is empty, then this value is randomly selected in the interval

$$[0, T_0 - (WakeTime + 2 * TurnAroundTime)] \qquad (1)$$

where *WakeTime* is the transmission time and *TurnAroundTime* is the amount of time the radio needs to change its state. If the W_{TBL} is not empty, then every node tries to set its transmission time to a value different from those of its neighbors in order to avoid collisions due to simultaneous transmissions. The value is chosen so that the time interval reserved for the transmission (*WakeTime*) does not overlap with the transmission time of any neighbor. The node chooses the two consecutive entries in the table, *i* and *i+1*, whose offsets' difference is maximum and check if this difference is greater than

$$2 * WakeTime + 4 * TurnAroundTime \qquad (2)$$

Note that the node also checks the time intervals:

$$[0, offset[0]] \quad and \quad [offset[n], T_0 - D] \qquad (3)$$

where *offset[0]* and *offset[n]* are the offsets associated to the first and last entry respectively, while *D = WakeTime + 2*TurnAroundTime*. If this is the case, then the transmission time is chosen within the interval

$$[offset[i] + D, offset[i+1] - D] \qquad (4)$$

where *offset[i]* and *offset[i+1]* are the offsets associated to entry *i* and *i+1* respectively.

In order to maximize the probability that all its neighbors receive the message, a node sends the Pkt_{ANN} three times. After this announcement phase, if it has not received any Pkt_{ALERT}, the node inserts a new entry in the W_{TBL} containing its own address and the scheduled transmission time.

3.2 Steady State

After the warm-up period, a node performs the periodic listening and sleep phase. The most important events in this phase are: the periodic wake up for a transmission, the scheduled wake up for the reception of packets from a neighbor, and the arrival of a new node into the WSN.

The activity of each node is ruled by its W_{TBL}. For every scheduled event the node switches on its radio, handles the correspondent event and then switches off the transceiver. In this phase, there are only two kinds of events to handle: the transmission of a buffered packet (every T_0 seconds) and the receiving of packets from a neighbor that has a scheduled transmission. The node sets a timer for the next scheduled event in its W_{TBL}. When the timer expires, if the event scheduled is a data transmission (the entry contains its own address) then the node wakes up, checks the presence of packets to be transmitted in its queue and, if there is any, starts the transmission. As the transmission ends, the node waits for an ACK from the intended receiver. If this is not received, then the message must be sent again. At the end of its transmission interval, the node schedules the next event and switches off its radio. Otherwise, if the event scheduled is a data reception (the entry contains the address of a neighbor), the node switches on its radio and starts listening the channel. If a packet arrives, it sends an ACK, and at the end of the receiving interval, it switches off the radio component. If nothing is received, then the node updates the counter in the corresponding W_{TBL} entry about the number of times that no packets have been received from that sender. After a fixed number *m, m > 0*, of consecutive failed receptions, the entry is removed so as to avoid useless awakenings.

When a new node joins the network, it first listens to the channel for an interval of time equal to $2*\text{T}_0$ with the aim of detecting the transmissions of its current neighbors. For each packet received from an unknown node it adds an entry for the sender to its W_{TBL}. After that, it announces its presence by sending a Pkt_{ANN} once for each neighbor. Let's observe that a node delays the sending of its data packets of a period of time δ, respect to its transmission interval. This time interval is used by the new node to transmit its Pkt_{ANN}s.

4. TEST BED ENVIRONMENT

The functionality of the MAC scheduler previously described has been evaluated by means of two real test beds deployed at the IDA Lab of University of Salento in Lecce and at the Real-Time Systems Laboratory of Scuola Superiore SantAnna in Pisa. The choice of using different hardware devices permits to analyze the portability of the scheduler as a function of devices characteristics (e.g., clock speed, memory) and architectures (e.g., ARM, MIPS).

In the following, the two selected hardware platforms are

presented before to discuss the MAC scheduler implementation in Contiki OS.

4.1 Hardware platforms

The first device selected to evaluate the scheduler performance is the MB851 (Figure 1a) developed by ST Microelectronics. The board is equipped with a 32-bit ARM® Cortex™-M3 microcontroller operating at a clock frequency up to 24 MHz and embedding 8 Kbytes of RAM and 64 Kbytes of eFlash as ROM. Moreover the board integrates a 2.4 GHz wireless transceiver compliant with the IEEE802.15.4 standard, and providing in hardware MAC functionality. The MB851 board has been mainly selected due to the mounted microcontroller that is highly optimized to guarantee high performance at a very low power consumption.

The second board selected for the test bed is the Seedeye (Figure 1b) developed by Scuola Superiore Sant'Anna and Evidence S.r.l.. The board mounts a 32-bit Microchip™ PIC32MX795F512L microcontroller based on the MIPS architecture and able to reach a maximum clock speed of 80 MHz. The microcontroller is characterized by 128 Kbyte of RAM and 512 Kbyte of ROM. Wireless communication capabilities are provided by the Microchip™ MRF24J40MB transceiver which is fully compliant with the IEEE 802.15.4 standard, and characterized by a transmission frequency of 2.4 GHz with a transmission power up to +20 dBm.

The main differences in microcontrollers architecture, available device memory and transceiver capabilities permit to validate the developed solution on both platforms.

4.2 MAC scheduler implementation

The developed MAC scheduler has been implemented as additional module of the Contiki OS communication stack. Contiki is a popular open-source OS targeted to small microcontroller architectures. The developed communication stack is organized in several layers in which both protocol solutions and device functionality can be easily configured.

The first layer, from bottom to up, is the *NETSTACK_CONF_FRAMER* that is in charge of the data packet format conversion before the transmission over the physical channel. The layer above, the *NETSTACK_CONF_RADIO*, directly manage the wireless transceivers functionality through the appropriate device driver. These two first levels can be considered the PHY Layer of the ISO/OSI network model. The third layer of the stack is the *NETSTACK_CONF_RDC* and has not a direct correspondence with the ISO/OSI model. It is just below the MAC Layer, identified as *NETSTACK_CONF_MAC*, and it is in charge of managing the radio duty-cycling to provide energy saving capabilities. Last layer of the stack is the *NETSTACK_CONF_NETWORK* providing IPv6 functionality.

Considering the Contiki communication stack architecture the proposed MAC scheduler has been developed as additional module of the NETSTACK_CONF_RDC layer. From an implementation point of view the new RDC module exposes Contiki based APIs to the upper layer while using the ones provided by the radio drivers. In order to develop a software solution to be used on several architectures the scheduler timing has been implemented completely independent from the system clock, while making possible to configure at compilation time the transmission buffer dimension. In the implementation phase a very big issue encountered was on the transceiver timing in switching from the active to the sleep state. We observe that a real transceiver device has to switch from one state to another. This switching needs to satisfy stringent delay constraints, which, if they are not respected in the driver implementation, could cause a block of the transceiver, thus making impossible further wireless communications.

5. RESULTS

In order to evaluate the behavior of the proposed scheduler in a timing-accurate and controlled environment, several simulation campaigns by using Contiki's simulation tools were carried out. They consist of the Cooja network simulator and the MSPsim device emulator. This environment combines a cycle-accurate simulation of the Tmote Sky platform with a bit-level accurate simulation of its CC2420 radio transceiver.

The network topology considered is a 16-node grid topology where each node can have at most 4 neighbors. All nodes run Contiki and the Contiki Collect protocol [16], an address-free data collection protocol that builds a tree rooted in one or more sinks, towards which packets are routed.

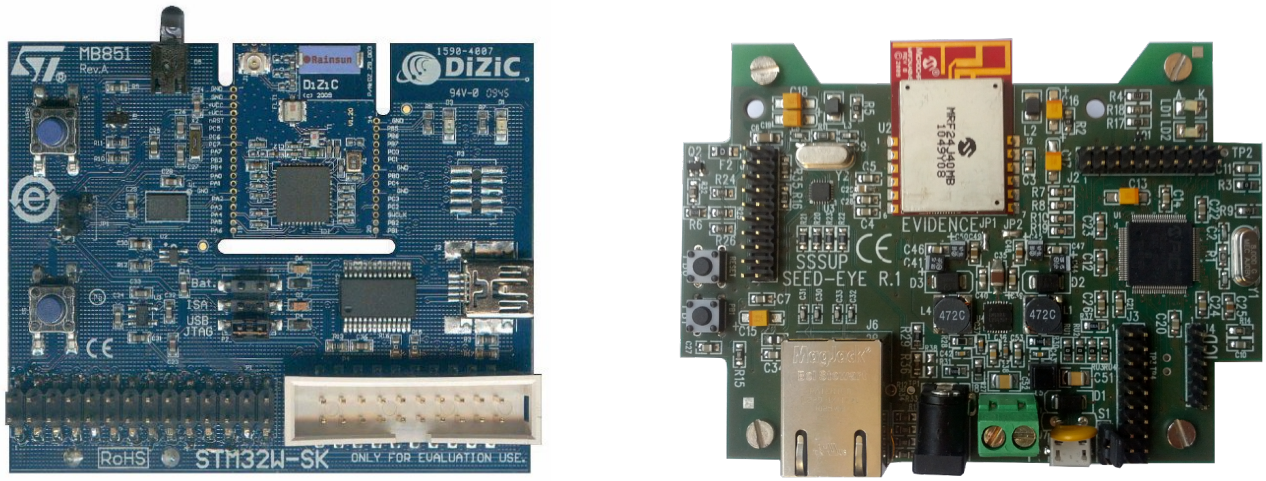
In all simulations, nodes send data packet towards the sink at a constant packet rate (CPR) of one packet per minute, a typical data rate used in sensor networks [17]. Every transmission is sent with 31 hop-by-hop retransmissions. Each node sends 100 packets towards the sink. The proposed MAC scheduler was compared with the MAC protocol adopted by the current version of ZigBee and the X-MAC duty cycling protocol. To ensure a meaningful comparison between our scheme and the X-MAC protocol, the values of T_0 and *WakeTime* of the proposed solution and the channel check rate of the X-MAC protocol have been chosen so as to ensure a successfully transfer of all packets to the sink. The main simulation parameters are reported in Table 1.

Table 1- Simulation Parameters

Parameter	**Value**
Network Topology	Grid Network
Number of nodes	16
Rate	1 packet per minute
Number of packets	100
T_0	5 s
WakeTime	50 ms

We measured the power consumption with Contiki's Powertrace tool [18]. The results, characterized by a 95% confidence interval with a 5% maximum relative error, are reported in Figure 2. It shows the power consumption versus the distance between sender and receiver (in number of hops).

The curves clearly show that the proposed scheduler reaches



(a) MB851 board *(b) Seed-eye board*

Figure 1- Evaluation boards

a substantial reduction of the power spent by each node with respect to the ZigBee standard protocol, as expected since idle listening and collisions are avoided. It also shows a reduction of power consumption with respect to the X-MAC protocol. Since the idle power consumption is the dominating factor of the system power consumption, there is not a significant difference between nodes closer to the sink, that forward messages generated by others too, and nodes further away.

We also tested the proposed algorithm on real devices to demonstrate its actual functionality and to validate our implementation. Specifically we performed a 1-hour-long experiment, using 5 Seed-Eye boards exchanging more than 5000 messages, to prove that messages are indeed sent during the scheduled intervals. Nodes were configured in a star topology in which the 4 peripheral nodes periodically send data to the central sink. The sending period and the other experiment parameters are reported in Table 2. For every period and node, we measured the time Δ_{RX} elapsed from the switching on of the sink radio (i.e., the beginning of the scheduled reception event) and the actual reception of the message (i.e., the actual event occurrence).

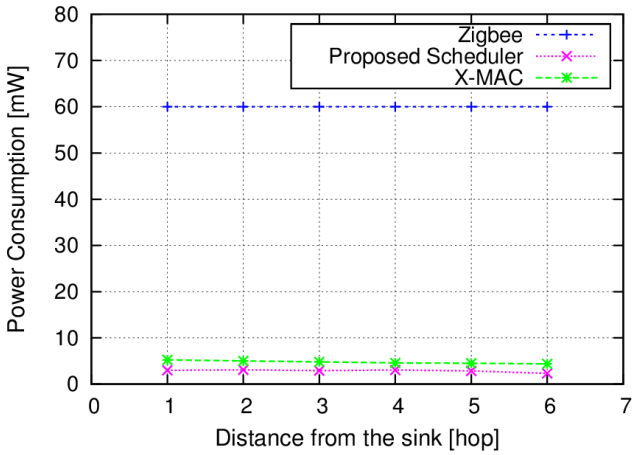


Figure 2- Performance comparison of the proposed scheduler with X-MAC and ZigBee standard in terms of power consumption

Table 2- Experiment Parameters

Parameter	**Value**
Network Topology	Star Network
Number of nodes	5
Sending Period	3 s
Number of packets	5568
T_0	3 s
WakeTime	150 ms
Sending delay	60 ms

Results, reported in Figure 3 using a box-plot, show that all four sending nodes had statistically similar Δ_{RX}s and that no Δ_{RX} was negative or greater than the wakeup time, meaning that no messages were sent too early or too late. Moreover the average value of Δ_{RX} is 60.49 ± 0.05 ms (99% confidence level) which, as expected, is close to the configured sending delay δ. A similar experiment has demonstrated the correct functionality of the protocol on MB851 boards.

6. CONCLUSION

This paper extended the energy-efficient MAC scheduler for WSNs introduced in [2], discussed its implementation in the

Contiki OS, and evaluated such implementation with simulations and real test beds.

The main improvement over the previous work is the introduction of the *Alert Packet* which greatly reduces the hidden node problem. Moreover, the implementation in the Contiki OS and its functional validation prove the feasibility and correctness of the proposed scheduler. The implementation was also used to perform a more realistic performance evaluation using the Contiki's simulation tools. Results show that, in terms of power consumption, the proposed scheduler clearly outperforms the MAC protocol used by ZigBee, and performs slightly better than the X-MAC duty cycling protocol. The performance evaluation on real devices of the proposed solution will characterize future works.

REFERENCES

[1] J. Hill and D. Culler, "Mica: a wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, November 2002.

[2] L. Anchora, A. Capone, and L. Patrono, "An asynchronous scheduler to minimize energy consumption in wireless sensor networks," in *Proceedings of 11th International Conference and 4th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking*, St. Petersburg, Russia, August 2011.

[3] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[4] "ZigBee Alliance," ZigBee Specification Document 053474r17.

[5] M. Buettner, Ga. V. Yee, E.Anderson, and R. Han, "XMAC: A Short Preamble MAC Protocol for DutyCycled Wireless Sensor Networks," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Boulder, Colorado, USA, November 2006, pp. 307–320.

[6] "MB851 User Manual," http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/USER_MANUAL/CD00262415.pdf.

[7] B. Dal Seno, M. Ghibaudi, and C. Scordino, "Embedded boards for traffic monitoring," in *Poster session of the IPERMOB project final workshop*, www.ipermob.org/files/DemoSAT/afternoon/2011-05-18_poster_hw_oo3.pdf, May 2011.

[8] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Baltimore, Maryland, USA, November 2004, pp. 95–107.

[9] C. Enz A. El-Hoiydi J.D. Decotignie and V. Peiris, "WiseNET: An Ultralow-Power Wireless Sensor Network Solution," *IEEE Computer*, vol. 37, no. 8, 2004.

[10] C.J. Merlin and W.B. Heinzelman, "Duty Cycle Control for Low-Power-Listening MAC Protocols," *IEEE Transactions on Mobile Computing*, vol. 9, no. 11, 2010.

[11] J. Heidemann W. Ye and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, June 2004.

[12] T. Van Dam and K. Lengendoen, "An adaptive energy efficient MAC protocol for wireless sensor networks," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Los Angeles, California, USA, November 2003, pp. 171–180.

[13] W. Ye, F. Silva, and J. Heidemann, "Ultra-Low Duty Cycle MAC with Scheduled Channel Polling," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Boulder, Colorado, USA, November 2006, pp. 321–334.

[14] B. Jang, J.B Lim, and M.L. Sichitiu, "AS-MAC: An asynchronous scheduled MAC protocol for wireless sensor networks," in *Proceedings of 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Atlanta, Georgia, USA, September 2008, pp. 434–441.

[15] J. Hill, R. Szewczyk, A.Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of 9th International Conference in Architectural Support for Programming Languages and Operating Systems*, Cambridge, Massachusetts, USA, November 2000, pp. 93–104.

[16] A. Dunkels, F. Osterlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Sydney, Australia, November 2007, pp. 335–349.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of International Conference on Embedded Networked Sensor Systems*, Berkely, California, USA, November 2009, pp. 1–14.

[18] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," in *Technical Report T2011:05, Swedish Institute of Computer Science*, March 2011.

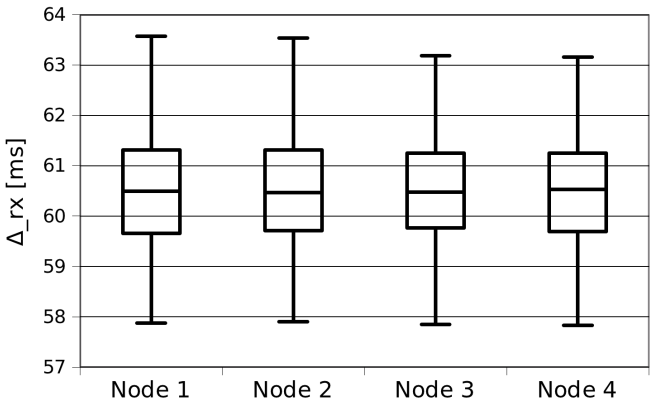


Figure 3- Functional validation