Flexible Prototyping for Ad Hoc Wireless Sensor Network Protocols

Juan P. Leal Licudis[1], Juan C. Abdala[1], Guillermo G. Riva[2], and Jorge M. Finochietto[1,3]

[1] Universidad Nacional de Córdoba, Córdoba
[2] Universidad Tecnológica Nacional, Córdoba
[3] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Córdoba
leal.licudis@gmail.com, jcabdala@ieee.org, griva@scdt.frc.utn.edu.ar, jfinochietto@efn.uncor.edu

Abstract. The development of sofisticated energy-efficient protocols and the increasing complexity of applications in wireless sensor networks (WSNs) imposes the use of open and flexible programming architectures that enable access to the communication stack in a simple way. Nowadays, researchers in WSNs focus on not only the development of efficient mechanisms in application level but also the interaction with the communication stack, in order to improve the performance. Because radio communication is the most energy consuming component of a sensor node, the main challenge in WSNs is to reduce the communication cost by means of efficient in-network distributed processing.
In this work, a probabilistic query routing mechanism is designed and implemented in a WSN, and a study of different operating systems (OS) and communication stacks available for WSNs is analyzed. This implementation enables a flexible prototyping of novel mechanisms by using light communication protocols over a versatile operating system.

Keywords: Wireless Sensor Networks, Embedded Operating System, Communication Stack, In-network Processing, Probabilistic Routing

1 Introduction

A WSN consists of autonomous battery-powered sensor nodes which can sense physical parameters of the environment (e.g. temperature, humidity, light intensity, movement, audio, etc) and send this information to a sink or coordinator node through multi-hop wireless communication. Sensor nodes are deployed over an area where a specific parameter must be monitored. Applications of WSNs range from structure monitoring (buildings, bridges, etc), animal monitoring, patient and athlete monitoring, home automation, logistic, etc. Sensor nodes have limited computation, storage and communication capabilities. The energy cost of transmitting 1 bit is approximately equivalent to compute 2000 code instructions. Since communication cost is much higher than computation one in terms of energy, it is preferred to implement in-networks processing tasks to reduce

the message exchange [1]. The constrained resources of sensor nodes imposes computationally simple and efficient tasks.

Actually, a lot of works in WSNs are based on the design of efficient mechanisms, algorithms and protocols validated by simulation (e.g. by using Omnet++, NS2, etc), only a few works consider the development of proofs-of-concept to show the real network behaviour. Therefore, many implementations in WSNs consider the use of parameters of lower layers in order to improve the network functionality (such as cross-layer optimization). In this context, our work considers the case of using open communication stacks to enable the optimization of different applications and protocols. Two kind of situations can be considered to requiere an open and flexible communication stack. The first one is the need to use low layer parameters in order to optimize the performance of the application. Example of this is the case of adjusting the radio power level, or changing the routing protocol from broadcast to unicast on demand. The second one consider the case of testing new data-centric communications protocols.

Therefore, actual trends in WSNs impose the use of heterogenous networks, in which different kinds of technologies can be interconnected. For this reason, the development of applications based on dedicated firmware given by WSN suppliers is appropriate. Because of this, embedded operating system with multiplatform support is the best option.

The rest of the paper is organized as follows. Section II discusses related work on embedded operating systems for sensor networks. The description of the operating system and communication stack used in this work is given in Section III. Section IV describes a case study where a probabilistic query routing is designed and implemented. Finally, Section VI discusses main conclusions and future work.

2 Related work

Nowadays, both free and comercial embedded operating systems are available, many of which are suitable for energy-constrained WSNs. Examples of open OSs are Contiki OS [2], MantisOS [3], TinyOS [4], etc.

We can classify OSs based on its architecture, execution model, etc. The architecture can be classified in monolithic (e.g. TinyOS), where all source code is compiled into one logically undivided block of object code, and in modular (e.g. MantisOS and Contiki OS). Respect to the execution model, three different types are currently considered, event driven (ie. TinyOS), multithread driven (e.g. MantisOS), and hybrid (e.g. Contiki OS). In the first one, every action is triggered by event (ie. a timer interrupt, an interrupt indicating incoming packets or new sensor reading). In the second one, OS multiplexes execution time between different tasks implemented as threads. While switching between threads, the current context has to be saved and the new context must be restored. In the third one, advantages of event-based and thread-based OSs are considered.

TinyOS is a free and open source component-based operating system developed since 2000. TinyOS started as a collaboration between the University of

California, Berkeley in co-operation with Intel Research and Crossbow Technology, and has since grown to be an international consortium, the TinyOS Alliance. Even if both good technical support and multiplatform support are provided, the main disadvantage of using this OS is that core and applications are written in nesC programming language [5], which is a dialect of C language optimized for memory-limited devices, as a set of cooperating tasks and processes. NesC programming imposes difficulties in understanding how parameterized interfaces can be used, and how to manage their keyspaces. Therefore, the incorporation of tasks into the nesC language is very difficult, and its learning curve is slow for a normal developer.

MantisOS is a thread-driven operating system model for sensor networks. MantisOS suffers from the overheads of context switching and the memory allocated (in the form of stack) per each thread. This overhead is signifficant in resource constrained systems like WSNs.

Contiki OS is a small, free and open source, highly portable, hybrid operating system developed for use on a number of memory-constrained networked systems. Its execution model takes the advantage of event-based and thread-based OSs [6]. Contiki include two communication stacks Rime and uIP. The first one is a lightweight, layered communication stack which is designed to significantly reduce the implementation complexity and simplify the development of communication protocols. The second one enables TCP/UDP transport protocols, and IPv4/IPv6 Intenet protocols. Applications can use either or both stacks, and uIP can run over Rime and viceversa. One of the most atractive characteristic of Contiki is the multiplatform support (such as RZRAVEN, ZigBit, TmoteSky, MicaZ, Sensinode, etc). Therefore, a big community of Contiki's developers works in the improvement of Contiki and in the migration to new platforms.

Most of the OS for WSNs have support for the *IEEE 802.15.4* standard [7], which defines Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs). It targets ultra-low complexity, cost, and power for low-data-rate wireless connectivity among fixed, and mobile devices. Actually, 802.15.4 is considered the most appropriate standard for a WSN.

ZigBee is a specification for a suite of high level communication protocols based on IEEE 802.15.4. It adds a complete networking solution defining upper layers increasing the complexity. Even if the ZigBee specification is freely available for non-comercial purposes, it is commercially licensed.

We propose the implementation of a novel probabilistic query routing mechanism using Contiki OS with Rime communication stack. Although the Rime stack implements sensor network protocols ranging from reliable data collection and best-effort network flooding to multi-hop bulk data transfer and data dissemination, our proposal is builded over the lower layers of Rime. The layered model of Rime communication stack enables flexibility and efficient use of resources.

3 Contiki Operating System

Contiki consists of an event-driven kernel, on top of which application programs can be dynamically loaded and unloaded at run time. A typical Contiki configuration consumes 2 kilobytes of RAM and 40 kilobytes of ROM. In this OS, the processes use lightweight protothreads that provide a linear, threadlike programming style on top of the event-driven kernel. Contiki also supports per-process optional multithreading and interprocess communication using message passing. Two communication stacks are provided in Contiki, a full IP-networking stack named uIP [8], and a low-power layered communication stack named Rime [9]. The first one is designed to support both IPv4 and IPv6 (using 6LowPan abstraction layer [14]) standards. A comparison between the main characteristics of both stacks is detailed in Table 1.

Table 1. Comparison of Contiki's communication stack

Characteristic \ Stack	Rime	uIP
Complexity	low	high
Code Size	low	high
Energy Consumption	low	high
Funcionality	high	high
Documentation	less	more

Contiki also implements an adaptative communication architecture by mean of the Chamaleon adaptation stack which enables the use of heterogeneous networks described in section 3.2.

3.1 Rime Communication Stack

Rime is a light and layered communication stack for sensor networks which is designed to significantly reduce the implementation complexity and simplify the development of communication protocols facilitating code reuse. Rime is implemented in Contiki OS, with a code size less than two kilobytes and data memory requirements on the order of tens of bytes. The layers are designed to be extremely simple, both in terms of interface and implementation. The lowest level primitive in Rime is anonymous best effort broadcast (abc). The abc layer provides a 16-bit channel abstraction but no node addressing; it is added by upper layers. The identified sender best-effort broadcast (ibc or broadcast), which is the smallest module with a code size of 100 bytes, adds a sender identity header field. One of the more important features of Rime is to shift the burden of memory use from protocol implementations to Rime core. By making Rime part of Contikis system core, which is always present in memory, loadable programs are made smaller. Consequently, the energy consumption for program loading

is reduced. To reduce memory requirement, Rime uses a single buffer for both incoming and outgoing packets. Layers that need to queue data, copy the data to dynamically allocated queue buffers. Figure 1 shows the Rime architecture, in which many channels between interconnected nodes can coexist. The reference [9] gives a detailed description of the functionality of each Rime layer.

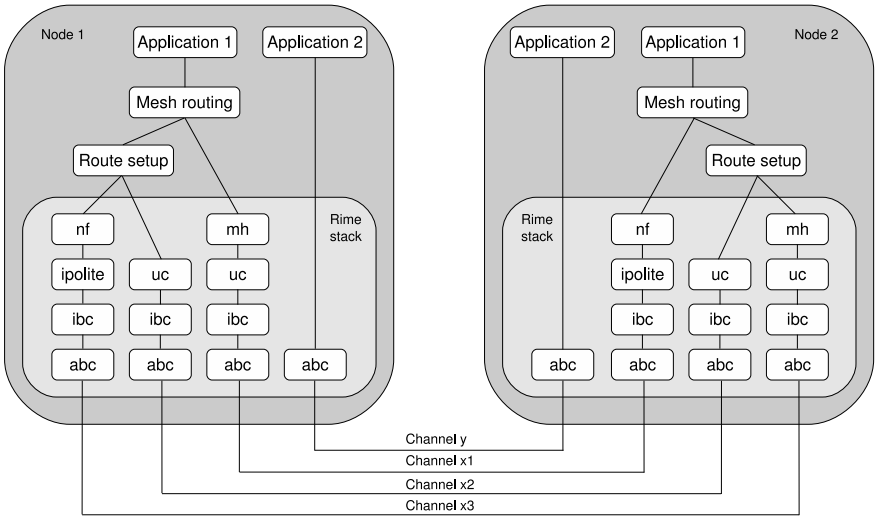


Fig. 1. Contiki stack diagram.

3.2 Chameleon Adaptation Layer

Chameleon is an adaptative communication architecture for sensor networks. It consists of a set of packet transformation modules which adapts frames comming from Rime stack to a standard communication frame. It is designed to be able to adapt to a variety of different underlying protocols and mechanisms while being expressive enough to accommodate typical sensor network protocols.

One of the main problems of specifying an interoperable communication architecture is finding a universal header format. Such a header format must be both expressive enough to encompass all communication patterns supported by the architecture and flexible enough to allow for future expansion of the architecture. For a WSN architecture, the problem is even more challenging, as the header format must be small enough to be efficient over low-power radio links with small maximum packet sizes. Chameleon takes a different approach to the problem of finding a common packet header format. It does not define any packet headers at all. Rather, it uses packet attributes, an abstract representation of the information usually found in packet headers. Packet headers are produced by separate header transformation modules that transform application

data and packet attributes into packets with headers and payload. By using different Chameleon modules, it is possible to create packets that conform to any given packet header specification. Unlike other communication architectures, the application running on top of Chameleon does not need to be changed to make it run over those different underlying communication mechanisms. The use of packet attributes makes it possible to adapt the output from the protocol stack to other communication protocols such as link and MAC layer protocols and TCP/IP.

4 Case Study: Probabilistic Query Routing

4.1 Motivation

The most used mechanism to obtain information from WSNs is the pull-based approach. In this case, the coordinator node injects data requirements in the network, and sensor nodes propagate those queries through multihop communication, flooding the network. This process is normally known as query diffussion or dissemination. After that, all nodes begin to answer to the coordinator. This last process is named data collection phase. When coordinator receives all answers, it can obtain different kinds of data statistics. This approach is unapropiate for large sensor networks, where a lot of energy is expended in data communication. In most recent approaches, in-network data processing techniques are applied in data collection phase, such as data aggregation, data fusion, etc. With these mechanisms, a big reduction of the energy consumption is obtained. However, some applications enable the possibility to make more efficient the query diffusion reducing the search space in order to save more energy and to prolong the network lifetime. Applications that are included in this group are for example to determine which nodes have readings in a specific range, the maximum/minimum reading, etc. For this case, the query diffussion mechanism can be improved by using probabilistic data-centric routing. The data-centric approach enables to improve the routing mechanism by making use of the sensed data in queried nodes. The only restriction is that functions implemented in nodes must be simple and energy reduced, because the constrained energy in WSNs. This last application motivates us to develop an efficient query routing mechanism. In section 4.3 a detailed description of this mechanism is given.

4.2 Network Model

We consider a flat and unstructured WSNs of uniformly scattered sensor nodes over a square geographic area. A sink or coordinator node injects query messages that are routed through the network to search relevant data. In flat networks all nodes have the same functionality. In unstructured networks sink node has no knownledge where the target resides, it uses blind sequential search for querying. Nodes can exchange messages with all neighbor nodes by using broadcast transmission mechanism within a fixed circular communication range. The coordinator node is connected to a PC through USB port in order to analyze the responses.

4.3 Probabilistic Query Routing Description

We describe the case of searching the maximal value sensed in the network. The coordinator node sends query messages to nodes requiering information. When each sensor node receives the query message, it compare the query value with its reading and determines if it has a significant value. In this sense, each node can take three different desitions:

• If its reading is bigger than the value propagated in query message, the node is set to answer to the coordinator after a periode of time, and forward the query message to its neighboors updated with its value (1).

• If its reading is lower than the value propagated in query message, the node is not set to answer the query, and it computes the probability to continue forwarding the message with the original value. This desition is based on the comparison between a number which is dependent of the difference between sensor reading and the value in query message and a parameter named T, and a random number. If the random number is lower than the first one, the query message is forwarded but with the original value (2). If the random number is higher than the first one, the message is dropped and node does not continue forwarding the message (3).

The parameter of control of this probabilistic mechanism is the T value. With high T values, the proposed mechanism performs as flooding, the query error is zero but the energy cost is high. In the other case, with low T values, this mechanism performs as hill climbing algorithm, where the query propagation continues while improvements in the solution take place. In this last condition, a low energy cost is requiered to fetch data from relevant nodes, but the query error is high. There is a compromise between query error and query cost that must be considered based on the specific application of the sensor networks.

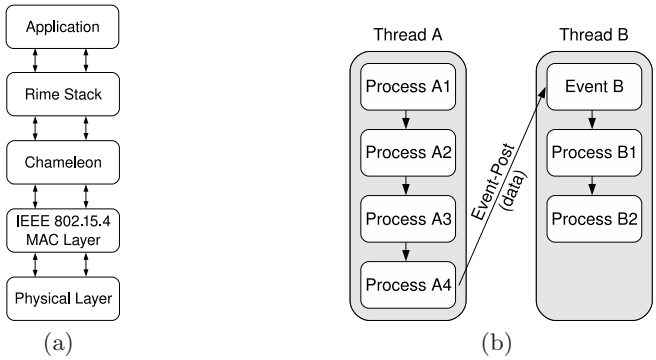
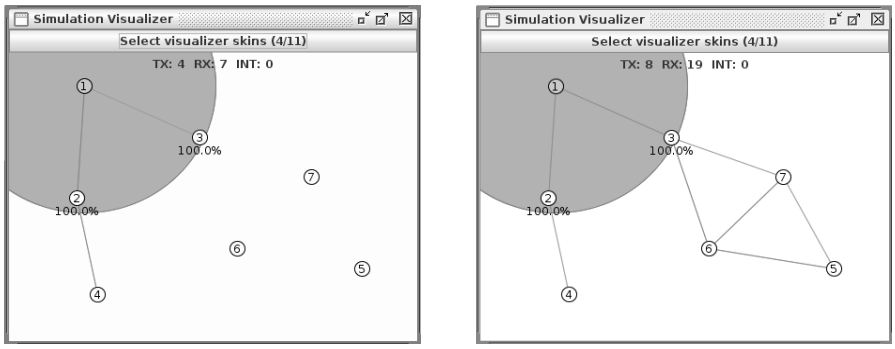


Fig. 2. Contiki stack (a), Threads implementation (b).

The query routing mechanism was implemented using two threads as is showed in Figure 2. This application takes advantage of using the lower layer of the Rime stack (abc), such as Application 2 in Figure 1.

4.4 Simulation

In order to evaluate the performance of the query routing mechanism, we use Cooja cross-level network simulator [10]. Cooja is an open discrete event simulation package for the Contiki operating system that enable simultaneous simulation at diferent levels, such as networking level, operating system level and machine code instruction set level. Cooja is implemented in Java, but allows software to be written in C by using Java Native Interface. Models of different kinds of sensor nodes are included in the simulator (ie MicaZ, SkyMote, and ESB), in order to obtain realistic simulations. Different configuration parameters are available in this simulator, such as radio coverage, data traffic, etc. A time line tracks the state of each radio transceiver, through which it is possible to determine the power consumption in the network. Figure 3 shows the network behaviour using the proposed mechanism in cases of low and high T values. We can see that the routing mechanism is more aggresive in the last case than in the first one.



(a) (b)

Fig. 3. Simulation in Cooja environment. Behaviour of the proposed mechanism with (a) low T value, and (b) high T value.

4.5 Implementation

In order to evaluate the behaviour of the proposed mechanism in a real scenario, it was implemented on Atmel RZRAVEN sensor nodes [11] [12], as is showed in Figure 4. The network is conformed by one USBRAVEN connected to the PC through USB interface as coordinator, and six AVRRAVEN sensor nodes. The first one makes use of a AT90USB1287 8-bits low-power microcontroller. The second ones make use of two 8-bits low-power microcontrollers (ATmega1284P with 128 Kbytes of flash and 16 Kbytes of SRAM and ATmega3290P with 32 Kbytes of flash and 2 Kbytes of SRAM) interconnected between them by a serial interface. The ATmega1284P is connected to the radio transceiver, and the

ATmega3290P has the function to manage the sensors and optionally to control the display. The AT86RF230 low-power 2.4GHz radio transceiver [13] for IEEE802.15.4-2003 applications is used both in coordinator node and in sensor nodes. Devices availables in these sensor nodes are temperature sensor, microphone, and speaker. In the actual implementation, only temperature sensors readings are considered as data source.

Fig. 4. Implementation of probabilistic query routing using RZRAVEN sensor nodes.

In table 2, a comparison of the memory requierement using Rime and uIP with UDP/IPv6 for the same application of a probabilistic routing mechanism in sensor nodes is shown.

Table 2. Memory use for the application using Rime and uIP comm stacks

Stack / Memory (bytes)	Rime	uIP & IPv6
Program	20728 (15.8%)	37416 (28.5%)
Data	1776 (10.8%)	4076 (24.8%)
EEPROM	8 (0.2%)	8 (0.2%)

In order to capture and analyze the data traffic in the network, a coordinator node hearing the wireless channel and sending the packets through USB to a PC executing Wireshark network analyzer was used.

4.6 Energy Consumption Analysis

The most energy consumption in WSNs resides in the radio communication. In this way, we want to compare the benefits by implementing our mechanism using Rime stack respect to the a IPv6-enabled implementation. To quantize the energy saving by using the different kind of communication protocols, an analysis of the energy consumption requiered to transmit until 10000 packets was taking in account. Figure 5 shows the energy consumption using IPv6, Rime in the actual implementation, and Rime in a future implementation (implementing bit code optimization). The exact energy consumption values of sending 10000 packets are 1.10 J for IPv6, 0.89 J for Rime, and 0.82 J for Rime optimized. This parameters are estimated considering the maximal power transmission (+3 dBm) and maximal baudrate (250 kBps) of the AT90RF230 radio transceiver [15].

Normally, low-power microcontrollers implemented in WSNs approximately requiere 1 nJ per instruction. The diference in energy consumption between using IPv6 and Rime in the actual implementation is equivalent to compute 210 millon of instructions.

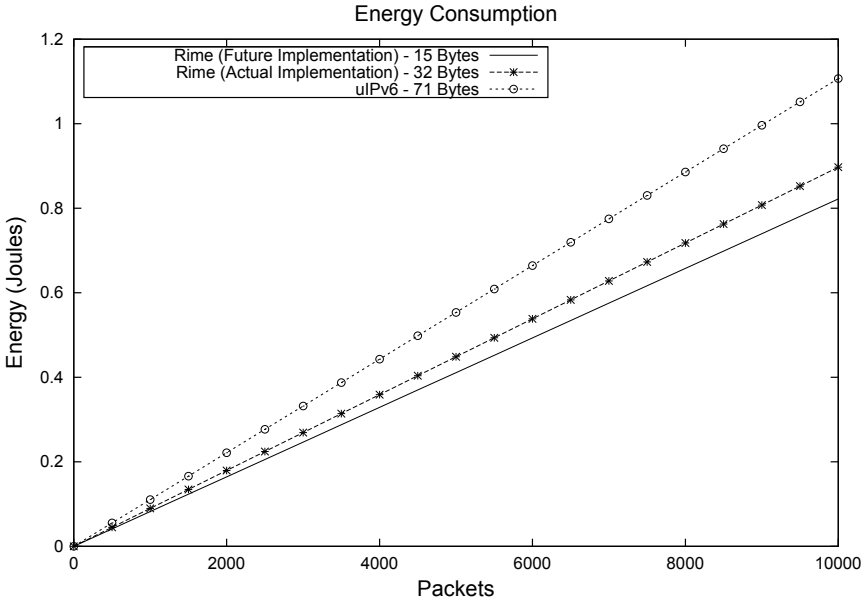


Fig. 5. Energy consumption by using Rime and IPv6.

5 Conclusion and future work

In this paper, we proposed a flexible and energy-efficient form to implement and validate new in-network processing mechanisms and communication protocols

in WSNs, such as the proposed probabilistic query routing mechanism. As a future work, we want to increase the network size and implement over the air programming in order to disseminate the code fastly.

Acknowledgment This work is partially funded by the Universidad Tecnologica Nacional - FONCyT IP-PRH 2007 Posgraduate Grant Program and by the SECYT-UNC 2010-2011 Research Program.

References

1. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy Conservation in Wireless Sensor Networks: A Survey. Ad Hoc Networks 7, pp. 537-568 (2009)
2. Dunkels, A., Groenvall, B., Voigt, T.: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, IEEE Computer Society Washington, DC, USA (2004)
3. Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., Han, R.: MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. ACM/Kluwer Mobile Networks & Applications, Special Issue on Wireless Sensor Networks, Vol. 10, No. 4 (2005)
4. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System Architecture Directions for Networked Sensors. In Architectural Support for Programming Languages and Operating Systems (2000)
5. Gay, D., Welsh, M., Levis, P., Brewer, E., Von Behren, R., Culler, D.: The nesC language: A holistic approach to networked embedded systems. In Proceedings of the ACM SIGPLAN 2003, Conference on Programming Language Design and Implementation, USA (2003)
6. Dunkels, A., Schmidt, O., Voigt, T., Ali, M.: Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems. In Proceedings of the Fourth ACM Conference on Networked Embedded Sensor Systems, USA (2006)
7. IEEE Computer Society: IEEE Standard 802.15.4, IEEE Standard for Information technology, Part 15.4; Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (2003)
8. Dunkels, A.: Full TCP/IP for 8-Bit Architectures. In Proceedings of the first international conference on mobile applications, systems and services, USA (2003)
9. Dunkels, A., Oesterlind, F., He, Z.: An Adaptive Communication Architecture for Wireless Sensor Networks. In Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems, Australia (2007)
10. Oesterlind, F., Dunkels, D., Eriksson, J., Finne, N., Voigt, T.: Cross-Level Sensor Network Simulation with COOJA. In Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (2006)
11. Atmel - AVR2015: RZRAVEN Quick Start Guide
12. Atmel - AVR2016: RZRAVEN Hardware User's Guide
13. Atmel - AT86RF230 Datasheet
14. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Network Working Group - RFC 4944 (2007)
15. Atmel - AVR2007: IEEE802.15.4 MAC power consumptions for AT86RF230 and ATmega1281