Soft Error Reliability Prediction of SRAM-based FPGA Designs

An Energy-autonomous Wireless Sensor Network Development Platform

Michelangelo Grosso, Salvatore Rinaudo
STMicroelectronics s.r.l.
Torino, Italy
{michelangelo.grosso, salvatore.rinaudo}@st.com

Edoardo Patti, Andrea Acquaviva
Dipartimento di Automatica e Informatica
Politecnico di Torino – Torino, Italy
{edoardo.patti, andrea.acquaviva}@polito.it

***Abstract*—Internet-of-things enabled applications are increasingly popular and are expected to spread even more in the next few years. Energy efficiency is fundamental to support the widespread use of such systems. This paper presents a practical framework for the development and the evaluation of low-power Wireless Sensor Networks equipped with energy harvesting, aiming at energy-autonomous applications. An experimental case study demonstrates the capabilities of the solution.**

Keywords—wireless sensor network; energy autonomy; energy harvesting; low-power.

I. INTRODUCTION

Thanks to the availability of advanced and low-cost electronic components, and to the ubiquitous connectivity afforded by the internet, Internet-of-Things (IoT) applications are growing in number and the connected devices are expected to exceed 20 billion units by 2023 [1]. Within such applications, internet-enabled wireless sensor networks (WSNs) play an important role in many of today's most promising fields, such as smart home, smart city and smart industry. In our houses, wireless sensor networks can help in optimizing heating and air conditioning control, or contribute to intrusion alarm systems. In the smart city context, WSNs are used, e.g., to monitor traffic and pollution, or to improve the management of public lighting and parking. In the industry, smart connected sensing devices contribute to enhance production efficiency, security, logistics and maintenance planning. WSNs potentially provide a huge quantity of data that become especially valuable when supported by mining techniques and artificial intelligence in the cloud.

A wireless sensor network is based on a set of devices, or *nodes*, each typically constituted of a microcontroller (MCU) and a radio transceiver, and one or more sensing elements. Internet access is usually provided by a special node called *gateway*. Different network architectures are then possible, each characterized by specific memory, computational speed and communications bandwidth: the design and deployment of a wireless sensor network entails weighing trade-offs between costs and performance, in the selection of components and in the management of the application and the network, while taking into account the application requirements. Due to the large number of involved devices and to the characteristics of installation contexts, other critical parameters to be considered are self-configuration and self-reorganization, reliability and operating time allowed by the selected energy source.

The system lifetime (or mean-time-before-maintenance) required by stakeholders is often in the range of several years: along with ultra-low power consumption and efficient batteries, energy harvesting from the environment is becoming a realistic solution to alleviate, if not to solve, the energy problem.

When designing a WSN for a specific application, an extremely wide range of implementation possibilities opens before us. Starting from scratch, with the selection of the node components and the development of custom firmware, requires a significant investment in terms of resources and time, while relying on already available complete systems may not optimally fit the application requirements. The main challenge in the development of an energy-autonomous system consists of matching node power requirements and energy supply potential of the harvester by properly tuning network features and using as much as possible the available low-power capabilities of the node. Keeping under control overheads, delays and latency in a heavy constrained system is not trivial, as well as guaranteeing a high reliability in connections between nodes.

This work presents a flexible framework for the development and the evaluation of ambient monitoring wireless sensor networks with low data-rate using off-the-shelf components. An aggressive but scalable synchronized duty cycling approach is used, with the aim of enabling energy autonomy at the expense of additional latency in the measure transmission, considering that often the parameters to be monitored are slowly changing in time. The composing elements of such framework are two: the former is a set of ultra-low-power devices, including a microcontroller family, a radio transceiver and add-on sensors, defining the general architecture of a node, and an application template relying on the Contiki operating system, supporting a 6LoWPAN mesh network. The latter is a simulation environment enabling the validation of the network application and the optimization of parameters while estimating power consumption and production, and thus operating time, under realistic conditions.

Within this framework, an experimental case study was developed and empirically validated, consisting of an energy-autonomous ambient monitoring network based on devices from the Nucleo ecosystem by STMicroelectronics. This solution provides ultra-low power, low-cost, easily customizable components supported by firmware templates, embedded operating systems, a free integrated development environment and a live community of users.

This paper is organized as follows: Section II presents some background information on low-power wireless sensor network architectures and simulation. Section III describes the proposed framework, while in Section IV the case study is presented with some experimental data. Section V concludes the paper.

II. Background

This Section summarizes general design concepts involved in the development of a WSN, focusing on node and network structure, energy and validation by means of simulation.

A. Wireless Sensor network Design

The design of WSNs requires ample knowledge of a wide variety of research fields including wireless communication, networking, embedded systems, digital signal processing, and software engineering. This is motivated by the close coupling between several hardware and software entities of wireless sensor devices as well as the distributed operation of a network of these devices. Consequently, several factors exist that significantly influence the design of WSNs [3]. Fig. 1 shows the generic overview of a harvesting node divided into the main functional blocks [4]. The functional part includes a microcontroller, a radio transceiver, sensors and actuators. Radio transceivers usually work in the Industrial, Scientific and Medical (ISM) band, with different solutions available depending on bandwidth, power and standardization requirements: e.g., a Wi-Fi-enabled node can be easily used in a home network, but its energy requirements are sizeable. The harvesting module typically includes, in addition to the energy-scavenging device, a power conversion module able to maximize the energy transfer from the environment (such as with Maximum Power Point Tracking – MPPT – techniques) and to regulate the voltage as needed to supply the functional modules and to manage charge/discharge of a buffer (such as a supercapacitor or a battery).

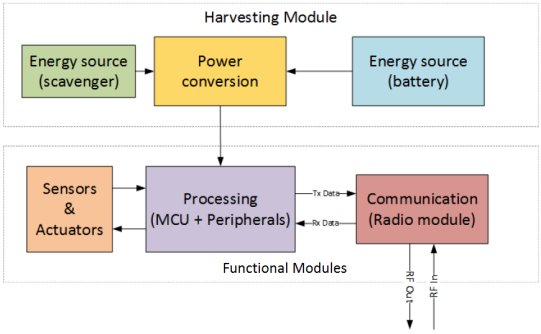


Fig. 1. Typical architecture of a wireless sensor network node

A WSN can assume several network topologies spacing from a star network to a complex multi-hop mesh network. These two topologies have different characteristics and present advantages and disadvantages depending on the features required by the application. Star networks consist of a central, internet-connected node connected to many *leaves* in charge of collecting data. The simple architecture has a main drawback in the limited radio transmission range. A mesh network, conversely, consists of a large number of nodes randomly placed more or less close to each other. As opposed to the star topology, nodes can exchange messages with each other. This allows what is known as multi-hop communications, i.e., if a node wants to send a message to another node that is out of radio reach, it can use intermediate nodes to forward the message to the receiver. Mesh network topology has the advantage of redundancy and scalability: If an individual node fails, a remote node still can communicate to any other node in its range, which, in turn, can forward the message to the desired location. The disadvantage of this type of network relates with the power consumption of the nodes that implement the multi-hop communications, which is generally higher than for the nodes that do not have this capability, often limiting the battery life due to the increased number of messages sent. Additionally, as the number of communication hops to a destination increases, the time to deliver the message also increases, especially if low power operation of the nodes is a requirement.

Regarding the communication protocol, standardization becomes a major issue in order to improve quality and performance and above all reusability and interoperability. In this respect, IEEE 802.15.4 was formed with the aim to provide a standard physical layer and media access control for Wireless Personal Area Networks (WPANs). A WPAN is a computer network used for data transmission among devices with low-range wireless transceiver technology with long battery life and very low complexity. WPAN extension typically varies from a few centimeters to a few meters. For low data-rate networks a more accurate distinction identifies the Low Rate Personal Area Networks (LR-WPANs). IEEE 802.15.4 is extended with upper layers by several other specifications such as ZigBee, ISA100.11a, WirelessHART, MiWi, SNAP, and Thread. It can be used also with 6LoWPAN to deliver the IPv6 version of the Internet Protocol (IP) over WPANs.

A number of operating systems have been developed specifically for WSNs, differing in size, structure and modularity, networking support, multithreading support and real-time performance [5], such as TinyOS, Contiki and LiteOS. The best fitting for one application also depends on the availability of the porting for the selected microcontroller, of simulators and of live user communities to support development.

B. Energy consumption

Power consumption in WSNs can be kept under control relying on many different complementary approaches. Silicon manufacturers push CMOS technology miniaturization to reduce active power consumption while devising techniques for limiting leakage; similarly, efforts are spent in maximizing energy conversion: commercial devices have to be assumed as optimized in this sense. Microcontrollers, which are among the most power-hungry devices in the system, often provide a set of programmable features to reduce consumption, such as voltage and frequency scaling, and many low-power idle modes where some parts (e.g., the core, the peripherals, the PLL and the internal regulators) are selectively turned off by means of clock and/or power gating. Deeper "sleep" or "stand-by" modes typically require longer times for switching to and from the running mode, and may or not assure data retention in registers and memory blocks. In general, the microcontroller speed should be kept at the minimum while satisfying the computational requirements; fortunately, the processing needs of WSN applications are more predictable and stable with respect to general-purpose systems.

Another large part of energy is used by the radio transceiver. The power consumption for transmitting and receiving are almost comparable, but, while transmissions usually last only a few milliseconds, the radio should be able to receive packets continuously to handle random data requests and network

management messages. Several techniques, aimed at keeping radios turned off as much as possible, exploit a duty cycling mechanism. The radio duty cycling (RDC) tries to activate the receiving mode only when there is an incoming packet on the channel. Designers aim at very low duty cycles (< 1%, meaning that the radio will be active less than 1% of the time), using synchronous or asynchronous techniques, but to achieve this objective they will have to compromise on other network performance goals [6], such as end-to-end message delay, collision rates and control packet overhead.

Considering the power consumption of a sensor, one of the largest contribution is usually in the analog-to-digital converter, especially if it needs to work at a high rate or with high bit resolution. Then, the presence of arrays of sensing elements or active transducers, such as the ones in sonar, radar or laser rangers, can complicate the device architecture and increase consumption as well. Sensors used for detecting or identifying biological or chemical agents are particularly energy-demanding, especially when the sensing device needs to be heated or moved. Many techniques have been proposed for reducing sensor energy consumption in an IoT system and in wireless sensor networks. The general idea behind them is that a sensor should acquire a measurement sample only if needed, when needed, where needed and with the right level of fidelity, avoiding the generation of excessively large or precise data [7]. In order to do so, it is possible to exploit the sensor scalable performance, or periodic shutdowns, to adapt the behavior of the device to the specific environmental situation, e.g., changing bit resolution and sampling rate.

Complementarily to these device-level approaches, system-level methodologies can afford relevant savings in power consumption. In this case, the application requirements need to be considered at a higher level and translated into suitable network management policies. A sensor node, for instance, may have the possibility of locally processing some data and send only the results of computation, or transmitting a larger quantity of raw information to a non-power-constrained remote server: the most power-efficient solution depends on the system architecture and on the data to be processed and transmitted. Another example is the case of slowly changing environmental parameters, which may be sampled at distant instants, while turning off the complete network (and not just the transceiver) for long periods, applying a system-level duty cycling. Adaptive spatiotemporal sampling is yet another technique based on turning on and off specific devices distributed in an area to measure the effects of an event, considering redundancy and hierarchy of the sensing devices.

C. *Functional and power simulation*

For validation and optimization of WSNs, communication performance and robustness need to be evaluated. Prototypes can get expensive and require a long time, since many devices need to be used and kept working for days or months. The problem is exacerbated when energy need to be evaluated in the long time in the presence of harvesters with variable power output. For these reasons, a simulation approach is often preferred over hardware prototyping, also to enable a precise analysis of the internal states of the devices and to investigate at a greater system scale [8].

Many simulators for WSNs are available, and some of them also include features for evaluating power consumption and production: a few examples include WSNsim, OMNeT++, ns-3 Energy Framework, PASES, GreenCastalia and SensEH [9]. Their main differences are in genericity, in the ease of use, in openness and modularity, in availability of evaluation tools and in the possibility of running the actual code that will be uploaded in the real nodes. The latter is a specific characteristic of WSN simulators such as TOSSIM and COOJA, which are designed to use (through emulation) the deployment-ready code.

III. A Wireless Sensor Network Development Framework

The typical target system addressed by the proposed framework consists of a self-forming and self-healing multi-hop mesh network for ambient conditions monitoring, requiring a moderate data rate and tolerating a certain amount of latency. Sample applications include indoor climate monitoring, evaluation of strain or inclination in structures, vibration and impact checking, etc. The nodes are powered by batteries or energy harvesters. The data acquired by each node are transmitted and collected by one (or few) sink(s). The sink also provides the internet access to the other nodes of the network.

The framework enables the practical development of the WSN with the customization of the nodes and of the network structure, as well as the definition of network management policies. In addition, it supports system validation and parameters optimization by means of simulation, evaluating data exchange metrics, taking into account energy consumption and, when available, local energy scavenging.

A. *System architecture*

Each node consists of:

- A low-power microcontroller (ARM Cortex) with several integrated peripherals, such as timers, standard interfaces (I^2C, SPI, USART) and a real-time clock;
- An ultra-low power, Sub-GHz radio transceiver, such as SPIRIT1 by STMicroelectronics. Sub-GHz radios provide a valid trade-off in terms of data-rate, reach and power consumption;
- Digitally interfaced sensors, e.g., for temperature and humidity;
- A power module equipped with batteries or a harvester with a suitable energy buffer.

The Contiki operating system provides a basic set of functionalities to support the development of a mesh network, on which an application can be designed and implemented. The message exchange between nodes is based on the User Datagram Protocol (UDP). In order to ensure quality, UDP functionality is extended by means of adding a small acknowledgment message of received packets and retransmission of lost packets. Adopting an acknowledge mechanism for data exchange improves the reliability of the connection reducing the risk of packet loss and allows a lower redundancy on sent packets by retransmitting packets only when strictly necessary. Moreover, the acknowledge message enables the use of the Expected Transmission Count (ETX) routing metric for routing optimization, which expresses the robustness

of a link as a function of the number of retransmissions required for each packet. Contiki natively implements the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL); here the nodes are part of a Destination-Oriented Directed Acyclic Graph (DODAG) rooted in the sink. This organization supports communication between any possible couple of nodes: each node dynamically selects a preferred parent, creating a link towards the root, by evaluating robustness and distance metrics.

As anticipated, a viable approach for reducing the power consumption at the application level relies on a synchronous system-level duty cycling. The strategy we use consists in setting all nodes in a stand-by mode and periodically wake them up synchronously for short time windows (few seconds), with the exception of the gateway that is assumed to be powered by mains and thus always active, as depicted in Fig. 2. When in stand-by mode (i.e., with the microcontroller and the transceiver in an off-state, with data retention), the node power consumption must be in the order of few μW, and a real-time clock with a crystal oscillator needs to be running and able to wake-up the system by means of a programmable alarm. The active phases need to be long enough to enable the transfer of sensor data as well as the exchange of operating system messages aimed at maintaining network connections with discovery and repair operations. The more precise timing synchronization is, the shorter the active periods can be, since lower tolerances are needed to guarantee the overlap of active windows in the different nodes.

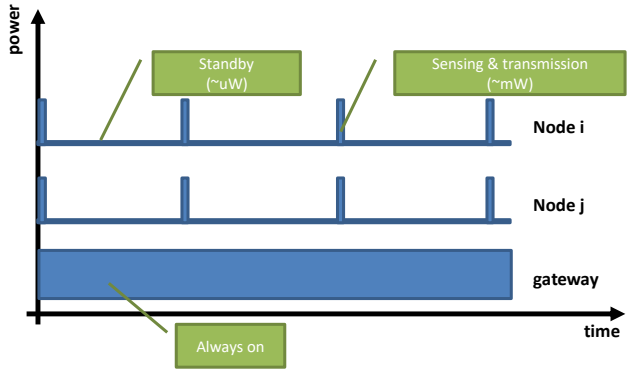


Fig. 2. System-level synchronous duty cycling operations

To enable the synchronization of the nodes, also accounting for drifts caused by crystal variability, the gateway periodically broadcasts a beacon message used by the nodes for regulating their timers. In order to reduce the latency impact due to beacon retransmission on nodes far from the sink, assuming that the radio propagation time is negligible, each node is synchronized with its preferred parent, only, and later asynchronously broadcasts the beacon to its children.

Naturally, the power consumption has to be kept to a minimum also during active system windows: this is achieved with the use of low-power features of the used components as described in the previous Section.

Fig. 3 reports the flowcharts of the tasks executed by each node: *a)* is the general flow, where the system initially waits for network connection and synchronization, and then enters the periodic data read and transmission/stand-by loop; *b)* is the timing synchronization procedure, executed every time a synchronization message is received as an interrupt routine.

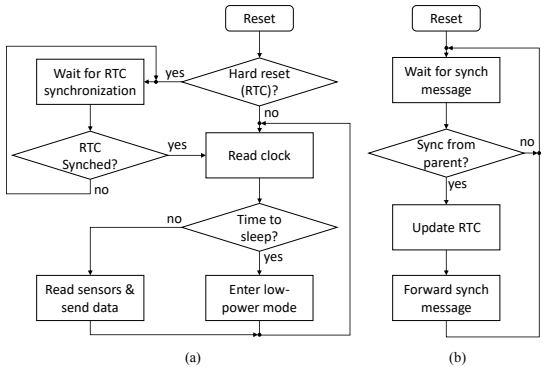


Fig. 3. Flowcharts of application-level node tasks: *a)* Client process; *b)* Periodic synchronization (RTC is Real Time Clock)

B. *Network simulator*

System validation is achieved using the Cooja Simulator, a network simulator specifically designed for WSNs and the Contiki operating system, and then some MATLAB post-processing. Cooja allows developers to test their code and systems long before running it on the target hardware. It allows analyzing RPL and network behavior both at the hardware level, which is slower but allows precise inspection of the system behavior, and at a less detailed level, which is faster and allows simulation of larger networks.

With Cooja it is possible to define the location of sensor nodes and then to monitor their input/output behavior and the hardware status, which is directly correlated with power consumption. In addition, Cooja can visualize the network structure and evaluate the communication performance taking into account radio behavior and interference.

For the system-level simulation of our system, to emulate the system-level duty cycling mechanism, the node clock tick requires to be suspended in each node during stand-by periods, with a modification of the Cooja main file (the really same action is done in the real implementation by disabling the interrupt on the systick). However, since the emulator works in an event-driven fashion, this configuration will force the simulator to check status variation even during low power phase when the operating system is inhibited: the result is a low-speed simulation during phases that are practically ineffective. A solution is easy to implement and consists of setting the next expiration timer event to the instant when the system is intended to wake up.

Data about the energy consumption and the energy harvested are combined using a MATLAB model of battery charge and discharge at the end of the simulation to evaluate operating time.

IV. Case Study

A prototype system was developed using nodes assembled with off-the-shelf devices belonging to the STM32 NUCLEO ecosystem by STMicroelectronics: a NUCLEO-L152RE board equipped with a 32-bit microcontroller running at up to 32MHz; an X-NUCLEO-IKS01A1 motion MEMS and environmental sensor expansion board (only the HTS221 temperature and humidity sensor is used by the application); an X-NUCLEO-

IDS01A4 radio transceiver board equipped with the SPIRIT1 transceiver (868MHz); and a STEVAL-ISV021V1 energy harvester board, equipped with an SPV1050 MPPT converter and battery charger, a 26.5cm^2 solar panel and a 120mAh Lithium coin cell battery. Fig. 4 presents the assembled prototype, obtained by plugging the expansion *shields* on the microcontroller board.

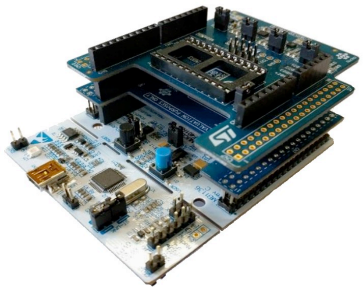


Fig. 4. A node prototype based on STM32 NUCLEO by STMicroelectronics

The system was validated and optimized in simulation, and then in a real-world experiment for monitoring indoor ambient conditions in a public building throughout one year. This case-study building of about 1,800 m^2 is located in Torino, Italy. It was originally built in 1902 and almost totally rebuilt at the end of the World War II. The building structure is a three-story load-based masonry with a heavy massive envelope, thus introducing significant attenuation and reflection of radio signals.

A. *System power optimization and characterization*

The prototype was first used for characterizing the harvester energy production in indoor conditions, in varying locations and in different periods of the year. The obtained power generation profiles (Fig. 10) are used in simulation. Power optimization was then performed on the microcontroller during the active system phases. With Contiki running and a sample data transmission application, idle operating system loops were counted at different operating frequencies, while measuring current drain, with the aim of minimizing their number: the results are in Fig. 5. The *sleep* mode, the fastest energy saving mode among the available ones, was also introduced in the main operative system loop to further lower consumption. Even if the system works also at 12MHz, a frequency of 16MHz was selected to provide some margin in case of additional unpredicted workload, at negligible power cost, using the sleep mode in idle cycles.

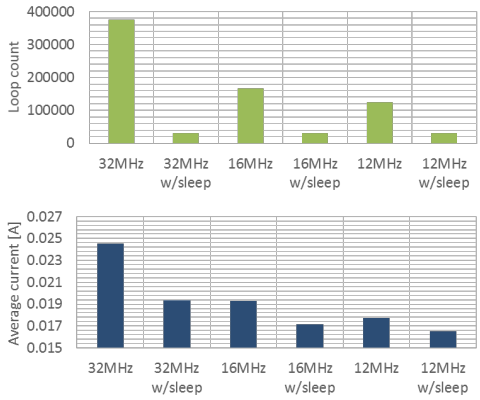


Fig. 5. Idle loops and current drain at different running frequencies

Table I reports measured power consumption values in active and stand-by states. In the latter state, the microcontroller is in *stop* mode, with data retention and real-time clock running, the radio is in *stand-by* and the sensor is in *power-down* mode. These data are also used in simulation (Fig. 6).

TABLE I. SYSTEM POWER CONSUMPTION

Mode	***MCU***	***Radio***	***Sensor***	***TOTAL***
Active	56 mW	Tx: 69mW Rx: 32mW	7µW	88÷125mW
Stand-by	9µW	3µW	2µW	20µW

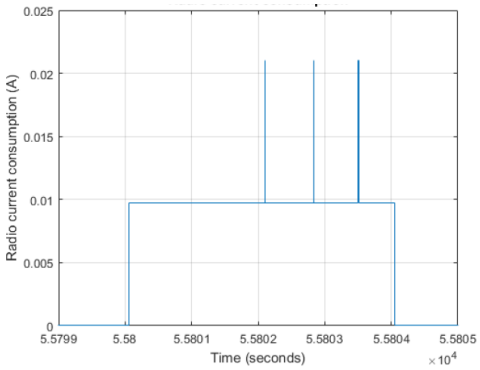


Fig. 6. Simulated current drain of the radio transceiver during an active window; the peaks correspond to data transmissions

B. *Simulation and results*

The behavior of the WSN was simulated using Cooja and MATLAB for data analysis, with different profiles, by changing number and relative position of nodes (up to 100 elements), message length, battery size, system duty cycling parameters and environmental light values. The simulations aimed at determining packet reception rate, power generation and power consumption. The received/sent packets ratio obtained in simulation was higher than 99.5%. However, these results only show how the network behaves without any external interference. Simulation, on a consumer-class laptop computer, runs at least 1,000 times faster than real time.

The average output power of the scavenger placed on a windows sill during a sunny day is in the order of 180µW. With a system acquiring data every 20 minutes, the simulation showed that it is possible to run the application relying only on the harvester energy. Fig. 7 shows the State of Charge of the battery across a 3-day simulation in such conditions. The graph also shows that with the small 120mAh Li-ion battery mounted on the harvester board, the system could tolerate a few days of total darkness.

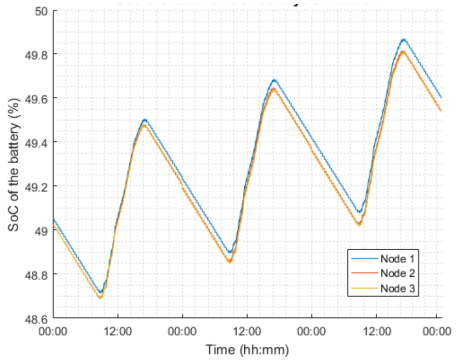


Fig. 7. Simulated State of Charge (SoC) of the battery across three days

The periodic node clock synchronization was experimentally validated, showing a precision of about 5ms (Fig. 8). Fig. 9 shows some of the results gathered from the real-world application, with 12 nodes installed in different rooms and running for about one year, confirming the validity and the reliability of the approach.

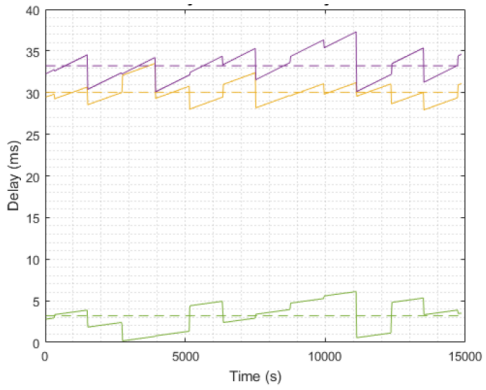


Fig. 8. Delay behavior of two nodes over some hours. Purple and yellow lines represent, respectively, the delay between node 1 and 2 with respect to a time reference at a distance of one hop (a 30ms fixed deterministic latency is due to message handling). The green line shows the relative delay between the two nodes. Slopes are due to oscillator variations w.r.t. the nominal value

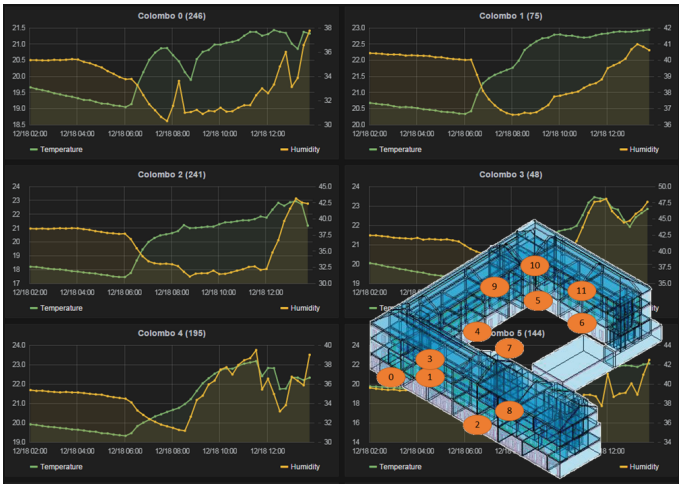


Fig. 9. Ambient data gathered from the real-world application

V. Conclusions

A framework for the development of custom WSNs for ambient data monitoring was presented, composed of a set of off-the-shelf low-power components and a simulation environment. Thanks to a system-level duty cycling methodology supported by a simple synchronization mechanism and to component power optimization, an energy-autonomous system was demonstrated to be feasible. A year-long real-world experiment further validated the system.

Works are ongoing to integrate power generation and consumption models within the Cooja simulator in order to avoid the MATLAB post-processing. This will also allow the nodes to evaluate in simulation the behavior of nodes using energy-based metrics for network management and dynamic system scheduling policies.

Acknowledgment

The authors wish to thank Francesco Giavatto for his contribution in the experimental development.

References

[1] Ericsson Mobility Report, Nov. 2017

[2] M. Belleville *et al.*, "Energy autonomous systems: Towards a ubiquitous sensor technology," Elsevier Microelectronics Journal, vol. 41, n. 11, Nov. 2010, pp. 740-745.

[3] M.C. Vuran, I.F. Akyildiz, "Wireless Sensor Networks," Wiley, 2010.

[4] A. Sassone *et al.*, "A top-down constraint-driven methodology for smart system design," IEEE Circuits Sys. Mag., vol. 14, n. 1, 2014, pp. 37-57.

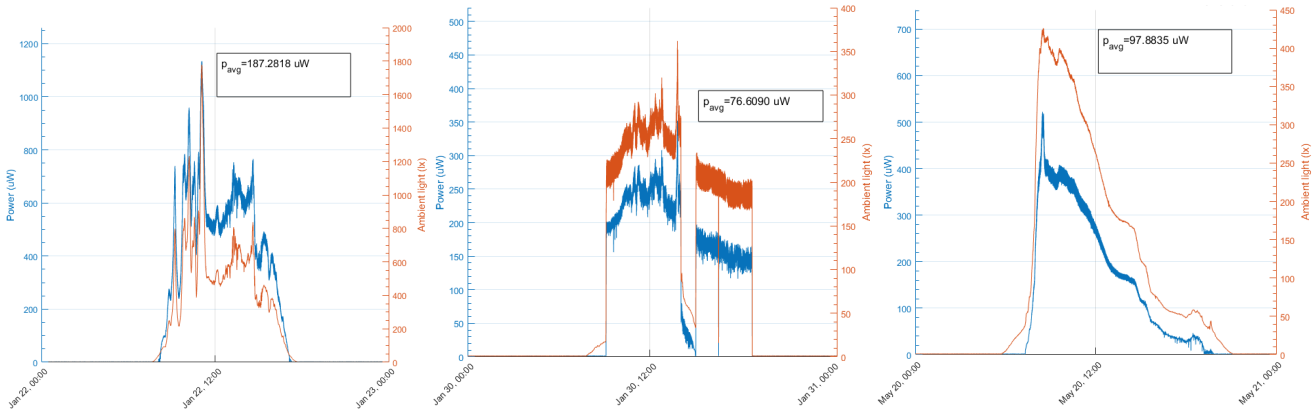
[5] T.V. Chien, H.N. Chan, T.N. Huu, "A comparative study on operating system for wireless sensor networks," IEEE Int. Conf. in Advanced Computer Science and Information Systems, 2011, pp 73-78.

[6] L.C.S. Magalhães *et al.*, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," IEEE Commun. Surveys Tuts., vol. 16, n. 1, 2013, pp. 181-194.

[7] V. Raghunathan, S. Ganeriwal, M. Srivastava, "Emerging techniques for long lived wireless sensors networks," IEEE Commun. Mag., vol. 44, n. 4, April 2006, pp. 108-114.

[8] R. Dall'Ora, U. Raza, D. Brunelli, G.P. Picco, "SensEH: from simulation to deployment of energy harvesting wireless sensor networks," IEEE Workshop on Practical issues in Building Sensor Network Applications, 2014, pp. 566-573.

[9] C. Tapparello, H. Ayatollahi, W. Heinzelman, "Energy harvesting framework for network simulator 3 (ns-3)," ACM International Workshop on Energy Neutral Sensing Systems, 2014, pp. 37-42.



(a) (b) (c)

Fig. 10. Luminance and harvested power in different ambient conditions: *a)* January in direct sunlight, *b)* January in the shade with artificial light, *c)* May in the shade without artificial light