UNIVERSITÉ LILLE 1 - SCIENCES ET TECHNOLOGIES

École Doctorale Sciences pour l'Ingénieur

IEMN - UMR CNRS 8520

Platform Hardware/Software for the Energy Optimization in a Node of Wireless Sensor Networks

Thèse soutenue le **27 juin 2016** par

Román José IGUAL PÉREZ

pour l'obtention du grade de

Docteur de l'Université Lille 1 - Sciences et Technologies

Spécialité

Micro et Nanotechnologies, Acoustique et Télécommunications

Composition du jury :

Rapporteurs :	**Olivier BERDER**	Professeur IUT Lannion - Univ. de Rennes
	Guillaume VILLEMAUD	MCF HDR INSA Lyon
Examinateurs :	**Nathalie ROLLAND**	Professeur Univ. Lille 1
	Jean-François DIOURIS	Prof. École Polytechnique Univ. de Nantes
	Congduc PHAM	Prof. Univ. de Pau et des pays de l'Adour
	Thomas VANTROYS	MCF Univ. Lille 1
Directeur de thèse :	**Laurent CLAVIER**	Professeur Télécom Lille
Co-encadrant :	**Alexandre BOÉ**	MCF Univ. Lille 1

À Jeannie,

À ma mère, mon père et ma sœur.

"La vida es sueño,
y los sueños, sueños son."

"Since is life a dream at best,
and even dreams themselves are dreams."

Pedro Calderón de la Barca

Remerciements

Avec ces mots, je voudrais exprimer ma sincère gratitude à toutes les personnes qui ont contribué à la réalisation de cette thèse.

Premièrement, je remercie mon directeur de thèse Laurent Clavier et mon encadrant Alexandre Boé, qui ont guidé mes pas d'une façon didactique et efficace pendant toute la période de la thèse. Un merci également à Thomas Vantroys. En outre, je remercie Nourddine Azzaoui, qui a contribué aux résultats de mes travaux. J'adresse également de grands remerciements à Nathalie Rolland et Christophe Loyez pour leur aide et pour la bonne gestion du laboratoire IRCICA et de mon groupe de recherche CSAM. À Rédha Kassi et Bernard Verbeke, qui m'ont accompagné dans mes travaux techniques, un grand merci, ainsi que à Lamine Koné, qui nous a permis d'utiliser la chambre anechoïque pour réaliser des expérimentations. De la même manière, je remercie le travail de Peggy Stankowski, Nora Benbahlouli, Ahmed Ben Abdeselam et Michel Soulage, qui assistent d'une façon efficace les chercheurs dans leurs problèmes logistiques et administratifs.

Je voudrais également remercier tous les membres du groupe CSAM pour leur soutien et la bonne ambiance dont j'ai pu profiter pendant toute ma période à IRCICA, notamment Viktor Toldov avec qui j'ai partagé d'innombrables heures de travail de recherche. Merci aussi à Rahul Vyas pour sa contribution directe à l'obtention d'importants résultats. Un grand remerciement aux personnes géniales que j'ai trouvé à IRCICA, mais dont la liste ne peut être exhaustive: Mauro Lopez de Freitas, Kamel Guerchouche, Aymeric Pastre, Rémy Bernard, Odile Robbe, Nicolas Araujo, Mohamed El Amine Seddik, Sébastien Jacob, Ousmane Tiemogo Hamidou, Xin Yan, Amal Abbadi, Omar El-Jouaidi, Laura Guerin, Émilie Soret, Philippe Mariage, Christophe Seguinot, Benoit Bensahla-Tani, Charles Anssens, Yoan Bourlier, Thomas Aviles, Wei Gu, Kouakou Kouassi, Paul Sangaré...

Pour finir, mon remerciement le plus chaleureux est dirigé vers ma famille. À Jeannie, merci beaucoup, la personne qui m'aide et qui me soutient à chaque instant de ma vie. À ma mère, mon père et ma sœur qui sont toujours un modèle pour moi. Au reste de ma famille, mes tantes, mes cousines et mes oncles et tantes français, merci pour leur soutien. De même, un grand merci à ma famille bretonne qui m'ont toujours ouvert leurs bras et accordé leur confiance. Je voudrais transmettre mon dernier remerciement à la famille qu'on choisit, mes amis. Merci à toutes les personnes que j'ai rencontré depuis mon arrivée à Lille, qu'ils soient français, espagnols, italiens, chiliens, argentins ou de n'importe quel autre pays.

MERCI À TOUS.

Abstract

Platform Hardware/Software for the Energy Optimization in a Node of Wireless Sensor Networks

The significant increase of connected objects in Internet of Things will entail different problems. Among them, the energy efficiency is essential. The nodes of wireless sensor networks should be operational for several years without replacing the batteries. This aspect leads to a significant challenge for the research community in last years. The present work deals with the energy efficiency, and more precisely with the study of the modeling of the energy consumption in the node.

We have designed a platform to instrument a node of wireless sensor network in its real environment. The hardware and software platform is made of:

- a hardware energy measurement platform;
- a software allowing the automatic generation of an energy consumption model;
- a node lifetime estimation algorithm.

The energy measurement platform recovers the current values directly from the node under evaluation. The current are measured component per component in the electronic circuit and function per function of the embedded software. This hardware/software analysis of the energy consumption offers important information about the behavior of each electronic component in the node.

An algorithm carries out a statistical analysis of the energy measurements. This algorithm creates automatically an energy consumption model based on a Markov chain. Thus, this platform allows to create a stochastic model of the energy behavior of a real node, in a real network and in real channel conditions. The model is made in contrast to the deterministic energy models found in the literature, whose energy behavior is extracted from the datasheets of the components.

Finally, we estimate the node lifetime based on battery models. We also show on examples the simplicity to change some parameters of the model in order to improve the energy efficiency, as well as the limitations of this solution.

This study represents one step of a global project that aims to optimize the energy consumption of the nodes in order to finally try to create an everlasting wireless sensor networks.

Résumé

Plateforme Matériel/Logiciel pour l'Optimisation de l'Énergie sur un Noeud de Réseaux de Capteurs sans Fil

L'incroyable augmentation d'objets connectés dans le monde de l'Internet des Objets impliquera plusieurs problèmes. L'efficacité énergétique est un des principaux. Les noeuds de réseaux de capteurs sans fil devront rester opérationnels pendant plusieurs années, sans remplacer la batterie. Le présent travail étudie l'efficacité énergétique et, plus précisément, la modélisation de l'énergie consommée par le noeud.

Nous avons créé une plateforme matérielle et logicielle appelée Synergie. Cette plateforme est composée d'un ensemble d'outils matériel/logiciel :

- un dispositif de mesure de la consommation d'énergie;
- un algorithme qui crée automatiquement un modèle de la consommation de l'énergie;
- un estimateur de la durée de vie du noeud.

La plateforme des mesures de l'énergie récupère les valeurs de courant directement du noeud. Ces courants sont mesurés composant par composant du circuit et fonction par fonction du logiciel embarqué. Cette analyse matérielle/logicielle offre information sur le comportement de chaque composant.

Un algorithme fait une analyse statistique à partir des mesures réelles d'énergie. Cet algorithme crée automatiquement un modèle de la consommation énergétique basé sur une chaîne de Markov. Ce modèle est une représentation stochastique du comportement énergétique du noeud en fonctionnement *in situ.* Le noeud fonctionne dans un réseau réel et dans des conditions réelles de canal. Le modèle a été créé à différence des modèles déterministes qui sont basés sur les données des datasheets des composants.

Finalement, une estimation de la durée de vie du noeud est réalisé en utilisant des modèles de batterie. L'estimation est possible grâce au caractère stochastique du modèle de la consommation. La possibilité de simplement changer les paramètres de consommation pour améliorer la durée de vie ainsi que les limitations de cet outil sont présentés.

Ce travail représente la première étape d'un projet global qui a pour but l'optimisation de l'énergie dans les noeuds pour finalement obtenir des réseaux de capteurs sans fil autonomes en énergie.

Contents

List of Figures

List of Tables

Introduction

The Internet of Things (IoT) is already a real technological and social revolution in the world. New studies count an increase of 30% reaching the 6.4 billion of connected objects [1] more in present year 2016 with regard to last year 2015 whereas, concerning the economic interests, this new business environment will produce a revenue of several trillion dollars [2] in next years. The research community is also very interested in IoT due to the large number of technical problems that this mass of connected devices will bring. One of the essential problems to study is the energy efficiency of the nodes included in the wireless sensor networks (WSN).

One of the main projects in the IRCICA laboratory is the energy optimization in WSNs. We work in the three axis of the energy optimization: energy harvesting, storage and consumption. Some members of our research team, CSAM (Circuits, Systems and Applications of the Microwaves) [3], work in the optimization of the energy harvesting devices through the improvement of the throughput in the photovoltaic cells [4]. Other members of CSAM team study also the increase of the capacity of the energy storage devices for WSN nodes, as the microbatteries [5] and the supercapacitors [6]. The energy consumption is studied in different dimensions as focused on hardware [7] or on the software [8] embedded in the nodes. This interest in the energy optimization has, as final objective, to design an everlasting WSN for the IoT world.

The work developed in this thesis is focused on the hardware part of energy consumption optimization. The different stages of this energy consumption project are represented in Figure 1. The process begins with a node of WSN composed by hardware components and programmed with a software. A platform is designed to analyze the energy behavior of the node under evaluation. This platform contains two parts: an analog part, where it is connected to the node in order to recover the energy consumption measurements; and a digital part, where several digital inputs/outputs (DIO) are connected between the measurements platform and the node in order to track the functions of the embedded software. Then, an energy model is created from the real energy measurements. This energy model represents

in a mathematical form the energy performance of the node. The third step consists in elaborating the design guides to modify the hardware and software in order to decrease the energy consumption. On the other hand, energy measurements and DIO indicators allow to realize an accurate analysis of the software in microcontroller of the node. Afterwards, an automatic treatment of the results is carried out to improve the software which manages the functions in the node. As seen in Figure 1, this is an iterative process. As perspectives, all these stages in the process should improve automatically the software in microcontroller. This adaptive process should choose the appropriate functions according to different parameters as the energy remaining in battery, the energy recovering in energy harvesting device or the interference level in channel.

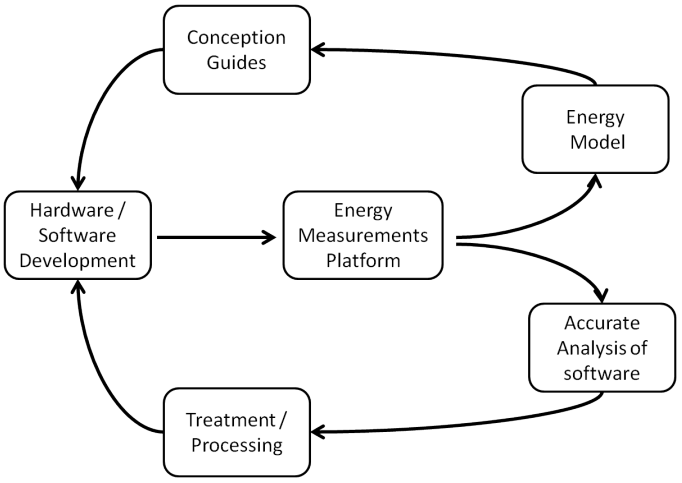


FIGURE 1: Energy consumption project in energy optimization for connected objects.

The global contribution of this work is the conception of a platform which measures, analyzes and allows to optimize the energy consumption in a node of WSN. We present the generic schema of the power meter used in this platform. The electronic components that compose this power meter can vary in order to improve the characteristics of data rate and accuracy. However, the schema of this power meter remains the same with real energy measurements recovered from each component in the node. In Chapter 1, we present the state-of-art of several power meters created by the research community and, then, another contribution of this work is the separation of the current values required by each hardware component in the node. This separation offers an important information about the behavior of each component according to the program included in the microcontroller of the node. Moreover, some indicators are included in the software of the microcontroller and made in the hardware in order to relate the instructions in the code with the energy measurements. This power meter and its experimental results will be exposed in Chapter 2.

In Chapter 3, we present another contribution of this thesis. This is an algorithm which works with the statistics from the energy measurements. This algorithm analyzes the current values recovered by the power meter and creates a Markov model composed of statistical data

of current and time as well as of the transitions of the states identified by this algorithm. This Markov model allows to estimate the node lifetime if the system works with the same hardware and the same software. Furthermore, the values in this model are easily modified in order to estimate the lifetime for a node with different electronic components or code programmed in microcontroller. Thanks to that, we can optimize the hardware and the software in the node to reach a more energy efficient network.

Finally, Chapter 4 concludes this study and presents the perspectives of the future work.

Chapter 1

Context - State of the art

1.1 Internet of Things

The era of Internet of Things (IoT) is already a reality. This phenomenon represents a new technological revolution in the world comparable to the Industrial Revolution of the 19th century. According to several organizations [9] [10], the number of connected devices has not stopped growing in the last years and this increase will continue to reach 50 billion of devices in 2020, *i.e.* seven connected devices per person in the world. Until now, computers, smartphones and tablets represent the most of these devices, but other new connected objects appear each year to facilitate our lives and to help to better manage industries. These connected objects are generally organized in networks, called Wireless Sensor Networks (WSN), in order to develop a more significant intelligence in a given space.

Indeed, IoT stays in constant evolution. According to [11], at the origins of IoT, objects were connected to the cloud in order to identify and localize the monuments or the sites of interest in the world. Later, objects are tracked during the delivery, from the sender to the client. Until this time, objects are equipped with tracking devices as GPS or RFID tags. Next stage of IoT is represented by WSNs, where objects measure the environmental conditions around them and communicate the measured parameters to the people who use the technology. In this stage, the interaction between human and electronic device, called human-to-machine communication (H2M), is important. This interaction becomes bidirectional in many cases because the objects can include sensors and, also, actuators. In next stage, sensor and actuator nodes are programmed to work according to external orders but, in this case, these orders are generated by another electronic devices. This is called machine-to-machine (M2M) communications. The system is managed by the nodes themselves and the human has a little influence. Last stage in this evolution of IoT corresponds to the moment when the whole

system will have a global intelligence and change the behavior of the connected objects depending on the environmental characteristics or on the actions of people. Moreover, the system learns by itself how to manage the different problems and the prediction of some of the future situations after this learning process is possible. These stages are explained as an evolution of IoT. At present, these stages are being studied and developed in parallel because they are composed of different concepts which are used together to improve the system.

Another point of view about the evolution of IoT is given by Jim Chase, from Texas Instruments [12]. As cited by other experts, Jim Chase forecasts that 50 billion connected devices will be installed in the world by 2020, but, furthermore, each person will interact with a trillion-node network in his whole life. Years after this date, the objects placed on our body will measure our activity and they could predict our needs in real-time. These needs will be automatically well supplied by the nodes placed around us, *e.g.* to switch the heating on, to vary the light intensity in the room or to prepare the bathroom for a shower.

In fact, the possible range of applications for the IoT is almost endless. The applications related with the tracking of objects are important, for instance the truck fleet, the merchandises in trucks or the containers in ships. In this type of applications, the localization of the object is the main information to be obtained. In other cases, different type of information is

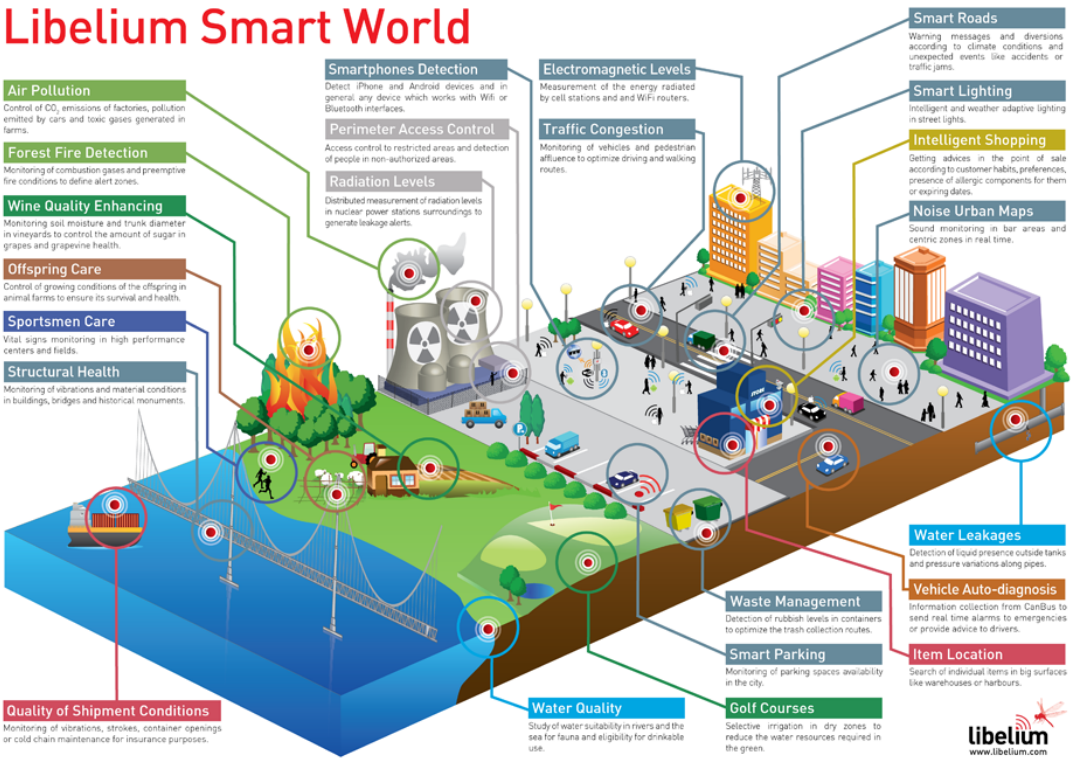


FIGURE 1.1: The world of IoT [13].

required as the temperature, the movement, the luminosity, the noise, the atmospheric pressure, the humidity, the air quality, the presence of another object or a person, among others. Figure 1.1 shows an example of this near future according to the vision of Libelium [13]. These applications are present in the Smart Cities with the monitoring of the water and electricity meters in the buildings, the monitoring of the streetlights as well as the traffic lights, but also in the Smart Buildings with the new devices for home automation, the detection of intrusions or the monitoring of the air quality indoor. Other applications about the detection of fire in forests or the structure health monitoring in buildings, tunnels or bridges are also under research [14]. The healthcare is also an essential field to develop the IoT: wearable objects to monitor heart rate, activity and sleeping periods of people are created. Many other applications related to IoT are being developed and others will emerge in the near future.

Dave Evans, from Cisco, explained already in 2011 [9] this exponential increase of connected objects and the main problems that this expansion could cause. Figure 1.2 shows the predictions of the increasing number of connected objects after some decades according to CompTIA [15] based on Cisco previsions, even if other sources differ from them (Gartner [1] forecasts "only" 6.4 billions of connected objects in 2016 and 20.8 billions in 2020 whereas another source as Business Insider [16] projects near 15 billions in 2016 and up to 34 billions in 2020). Among these main problems were the deployment of IPv6, the energy optimization and the creation of standards in the areas of security, system architecture and communications. Many advances in these fields have been carried out since then. The IPv6 communications protocol is a solution to identify each connected device in the world as a different device. This is possible in WSN thanks to 6LoWPAN protocol which consists in a compressed

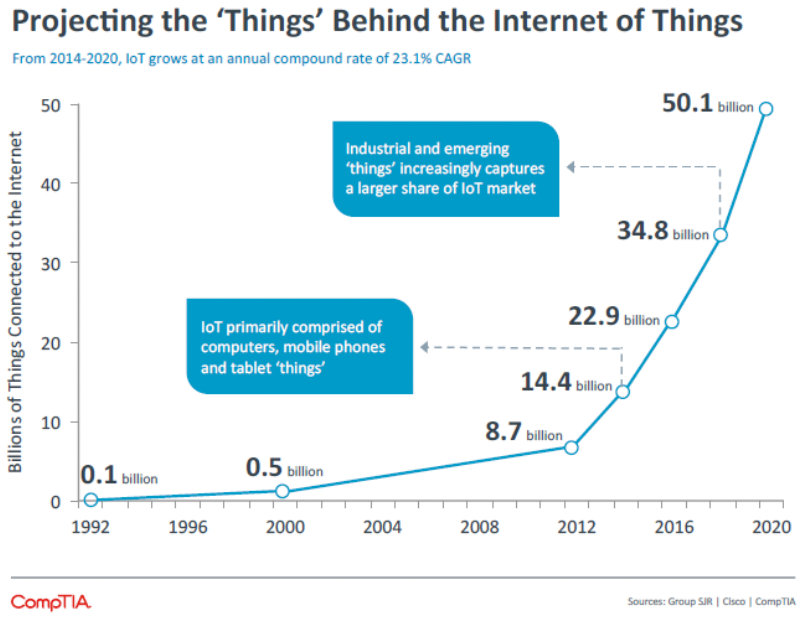


FIGURE 1.2: Estimation of the increase of connected objects until 2020 [15].

version of IPv6. The standards in different areas in order to use efficiently a WSN are also being studied and improved year after year. The standardization of the elements around the WSN represents a sign of the maturity of this technology.

The energy efficiency of the connected devices is one of the main technical issues in IoT. Most of the devices are supplied by individual batteries containing a limited electrical energy. The level of the charge in battery decreases depending on the activity of the device. A decrease of the charge entails also a decrease of the voltage. The relation between charge and voltage is not linear, the voltage can decrease to reach a value not sufficient for the electronic circuit to work. At this point, we consider that the node lifetime is expired and the battery must be recharged or replaced. Handling hundreds or thousands of nodes every month or every year becomes a really hard task. Another problem is the accessibility to the nodes if these are placed in harsh environments. Moreover, with the massive deployment of these WSN, the interference level as well as the channel occupancy increase dramatically, which implies loss of information packets and re-transmissions, then, higher energy consumption. Thus, the study of the energy optimization for these devices constitutes an essential work.

1.2 Energy optimization in WSN

Energy optimization or energy efficiency represents a very important topic to study in different fields of research in the world. We find some examples of studies related with energy optimization in electric cars [17], in buildings and houses [18] or in electronic devices such smartphones [19]. The research about energy efficiency in WSNs obtains a different dimension. In most cases, nodes included in WSNs are supplied by a battery. This battery cannot generally store a large quantity of energy because of its small size. Normally, the battery size is in line with node size in order to be unseen together with the objects where the nodes are installed. Moreover, in many applications, these nodes have to work for several years without replacing the batteries. These limitations make necessary the research about energy optimization in WSNs.

The energy efficiency and, thus, an extended lifetime in nodes of WSNs will be reached when the three axis of energy optimization are improved:

- Energy consumption.
- Energy storage.
- Energy harvesting.

A typical structure of a node of WSN is depicted in Figure 1.3. This node is composed of several electronic components as the microcontroller which is the brain of the node, responsible for ordering the other components to accomplish the tasks defined in the code by the user. The RF module is also essential in a node of WSNs because of the important characteristic of wireless in this type of technology. This RF device is in charge of communicating with the others nodes in the WSN by transmitting and receiving information packets. The sensing unit includes one or several sensors which take information from the environment and offer this information to the microcontroller. An external memory is an optional device that may improve the performance of the node according to the application executed by the network. Other components can be added as a different RF module or a wake-up radio module with the same purposes. All of these components consume energy (except the passive wake-up radio) with a different scale according to the type of component and the action that this component executes.

Another primordial element is, of course, the energy. This energy is stored and offered by an energy storage device, generally, a battery. However, it exists other possibilities to store the energy as in a capacitor and in a supercapacitor. Moreover, we can include in a same node two of these storage components since they contains different and complementary characteristics. The system composed of two devices, battery and supercapacitor, is called *Hybrid Energy Storage System* (HESS). The decisions to use the energy from or recharge each component are controlled by a power management unit. In order to achieve the everlasting sensor networks, the nodes should also integrate an energy harvester which converts the physical

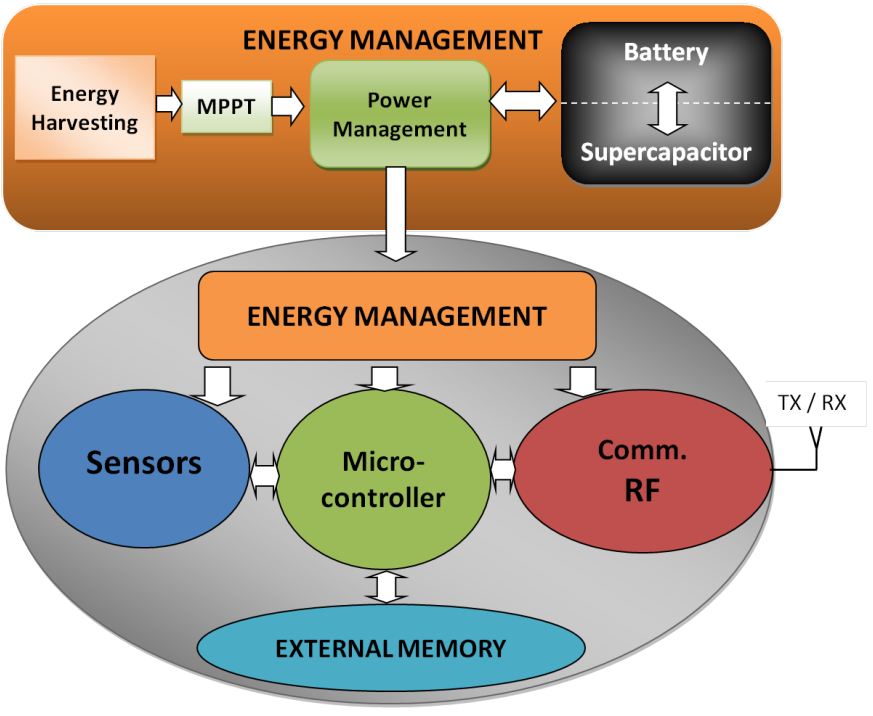


FIGURE 1.3: Node of WSN with energy management unit.

elements from the environment (*e.g.* light, movement, changes of temperature) in electrical energy. To optimize the process, the energy harvesting unit includes some other components. Among them, a *Maximum Power Point Tracking* (MPPT) device ensures the recovery of the maximum energy and attenuates the differences between the sudden energy peaks and lacks in order to avoid the possible damages in battery. After that, the energy recovered is accumulated in the storage unit. All of these energy inputs and outputs are controlled by the power management unit.

1.2.1 Energy Consumption

Minimizing the energy consumption in the node is essential in order to extend its lifetime. The most energy-hungry (with the highest current values) in the node is, generally, the transceiver because it must generate a signal strong enough to reach another node which is placed at a certain distance. The current required by the RF module is in the scale of tens of milliamperes (mA). Besides, the microcontroller consumes in the order of the unities of mA. There are a wide range of sensors, but, generally, they require currents in the scale of the microamperes (μA).The minimization in energy consumption must be carried out together in hardware and in software. If we choose the most energy-efficient hardware components to include in the device but the software programmed in microcontroller is not optimized, we will not reach this minimization of the energy consumption. We obtain a similar result when the choice of the communication technology configured in the transceiver of the node is not the best for the developed application. Many parameters must take into account when minimizing the energy consumption. This thesis deals mainly with the study of minimizing the energy consumption in the node.

1.2.2 Energy Storage

The energy storage is another element to optimize. It has to be increased to reach the optimization of the system. It exists different types of energy storage devices to supply a node of a WSN as batteries, supercapacitors, capacitors or fuel cells. The comparison of these elements according to their specific power and their specific energy is depicted in Figure 1.4. We can observe that the batteries store a large amount of energy compared to the capacitors, but batteries are limited in power peak. On the other hand, the capacitors offer a high output power but the energy stored is tiny, so that, they are rapidly discharged. This is why a new storage device has been created to fill the gap between batteries and capacitors. This device is called *supercapacitor* or *ultracapacitor* and it is able to offer a high output power and to store a large amount of energy.

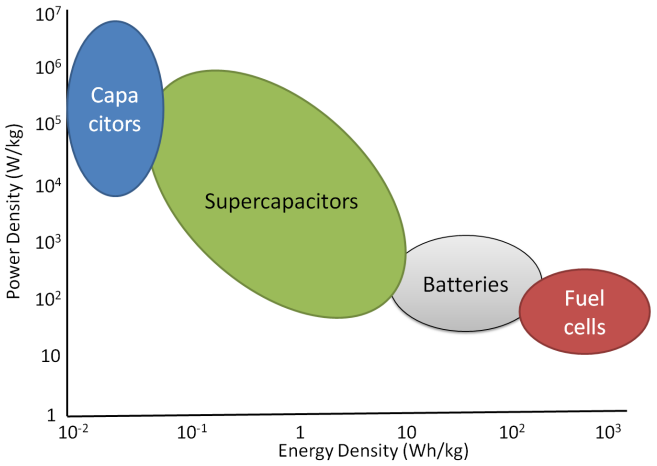


FIGURE 1.4: Comparison of the energy storage devices.

Another important difference between batteries and supercapacitors is the number of charge-discharge cycles. Batteries can sustain a limited number of charge-discharge cycles due to their chemical properties. For example, the lithium-ion batteries can reach 1200 charges whereas the nickel metal hydride batteries can have less than 1000 charges and the lead acid batteries only 800 charges approximately. On the other hand, some supercapacitors can reach up to one million charge-discharge cycles [20] without noticing a negative impact in the amount of stored energy. A comparison of the technical characteristics of a lead acid battery, a supercapacitor and a capacitor is shown in Table 1.1 [21]. Other storage devices used, but more rarely, in WSN are the fuel cells [22]. This component is able to convert fuel in electricity through a chemical reaction. The fuel cell can store a high amount of energy but, generally, it is not rechargeable or the recharging process must be manually executed. Then, we find again the problem of handling the nodes to charge or to substitute the energy source. An impossible task to complete when considering thousands of nodes or nodes placed in harsh environments.

TABLE 1.1: Comparison between lead acid battery, supercapacitor and capacitor [21].

Available Performance	Lead Acid Battery	Supercapacitor	Capacitor
Energy Density (Wh/Kg)	10 - 100	1 - 10	<0.1
Power Density (W/Kg)	<1 000	<10 000	<100 000
Cycle Life (cycles)	1 000	>500 000	>500 000
Charge/Discharge Efficiency	0.7 - 0.85	0.85 - 0.98	>0.95
Discharge Time	0.3 - 3 hours	0.3 - 30 secs	10^{-3} - 10^{-6} secs
Charge Time	1 - 5 hours	0.3 - 30 secs	10^{-3} - 10^{-6} secs

The most used energy storage device in WSNs is the battery. It contains a large amount of energy and its output power is not very high but most of the electronic components integrated in the nodes of WSNs become operational with a low power. In some cases, the use of a supercapacitor with a battery can overcome the constraints of battery as the low output

power or the limited number of charge-discharge cycles. This system of battery and supercapacitor is called *Hybrid Energy Storage System* (HESS). This set of energy storage devices is highly used in really different applications. For instance, electric vehicles [23] [24] where the supercapacitor is used to help the battery when starting the machine. Supplying the buildings in remote areas [25] and traction of vehicles as trains or trams as well as the industry of wind power plants [26] are other applications where a HESS can be present.

In a HESS, the supercapacitor is recharged via an energy harvesting as many times as necessary. On the other hand, the battery is just recharged when its energy level decreases until a critical threshold because the intermittent recharges can decrease the battery lifetime [27]. In [28] and [29], the authors present a HESS with a photovoltaic (PV) panel where different states are defined according to the charge level in the supercapacitor. The energy recovered by the PV cell is stored in the supercapacitor and, normally, the circuit is supplied by the supercapacitor. However, in some cases, mainly during night or the cloudy days, when solar energy is not sufficient, the battery help mode is activated and the power supply is delivered by the battery. In [30], a HESS is proposed together with several energy sources as a PV panel and a wind generator. In this case, a fuel cell is included in the system but it is considered as another energy source with whom the battery and the supercapacitors are recharged. In this work, the authors highlight the importance to search the Maximum Power Point (MPP) in the energy harvesters to recover as much as possible energy from the environment.

1.2.3 Energy Harvesting

Energy harvesting corresponds to the third axis in energy management to reach the energy optimization. Different methods exist to recover the energy from the environment. Figure 1.5 from [31], shows the sources and the methods used for energy harvesting. A method is chosen depending on the application. The photovoltaic cell is one of the most efficient devices because the solar light is a powerful and endless energy source. In WSNs, the PV cell is mainly used for outdoor applications but, in some cases, it is interesting to recover the energy from the artificial light in the buildings. In [32], Wang *et al.* carry out a comparison between the efficiency of the PV cells used for solar light and artificial light. They compare different sources of light as the sunlight, but also, the indoor lights as halogen, fluorescent or LED lights. The solar panels to recover the energy from these light sources are different due to the different wavelength where light has a higher power spectral density. Then, it is possible to recover energy from artificial light at night. The use of PV cells in WSN is an attractive solution to reach the everlasting sensor networks, even for indoor applications.

Another type of energy harvesting device used in WSN is the piezoelectric harvester, which converts the mechanical vibration or the pressure in energy. Each piezoelectric element has

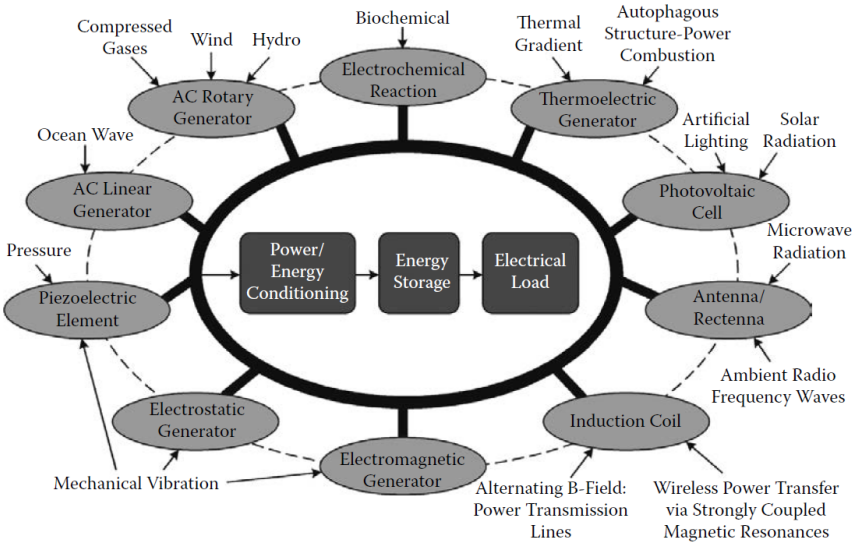


FIGURE 1.5: Energy harvesting methods and sources [31].

a resonant frequency where the recovered energy reaches the maximum values (about hundreds of microwatts [33]). This device is generally used in applications where the nodes are placed on infrastructures as bridges, tunnels or trail rails and, then, the piezoelectric element is excited by each passage of a vehicle or a train.

Thermal energy harvesting is another method in WSN to recover energy from an external source. In [34], the external source corresponds to the human body and the authors study the deployment of a WSN on the human body for healthcare purposes.

Besides, the wireless power transfer represents an increasingly studied energy harvesting method in WSNs. The effect of induction cells is highly studied because the power transfer efficiency can reach important levels, as in [35], more than 50%. On the other hand, Zhou *et al.* [36] present a theoretical receiver operation which takes the received signals for information decoding but, also, for energy harvesting.

1.2.4 Energy Optimization in Three Axis

The energy optimization is obtained by improving these three axis: energy consumption, storage and harvesting in the node of WSNs. In [37], IDEA (Integrated Distributed Energy Awareness) is presented. This system consists in measuring each energy component of each node and to share this information with the rest of nodes in order to perform a collaborative energy management in the network. Figure 1.6 shows the architecture of this system. The values of input charge (C_n), battery level (B_n) and load charge (L_n) are calculated in node n. These values are shared with all the nodes in the network and each node calculates the most energy-efficient route to send the data. Then, the node decides which node's immediate

neighbor to send the next packet in order to maximize the network lifetime. To avoid the part of measuring the energy consumption by each node, some states of operation are defined and all the nodes in the network know the energy load for each state. These states are called *client states* (s_k) whereas the set of states is called *global state* (S), defined as $S = \{s_1, s_2, \ldots, s_k\}$. The main goal of IDEA is to determine the optimal global state for the network according to the different variables: energy consumption per state ($L(S)$), battery level in all the nodes (B), energy harvesting charge (C), but also the *utility* ($u(S)$), *i.e.* the most interesting state selected by the node corresponding to a more attractive state in terms of throughput. The coefficient α represents the tradeoff factor between an energy-efficient utility for the node and a more interesting operation mode. If $\alpha = 1$, the system chooses a low-power state, while, if $\alpha = 0$, the state with a greater throughput is selected. For instance, the sleep mode is always the most energy-efficient mode, but if the node stays permanently in sleep mode, no action is made by the node and the situation is not interesting for the network. Contrariwise, if the node stays in active mode all the time, it takes steadily the measurements from the sensors and transmits these data to the other nodes, the throughput of these actions is excellent but the battery level decrease dramatically. This example shows the difference between maximum and minimum α. Then, the system must reach a compromise between both operational modes.

IDEA presents an interesting manner to extend the lifetime of WSNs. The three axis are included in the system: energy consumption, harvesting and storage. This dynamic and distributed system allows to estimate the optimal operate state for each node and the most energy-efficient route to send the packets across the network. However, actually, if the nodes send this additional information to its neighbors, the payload of the packets increases and, consequently, the active time of the transceivers in the node increases in the same manner, by limiting the battery lifetime. Furthermore, the node stays active during the calculation

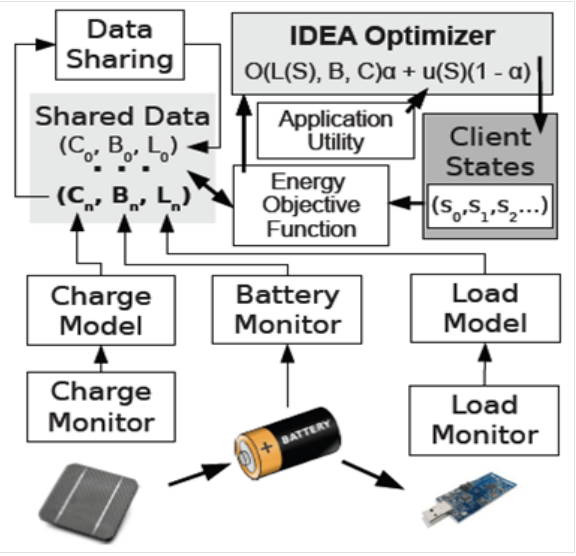


FIGURE 1.6: IDEA architecture [37].

of the optimal state for each node in the network. This operation may be hard for the small microcontrollers included in the nodes. Then, the time of calculation becomes important and, consequently, these operations impact in the lifetime of the nodes. Thus, the knowledge of the relation hardware/software in the node is important for its energy efficiency. This relation hardware/software corresponds to a topic studied in this work.

1.3 Energy efficiency in protocol stack layers

As previously explained, the energy consumption constitutes one of the three main axis to reach the energy optimization. Henceforth, in our research work, we focus on the energy consumption in the node.

The International Organization for Standardization (ISO) established in 1984 the Open System Interconnection (OSI) model [38] for computer networks. This model was created to ensure the deployment of communications systems around the world, since the coexistence of the emerging computer technology standards caused problems in the interconnection of the network. At the origin of the WSNs, the research community created a protocol stack based on the OSI model in order to solve the problems of wireless communications between nodes.

The protocol stack in WSN is a simplified form of the traditional ISO/OSI model. In WSNs, this protocol stack is composed of five layers: Physical (PHY) layer, Data Link layer with Logical Link Control (LLC) sublayer and Media Access Control (MAC) sublayer, Network layer and Application layer. Figure 1.7 depicts these layers placed in the protocol stack. This graph expresses that the PHY layer is linked to the single node whereas the other layers intercede in the communications between nodes and in the whole network.

The energy optimization may be done in each layer of this protocol stack. Each layer plays an important role to minimize the energy consumption. Many research publications have addressed this issue. Next, we present some works found in the bibliography about the energy optimization in the protocol stack layers.

1.3.1 Application Layer

Application Layer is the upper layer in the stack. The information received from other devices as well as the information recovered from the sensors in the node is used in this layer according to the final application. The manner to manage the energy of the node depends on the type of application. In some applications, the nodes are supplied directly by the power network from a socket or by a big battery as in home automation or smart vehicles [39]. In

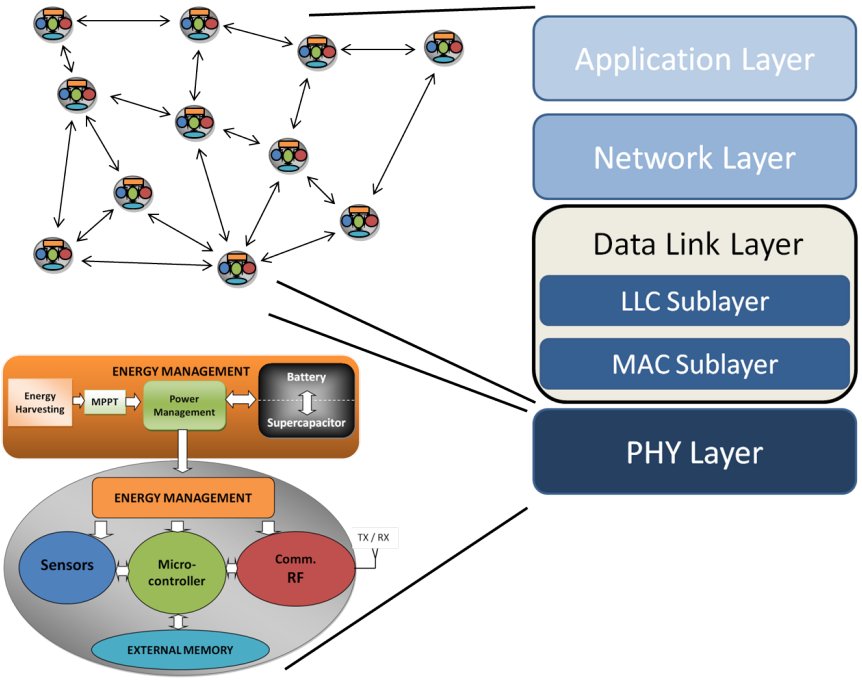


FIGURE 1.7: Node and network with the communication protocol stack in WSNs based on ISO/OSI protocol stack.

these cases, the energy is not considered as an issue. However, in many applications, it is an issue. An example is the application of habitat monitoring in [40], where the authors deployed a 32-node network in a small island in order to obtain in real-time the information of temperature, humidity, luminosity, among others. Another example is the wildlife tracking [41], where a node is installed on the body of an animal to recover the information about physiological characteristics. In other applications as water meter monitoring [42] or trash monitoring [43] in cities, nodes are easier to handle but the change of the batteries in thousands of nodes frequently becomes an unfeasible work.

We classify applications in three types: event-driven, demand-driven or periodic data collection. Some protocols have been created to facilitate the development of these types. Depending on the type, the energy management in the node changes. Figure 1.8, from [44], shows the difference between event-driven and demand-driven applications to monitor the same area with the same WSN.

In *event-driven* applications, the nodes recover data from the environment through the sensors installed in their circuits. Then, the data are processed and evaluated in the node. If the obtained values exceed some established thresholds, the alarm is activated and packets are sent in order to inform the sink about the new event. Generally, the transceiver in the node stays in sleep mode while no event is detected, but, once the alarm is activated, the node switches the transceiver on and a packet is transmitted to its neighbors. To reach the end devices, multi-hops are needed. However, if the node is programmed as a router, this router

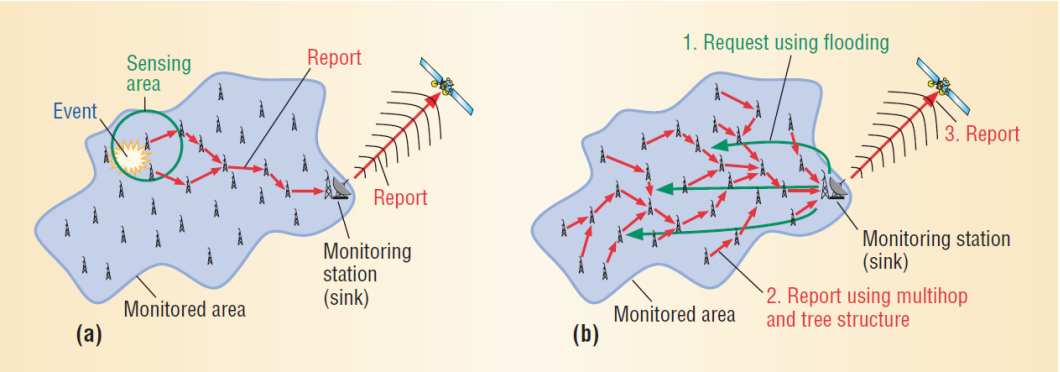


FIGURE 1.8: Two types of applications configured for the same network: a) event-driven and b) demand-driven applications [44].

has to switch its transceiver on periodically to listen to the channel and to receive a packet generated by a neighbor. These idle listening periods consume most of the energy in this type of applications because the alarms are not often activated but the router must verify the channel frequently. The event-driven method in WSN is used in some applications, as forest fire detection [45] or intrusion surveillance [46].

On the other hand, *demand-driven* (also called *query-driven*) applications in multi-hop architecture are based on the request from the sink to receive a value coming from one or several nodes. In this case, the routers as well as the end devices must listen to the channel periodically in order to receive the request from the sink or to forward the answer towards the sink. This double role for the routers increases the energy consumed as well as the latency in the response, which is higher than in event-driven applications.

The *periodic data collection*, also called *time-driven*, is another method to recover the information from the node to the sink. It consists in sending the data periodically from a node to its neighbor nodes. The synchronization between both nodes, the transmitter and the receiver, is necessary to achieve a correct transfer of the packets. Moreover, this synchronization is also important to avoid unnecessary or too long idle listening states and, thus, an increment of the energy consumption. In some cases, a packet transmission schedule is established for the nodes in order to reach this purpose [47]. Due to the different oscillators in nodes that control the frequency clock of the microcontrollers, the time measured in each node can be slightly different between nodes. After some time, the variations of the clocks may lead the transceiver to lose the synchronization. A strategy to solve this problem consists in calculating the gradient clock between the nodes and correcting dynamically these variations [48]. The calculation is realized by the same nodes as well as the correction of the speed clock in the program of the microcontrollers. This process of calculation consumes an extra amount of energy in the microcontroller but this energy generally is much lower than

in the case of incorrect synchronization between devices. Some applications based on periodic data collection have been developed in different fields, as air quality monitoring [49] or urban traffic monitoring [50], among others.

It is to be noticed that recent solutions have been proposed to suppress the multi-hop difficulty (which also suppress the routing complexity) proposing low-power long-range communications (*e.g.* LoRa, SIGFOX, Weightless), showing the difficulty to implement those solutions. One huge challenge in the coming years is to have an accurate analysis of the nodes' consumption whatever the type of application and to adapt the node behavior to the specific use case. One goal of our work is to develop an analysis tool for the nodes consumption that will be able to adapt to the different contexts and identify the strengths and weaknesses of the deployment.

1.3.2 Network Layer

The Network Layer in protocol stack specifies the route of the packets across the nodes in a network with a multi-hop architecture. Generally, in WSNs, the transceivers included in the nodes have a transmission range of only several tens of meters. This is due to the low output transmission power in order to decrease the input power supplied by the battery and, thus, to decrease the energy consumption. Moreover, both, end devices and routers, recover the information from the environment in the different positions where they are placed. Low power transmission and large area monitoring, these two reasons explain the need of organizing the wireless sensors in networks.

In a tree topology network, the routers guide the data packets from the end devices or from other routers until the sink or coordinator node. In a mesh topology, any node can be used as a router as well as an end device. The main challenges to solve with the study of routing protocols are presented in [51]. One of the major issue to solve is energy consumption [52]. Some protocols focus on the energy-efficiency: Low-Energy Adaptive Clustering Hierarchy (LEACH) [53], Power-Efficient Gathering in Sensor Information System (PEGASIS) [54], Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [55], Extending Lifetime of Cluster Head (ELCH) [56] or Sleep/Wake Scheduling Protocol [57]. This shows that the energy optimization is considered by the research community as a primordial constraint in network layer. If we will not study these routing protocols, we want to develop a measurement solution able to take it into account in order, for instance, to evaluate its weight on the global consumption.

1.3.3 LLC Sublayer

The Data Link Layer (DLL) is divided in two sublayers: Logical Link Control (LLC) sublayer and Medium Access Control (MAC) sublayer. The task of DLL consists in the creation and maintenance of direct communication associations, called links, between nodes and, moreover, it is responsible for sending the information across these links.

The LLC Sublayer is one of the sublayers of Data Link Layer. LLC layer controls flow control, frame synchronization and error checking in packets. In WSNs, MAC layer plays a more important role than LLC layer. This is why, the energy optimization in LLC layer is less studied.

1.3.4 MAC Sublayer

Medium Access Control (MAC) Sublayer is also included in DLL. This sublayer is responsible for providing proper access to the nodes that communicate in the same channel. The most important problems that MAC sublayer must solve are closely related to the energy efficiency:

- *Collisions*: Channel conditions or interference from other signal sources may corrupt the transmitted packets. This interference, especially when it reaches high levels, can also prevent totally the receiver from hearing the channel. In these situations, the transmitter retries to send the same packet. Then, the new trials cause a considerable increase of the energy consumption.
- *Overhead*: The discussions between nodes may contain several control packets as a preamble sequence, Request-To-Send (RTS) frame, Clear-To-Send (CTS) frame, acknowledgement (ACK) frame. Each control packets means longer periods in active state and longer periods to wait for the response from the other node, thus, more energy consumed. Optimal situation can correspond to a trade-off between robustness and energy-efficiency.
- *Overhearing*: In a network, the nodes accept the packets that they are able to hear. These packets are treated and evaluated. If the destination node address indicated in the packet header does not correspond to the node which has analyzed the packet, this node has employed some time to analyze the packet and, thus, it has unnecessarily consumed extra energy.
- *Idle listening*: Most of the time, nodes stay in sleep state. They wake up periodically according to the established duty cycle in order to listen the channel for receiving packets. The idle listening periods should be as short and infrequent as possible but without decreasing the throughput of the system.

The MAC protocols proposed by the research community try to limit these type of phenomenons. In [58], the author presents a classification of the existing MAC protocols to date, which are dedicated to improve the aspects as energy-efficiency, reliability and communication delay. The most of them carry out improvements related to the energy efficiency, as XMAC [59], BMAC [60], TMAC [61] or SCP [62]. Impact of re-transmission on idle listening, for instance, is difficult to evaluate experimentally or theoretically. Our work proposes a solution to address this task. Moreover, a part of our study is focused on the impact of interference in energy consumption, where we take into consideration the idle listening periods and the re-transmissions of packets.

1.3.5 Physical Layer

The Physical (PHY) Layer is related to the electrical and physical specifications of the devices. The main function of the PHY layer is to convert the bit streams coming from the upper layers in electromagnetic signals that carry the information across the wireless channel. PHY layer carries out the carrier frequency generation, the signal detection, the modulation and the data encryption. The antenna sensitivity and the transceiver characteristics are also important to determine the quality of the link connection between nodes.

Generally, the most energy-hungry device in a node is the RF module. This is due mainly to the throughput of the connection link, a relatively high output power in order to transmit further and a small sensitivity in reception to reach longer distances. The WSNs use the free Industrial, Scientific and Medical (ISM) bands, where a number of RF devices, even using different technologies [63], share the same frequencies. This fact, together with the widespread growth of the quantity of nodes in the world, brings serious problems because of interference and collisions.

The different technologies used in WSNs are mainly classified in three techniques [64]:

- *Narrow-band*: The nodes of the earlier WSNs use this type of technology. It is characterised by a narrow frequency band and, then, a low data rate together with a low energy consumption. SIGFOX [65] has further pushed this idea to an extreme, using ultra narrow band (UNB) signals [66].
- *Spread-spectrum*: This technique has been developed to improve the robustness against the noise and the interference. In this case, the bandwidth of the signal is larger, thus, the robustness increases. LoRa technology [67] utilizes a spread spectrum based modulation to avoid the corrupted packets because of the effect of other wireless technologies.

– *Ultra Wide Band (UWB)*: This technology is mainly used in indoor applications due to its robustness against multi-path effects. UWB is characterized by short-range as well as by low-power communications. One example of communications standard characterized by UWB technology is the IEEE 802.15.4 [68].

Other aspects (channel coding, equalization, ...) can compose the PHY layer. If it remains rather easy to be studied in point-to-point communications, the effect to networking the nodes and the link with the MAC layer are much more difficult to evaluate. Our work also aims at proposing solutions to this complex problem.

1.4 WSN Simulators, Testbeds and Power Meters

Numerous systems are dedicated to prove the energy efficiency of the algorithms developed in the different layers of the stack model explained in Section 1.3. Tests are carried out in three different ways: simulations, experimental testbeds and real power meters.

Generally, the simulators in WSNs offer a theoretical study of the behavior of the WSNs according to the topology, the routing protocols, the MAC protocols, hardware components, program in microcontroller, among other parameters configured for the simulation. These parameters are relatively easy to be modified if the simulator allows it. An advantage of using simulators is that they bear a wide deployment of nodes in the network. This task is not easy when we work with real hardware. Contrariwise, the results are obtained from theoretical approaches, *i.e.* the simulators take into account some phenomenons as the noise in the channel, the interference , the fading or the retransmissions of packets in a theoretical perspective. However, it is also important to work with real devices as in experimental testbeds.

The testbeds allow to study the behavior of a WSN in a controlled environment. All the elements are known: the type of nodes, their positions in the network, the characteristics of the channel and, in some cases, the energy consumption is estimated. This estimation is carried out by calculating the time spent by a node to realize a task, which is linked to a known power consumption. The testbeds are used to test the throughput of MAC protocols and routing protocols. Moreover, the reliability of the applications is also tested. However, real WSNs are submitted to different conditions of the physical channel. The research community is also interested to measure the energy consumed by a node integrated in a real WSN. Then, power meters are created to measure directly the current required by the nodes.

1.4.1 Simulators

The WSN simulators are generally used to verify the efficiency of routing or MAC protocols. They work properly when the goal deals with comparing several protocols and to reach conclusions about the way to improve the characteristics of these protocols. Moreover, high quantities of nodes in a network can be simulated. However, the estimated consumption values are often inaccurate if the current values considered are taken from datasheets. They can signifiantly differ in real environment: the hardware components behave differently when they work alone or in a circuit, with other components. The current requested by each component can modify the current level in all the circuit.

As explained in [69], the WSN simulators are divided in two fields: those who are used to any type of communications network, included the WSNs (generic networks simulators) and those who are specifically created for WSNs.

1.4.1.1 ns-2

ns-2 [70] [71] is an open-source event-driven simulator created in 1989 with the purpose of studying the behavior of computer networks. Later, the mobile ad-hoc wireless networks were included in this simulator. ns-2 is the most widely used communications networks simulator. It is implemented in C++ and it includes some models and protocols. The extensibility of ns-2 has been an important aspect to its success because of the development of new tools by the research community. We find some constraints in ns-2, mainly for mobile networks, as a hard scalability due to exponential simulation time slowdown. Also, ns-2 contains a simple energy model which decrements the initial amount of energy stored in the battery of a node for each transmitted or received packet. Some energy values are established for each transmitted or received packet. The nodes are actives until the accumulated energy of transmissions and receptions is higher than the initial amount of energy stored in the nodes. We can consider that this method is a first approach to determine the node lifetime, but the results are really far from a real situation.

Several extensions based on ns-2 platform were developed in order to simulate the behavior of WSNs.

SensorSim [72] is a simulation framework for sensor networks which provides new power and communication protocol models from ns-2. Its power model is divided in several models, one for each hardware component in the node. The energy storing device is a battery while the energy consumers in the node are the RF module, the microcontroller and various sensor devices as a geophone, an infrared or a microphone. These component models and the sensor function model, containing the network protocol stack and the sensor protocol

stack, are represented in Figure 1.9. Sensorsim has been programmed as an event-driven simulator, then, the battery model is activated when three different events occur: the battery level changes because of the action of the energy consumers, the battery level reaches the zero or the battery level reaches a certain threshold. Besides, the model of the most energy-hungry device in the node, the RF module, is composed by three parts: transceiver, amplifier and antenna (antenna characteristics to calculate the distance of transmission but not as energy consumer). Moreover, five different operating modes are defined to elaborate the energy model: transmit, receive, idle, sleep and off modes. According to all of these elements, an energy model is created for a node and for the sensor network. However, SensorSim suffers from the same scalability problems as in ns-2.

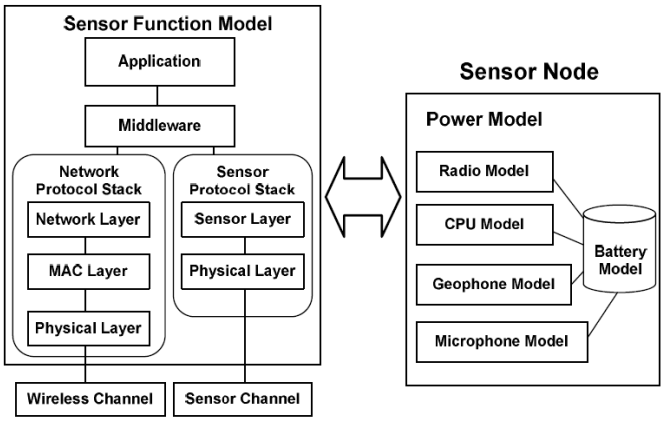


FIGURE 1.9: Structure of Sensorsim [72].

NRL Sensor Simulator [73] corresponds to a more recent simulation framework for WSNs based on ns-2. In this case, the power values of the operating modes in the node are determined by the user but this energy model does not include as many different operating modes as in the previous frameworks.

1.4.1.2 OMNET++

Objective Modular Network Test-bed in C++ (OMNET++) [74] is an open-source discrete-event general purpose network simulator based on C++. This simulator is extensible and modular. The elements used to create the simulated programs in OMNET++ are called *modules*. The *simple module* is a basic unit of execution. These simple modules are organized in *compound modules* which are linked by connections. The top-level of the modules are called *network module* and the communications between modules are established by messages.

Numerous extensions and frameworks have been created for OMNET++. Sensor Simulator (SenSim) [75] is one of the most popular extensions in OMNET++. WSNs simulations are possible with this software, which includes energy models representing a simple approach

of a battery in the node and, also, energy consumers as RF module and CPU. The battery discharge depends on the operating modes in the energy consumers. Then, the CPU model contains the idle, sleep and active modes whereas the radio model is composed of idle, sleep, transmit and receive modes. The battery model assumes that the battery is not damaged with age and there is not self-discharge. The structure of the nodes represents each layer of the protocol stack. Moreover, according to the structure of the network (communication between nodes), SenSim introduces the Sensor Node stack which simulates the behavior of the Application Layer and the Wireless Channel Model, which represents two types of signal propagation: free space propagation model and the two-ray ground reflection model.

Castalia [76] is a more recent simulator based on OMNET++ and dedicated to WSNs and Body Area Networks (BAN). New elements are included in this simulator as a custom modulation allowed by SNR-BER curve, interference handled with received signal strength or the mobility of the nodes. The energy model in Castalia is simple but it introduces new parameters as delays switching between the different operating modes in the radio module and in the CPU module, multiple transmission power levels and node clock drift in order to synchronize the nodes.

1.4.1.3 TOSSIM

TOSSIM [77] [78] is a code level WSN simulator, it is created as a library of the TinyOS [79] distribution. TOSSIM is a discrete-event simulator which simulates entire TinyOS applications based on MicaZ [80] platforms. The propagation loss for each pair of nodes are defined in the wireless channel model. Loss values are introduced from real measurements or calculated by a theoretical model. RF noise and interference in the channel are also simulated. In this simulator, the layers upper than MAC layer in protocol stack are not treated. TOSSIM does not include an energy model and, this is why, PowerTOSSIM has been created.

PowerTOSSIM [81] is developed as an extension of TOSSIM. It presents an energy consumption model for each node in the network and for each hardware component in the node, based on the current values introduced by the user. These values represent the current values for each operating mode of the components in the node and they are introduced from component datasheet or from real measurements, as the authors present in [81]. PowerTOSSIM uses TinyOS and TOSSIM component model to track the power consumption of the nodes. The high-efficient scalability of TOSSIM allows PowerTOSSIM to obtain the energy consumption in the simulation of a network with thousands of nodes. An error from 0.45% to 13% has been identified for different TinyOS applications. Moreover, this simulator can show the energy values for each component in a current-time graph.

1.4.1.4 Avrora

Avrora [82] is an instruction-level sensor network simulator implemented in Java which improves the flexibility and portability with regard to the previous simulators. Avrora has an operating system independence by simulating machine code and its cycle-accurate process simulation improves the precision with respect to other simulators as TOSSIM. Moreover, according to [82], the performance analysis of Avrora is up to 20 times faster than another cycle-accurate instruction-level simulator as ATEMU [83]. It does not contain an energy consumption model, this is why, an extension has been developed in order to reach this energy consumption evaluation and a lifetime prediction of the network.

AEON [84] completes the energy consumption study in Avrora. The first task to realize in AEON deals with measuring and adding in the simulator the current values of the operating modes for each hardware component included in the node. Then, the behavior of all the hardware components is emulated. The operating modes are related to the OS code instructions, therefore, the energy consumption of each instruction of the code is calculated. A current draw of the node is represented graphically to verify the energy consumption of application code. An energy profiling is created according to the OS code in each node and, after that, AEON predicts the node lifetime as well as the network lifetime.

1.4.1.5 WSNsim

WSNsim [69] is a discrete-event based simulator for WSNs. It has been created to study the energy efficiency of the WSNs. The nodes of the network are represented with different colors in a graphical user interface (GUI) in order to show the energy remaining in the storage devices. Different models have been developed in WSNsim to approach as much as possible to the real WSNs. The Energy Model is composed of energy sources, energy stores and energy consumers. The energy sources or energy harvesters contain a photovoltaic cell and a vibration harvester. The energy stores are composed of a battery, which simulates the dynamic rate discharge and self-discharge effects, and a supercapacitor, which suffers from higher leakage. The energy consumers modeled are the microcontroller, radio module, temperature and light sensors, Analog-to-Digital Converters (ADC) and the real-time clock (RTC). To model these energy consumers, the simulator takes into account the current value, the voltage and the duration of the corresponding state. An environmental model and a channel model are also determined. The environmental model includes physical parameters as the temperature, the light and the vibration. These three parameters influence the values obtained by the sensors and have an impact on the energy model, mainly in energy sources. Besides, the channel model is similar to the one used in Castalia simulator. Furthermore, the SNR depends on the temperature because this model takes into account the thermal noise.

The list of WSN simulators presented here is not exhaustif. There are other simulators used by the research community as WSNet, Cooja, ns-3, J-Sim, among others.

1.4.2 Testbeds

In Section 1.4.1, we have described some WSN simulators. The algorithms integrated in the simulators may reach a really high level of complexity. Different hardware components are included in the nodes, with several different operating modes for each component. Moreover, the WSNs simulators include models in order to get close to the real WSNs. Among these models, we find environmental models and channel models but, also, energy models. The use of these algorithms is interesting because they allow to evaluate rapidly the behavior of a WSNs with many nodes.

In real life, it becomes a very complex task due to several causes: the development of hundreds or thousands of nodes in a same place entails high cost and long time; each node in the network is programmed by a different firmware depending on the functions of this node; the emulation of some parameters, as the transmitted packets or the energy consumed by each node, leads to great complexity, mainly when the number of nodes increases; real channel is composed of physical effects that are difficult to control, as fading, shadowing or interference, among others.

In spite of these issues, the study of the behavior of a WSN in real life has always been a source of interest to the research community. As Imran *et al.* expressed in [85]: «*physical testbed strive to bridge the gap between simulation and real deployment*». This is why different testbeds have been developed around the world.

One of the most popular WSN testbeds is the MoteLab [86], developed by Harvard University. It was originally composed by 26 Mica2 [87] motes but later these motes were substituted by 190 Tmote Sky [88] sensor nodes. These motes work with TinyOS and are remotely programmed by the user through a web-based interface. The main interest of the development of this testbed is to test the programs created by users around the world in a physical experimentation scenario. The users can establish their own system design, communication protocols and applications in order to analyze the behavior of the network and reach conclusions.

Another testbed, SensLAB [89], created by Inria and LSiiT (Strasbourg) and included in FIT IoT-lab [90] platform, is composed of 1024 nodes distributed in four different places in France (Lille, Grenoble, Lyon and Strasbourg) and interconnected by Internet. The users program remotely the nodes for any Operating System (OS). This testbed includes different motes with different radio technologies as IEEE 802.15.4 (in 2.4 GHz and 868 MHz bands) or IEEE 802.11

in order to offer a rich variety of possibilities in the experiments. MAC protocols, routing protocols and applications are tested in a controlled environment because the activity of the nodes in this environment is known every time. Moreover, mobile nodes are included using train toys. The users can also test geolocalization protocols with these mobile nodes. The geolocalization is carried out by the fixed nodes via the Received Signal Strength of the packets sent by the mobile nodes. Another important tool of this testbed is the estimation of the node power consumption. The power consumption of the hardware components states in the node is known from the datasheets. The energy consumption of the experiment is estimated by calculating the elapsed time in each state.

Another example of experimental testbed is GNOMES [91], a low-cost testbed created to evaluate the properties of heterogeneous WSNs. This testbed includes an elaborate energy model with different types of batteries and a solar cell. They run different scenarios depending on the operating modes of the components in the nodes. The lifetime of a GNOMES node is estimated for each one of these scenarios as well as for three types of batteries.

1.4.3 Power Meters

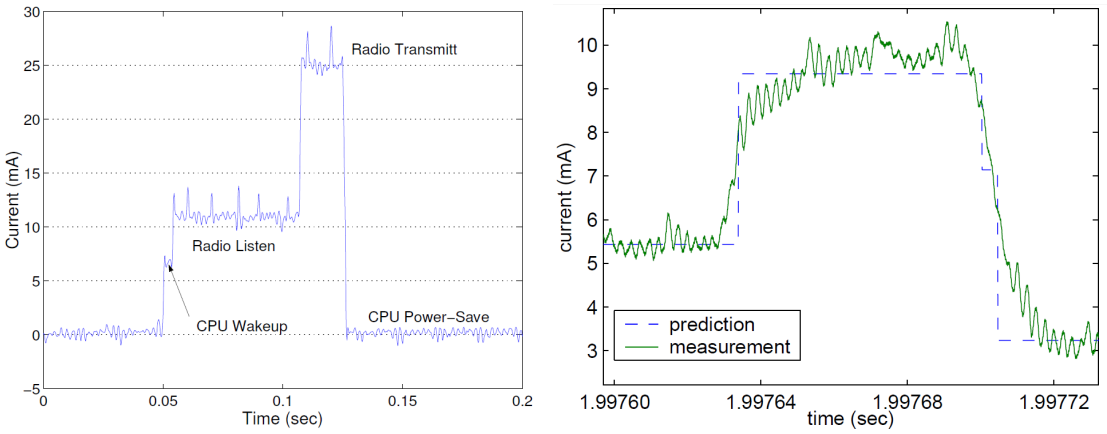
Real experiments are also needed to obtain the real energy measurements from the nodes of WSN. Different methods are employed to achieve these measurements.

1.4.3.1 Oscilloscope

The most typical method consists in using an oscilloscope in order to obtain the measurements of the electric potential difference between the terminals of a resistor placed in the supply input of the circuit.

Such a method is used in [92] to obtain the energy measurements in a sensor node for six different operating modes: sleep, packet transmission, LED activation, sensor value reading and two other states from the combination of these operating modes. All these states are identified on the screen of the oscilloscope. The average energy consumption is calculated for each hour and, then, the lifetime is easily estimated.

Figure 1.10a, from [81], represents the current in a Mica2 node when transmitting a packet. Only two components are actives in this test, radio module and microcontroller (CPU). Oscilloscope measurements allow to distinguish four operating modes. These values are inserted in the energy model of the simulator. Then, the energy consumed by a node is simulated according to its program, written in TinyOS functions. In AEON [84], each hardware component in the node has its energy model. After a processing period, the algorithm carries out



(A) Current measurements to include in the energy model of PowerTOSSIM [81].

(B) Current measurement and current prediction in AEON [84].

FIGURE 1.10: Current measurements in WSN simulators acquired by an oscilloscope.

an energy analysis, an energy profiling and, finally, the current draw of each component after the simulation time is represented (see Figure 1.10b).

Besides, Healy *et al.* [93] offer an interesting study about the energy consumption depending on the OS for WSNs. They compare four different OS: TinyOS, Mantis OS [94], SOS [95] and T2 [96] (TinyOS version 2). An oscilloscope measures the voltage over a one ohm shunt resistor placed between the battery and the power supply input of the node. From these energy measurements, the lifetime of the node is calculated for the same application program running in the different OS on a MicaZ mote supplied by a pair of AA batteries. The paper concludes with T2 as the most energy-efficient OS.

In [97], Courtay *et al.* create an energy consumption model for a WSN, which depends on the number of links between nodes. Before this, the current draw defined by the energy model of ns-2 simulator is compared with the real energy measurements obtained by an oscilloscope for the same application. The mote used in this case is the iMote2 platform [98].

In [99], the authors study the energy cost of the changes in firmware of WSN nodes. The voltage measurements on the terminals of a ten ohm resistor are made. Indicators are inserted in the firmware with the purpose of identifying the change of function in the energy measurements. These indicators are represented in the oscilloscope as trigger events, as shown in Figure 1.11, and help to calculate the average energy consumed in the entire node for each function. Generic energy model and energy measurements are embedded in Loosely-coupled Component Infrastructure (LooCI) [100] middleware to create a specific energy model. LooCI allows the reconfiguration of the node according to the channel and to environmental conditions. The work shows that the reconfiguration of the program in the node is energy efficient in some cases, even if the reconfiguration itself consumes extra energy.

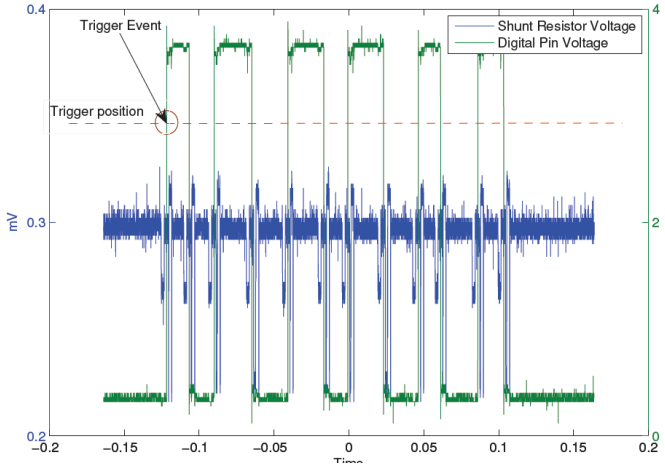


FIGURE 1.11: Voltage measurement with trigger event example made by an oscilloscope [99].

Oscilloscopes allow accurate measurements because of the high sampling frequency (in some cases higher than 20 GSa/s). However, the memory necessary to store the values limits the duration of the experiment. Besides, oscilloscopes cannot be easily used for measurements in a real environment and do not satisfy our purpose because they are not easy to transport and their cost is so high to use different oscilloscopes in different nodes.

1.4.3.2 Data Acquisition System

Data acquisition systems (DAQ) are able to measure the real physical conditions and to convert these conditions in sampled signals and store them before post-processing. This type of systems are created to work with high sampling frequency and to store large quantities of data. Moreover, the measurements are done with a remarkable precision.

In [101], the authors carry out different energy measurements on a TelosB [102] mote. For this purpose, they use a commercial DAQ, the DT9816 [103], from DATA TRANSLATION. This device contains six independent ADCs with a 16-bit resolution and a sampling rate of 50 kHz per channel. In this case, they do not use a shunt resistor placed in power supply input of the circuit, but a current-to-voltage converter, which offers the voltage values from the current requested by the circuit under evaluation. The current measured in the node is compared with the values represented in the respective datasheets of the hardware components. In microcontroller (TI MSP430), the different low power modes are energetically evaluated for two different clock frequencies (1 MHz and 4 MHz). The current values as well as the period of time for writing, reading and erasing in flash memory of the microcontroller are also evaluated. Moreover, the current draws of these functions in flash memory are represented with a 50-kHz sampling rate. The most important part of this paper corresponds to the study of the current in radio module (TI CC2420). The energy is evaluated in the ensemble of microcontroller and radio module because the DAQ measures the current in the whole circuit.

We observe that the results differ with datasheet even if the current of the microcontroller is added, showing that datasheets do not exactly describe the behavior of the node.

Macii *et al.* presents in [104] a study about the reduction of energy consumption by searching an optimal synchronization between nodes. To acquire the energy measurements, they use two digital multimeters, one for obtaining the voltage values in terminals of the battery and the other for measuring the current drain in battery. These multimeters are connected to a computer which collects and analyzes all the values. In this work, the authors use also an oscilloscope to verify the accuracy of the multimeters. We observe in the results a good accordance between theoretical model, simulation results and measurements.

1.4.3.3 Hand-made Power Meters

Oscilloscopes, DAQs, multimeters and other equipment are used to measure the energy consumption. The measure in these instruments are based on shunt resistor method. Other researchers opt to create their own power meter or to integrate a power management unit in the node itself. We present in this section some examples of hand-made power meters based on shunt resistor method and on other different methods.

In [105], the energy measurements in the node are made by a sensor node management device (SNMD) which includes a 16-bit ADC (AD7654). This device is composed also of a charge unit, responsible for battery recharge. It includes a battery monitor and charge controller (DS2770) capable of measuring the current through a 0.1 ohm resistor. Then, there are two possible ways to measure the current requested by the circuit. These energy values are recovered by a microcontroller integrated in the SNMD and sent to a host system via a USB connection. The authors of this paper use the energy measurements to complete an energy model which is based on a finite state machine (FSM). In our work, we present a similar approach: an energy consumption model made automatically from real measurements. The main difference is that in [105], the energy model is created before the measurements and, in our study, this energy model is made by an algorithm.

L. Barboni and M. Valle [106] use two different devices to measure consumption: a current shunt monitor, INA139 [107], that amplifies the voltage across a shunt resistor and, then these voltage values are measured by an oscilloscope; and a Coulomb Counter LTC4150 [108], which counts the units of charge (mAh or μAh) that arrive to the circuit coming from the battery. Mainly, the coulomb counter is a voltage-to-frequency converter (VFC). In the output of this device, we find narrow pulses that represent the fixed units of charge. The value of the units of charge can vary and is configured using a calibration process before the experiment. For example, in this study, the authors employ 5.2 μAh as unit of charge, *i.e.* each pulse in

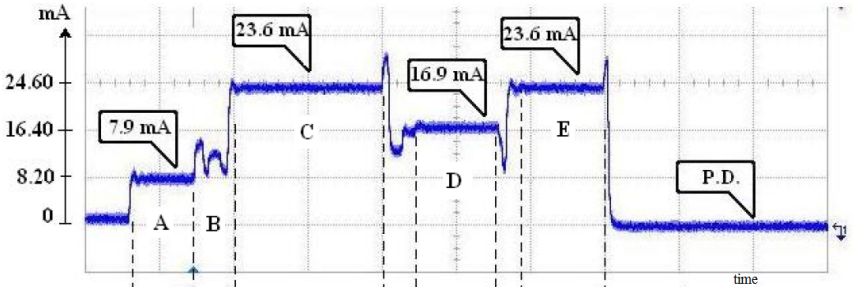


FIGURE 1.12: Current draw with different operating modes in MicaZ [106].

the output represents 5.2 μAh of energy consumed by the circuit. Figure 1.12 depicts a current draw of this study, where we can differentiate several operating modes in a MicaZ. In state A, the microcontroller wakes-up; in B, it wakes the radio transceiver up; in state C, microcontroller communicates with the transceiver and the MAC layer functions are executed; state D corresponds to the transmission of the packet; in E, the transceiver is in reception mode; and finally, the circuit goes back to power down mode. With this type of draws, the energy consumed in each state as well as the energy employed for different WSNs applications are calculated. Furthermore, the same authors complete their study in [109] with the estimation of the battery lifetime. The decrease of battery voltage in relation with the energy remaining in battery is also evaluated. The authors consider that the MicaZ mote, with a CC2420 [110], as RF module, and an ATmega128L [111], as microcontroller, is not operational when the voltage power supply drops below 2.4 V. They present a program in the node where a 25-bytes payload packet is transmitted each 50 ms with an output power level of -25 dBm, which corresponds to a current equals to 8.5 mA. The battery lifetime obtained by the authors of this study for all these components is about 30 days.

In [112], the platform used to obtain the energy measurements in the experiments is MSB-A2 [113] sensor node, created by the authors of the study. This circuit integrates a coulomb counter LTC4150. The values can be stored in a SD card or can be transmitted to a computer via a serial communication. The circuit includes the LTC4150 but, for this work, the authors prefer to use another method because of the low resolution of this coulomb counter. This is why, they take another MSB-A2 platform to use the ADC integrated in its LPC2387 [114] microcontroller for the energy measurements. The results, in this case, are still not as good as expected because the measurements include a lot of noise. In order to avoid this problem, the authors treat the data by two different methods, as seen in Figure 1.13: by applying a moving average and by using a Kalman filter. This paper just show the current values of the whole circuit for two applications, without going into details about the calculation of energy consumption or the estimation of the node lifetime.

E. P. Capo-Chichi and H. Guyennet present in [115] a hybrid WSN composed of two types of nodes: a Reduced Function Device (RFD) which is very simple and acts as an end device,

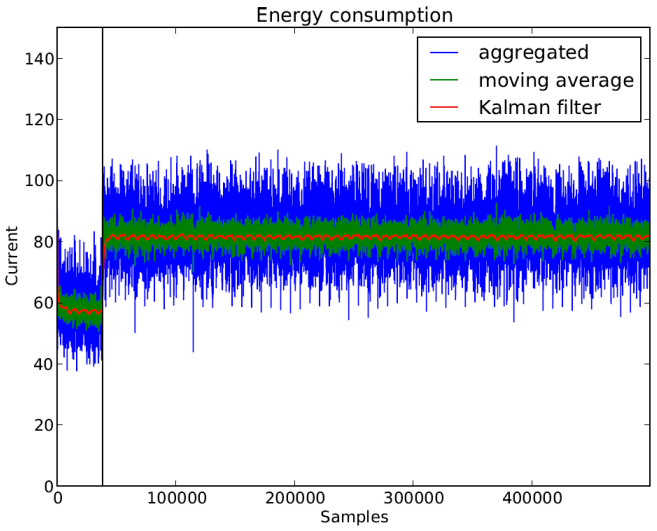


FIGURE 1.13: Current draw by using MSB-A2 platform [112].

represented by a Tmote Sky; and a Full Function Device (FFD) which corresponds to a more complex and less energy-efficient platform, represented by a FOX board [116]. The FFD requires a high capacity to process data and it can communicate with a RFD, with another FFD or with the coordinator node of the network. The RFD is forced to communicate only with a FFD. In this study, FOX Board is completed with a GPS receiver and an energy analyzer integrated in the circuit. The energy analyzer is placed in the power supply input of the circuit and measures the current consumption of the whole circuit. Figure 1.14 shows an example of the current measured by this energy analyzer in two cases: when FOX Board communicates via IEEE 802.15.4 or via Bluetooth. We notice that FOX Board has a current of 0.12 A in idle mode. We observe some peaks in the transmissions of IEEE 802.15.4 packets whereas the current for the Bluetooth RF module stays around 0.14 A for all the experiment. The authors conclude that the most energy-efficient use of one of these technologies depends on

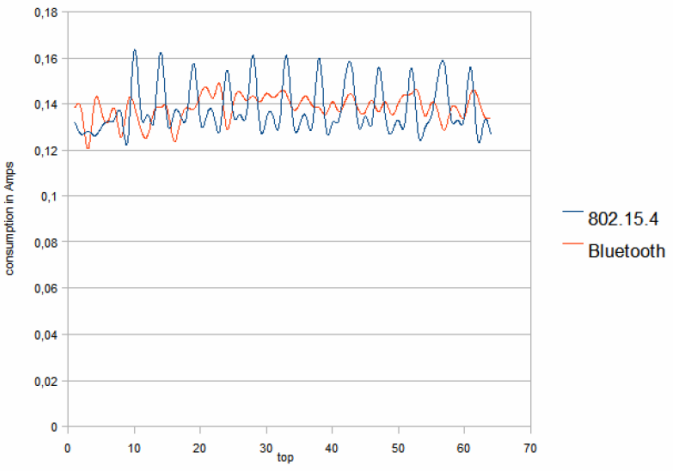


FIGURE 1.14: Comparison of current draws for IEEE 802.15.4 and Bluetooth technologies in FOX Board [115].

the application, but they emphasize the importance of using the sleep mode in IEEE 802.15.4 technology.

Another interesting study is realized by Jiang *et al.* [117]. They create the scalable power observation tool (SPOT). This device measures the power consumption in a node of WSN with a dynamic range of 45000:1, a resolution of less than 1 μA and a temporal resolution of several microseconds. SPOT is created to be attached to the circuit under evaluation, it becomes a part of the WSN node. The architecture of SPOT is divided in four parts: sensing, conditioning, digitization and energy output, as shown in Figure 1.15. The heart of the platform consists of a voltage-to-frequency converter (VFC) which receives the electric potential difference amplified from the terminals of a shunt resistor. This shunt resistor is placed between the battery and the power supply input of the node. The energy output part in the architecture includes an energy counter, which counts the number of narrow pulses generated by the VFC in function of the voltage level in its input. The energy output part also integrates a time counter. The values in the counters are transmitted to the same evaluated node via an I2C communication. An analog lowpass filter is placed before the VFC voltage input in order to avoid the effect of noise generated mainly by the VFC oscillator. Both counters, energy and time, work simultaneously. Some experiments show the accuracy of this system with an error less than 3%.

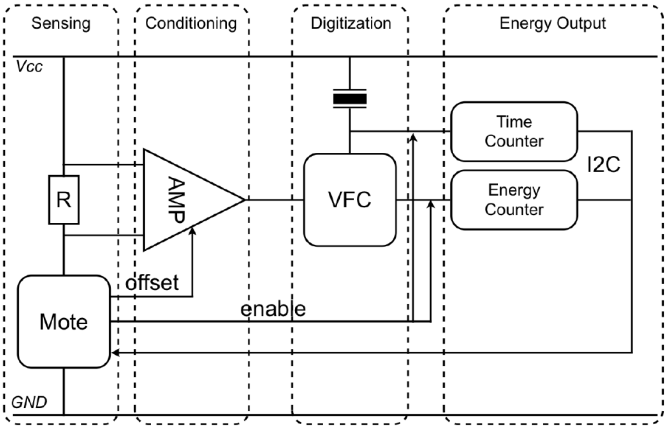


FIGURE 1.15: Architecture of SPOT [117].

One year later, in 2008, some of the authors of SPOT presented iCount [118], a simple but very efficient power meter. iCount consists of a pulse frequency modulated switching regulator instead of the VFC used in SPOT. In this case, the pulses correspond to the switching cycles generated by the regulator. The count of these cycles is carried out by an externally-clock counter installed with the microcontroller of the node under evaluation. As shown in Figure 1.16, a simple wire connected between the LX output and the counter is enough to create a real-time power meter. The Maxim MAX1724 [119] has been chosen as switching regulator in iCount due to its linearity between switching frequency and load current after bias compensation. The dynamic range of the power meter is 100 000:1 (from 1μA to 100

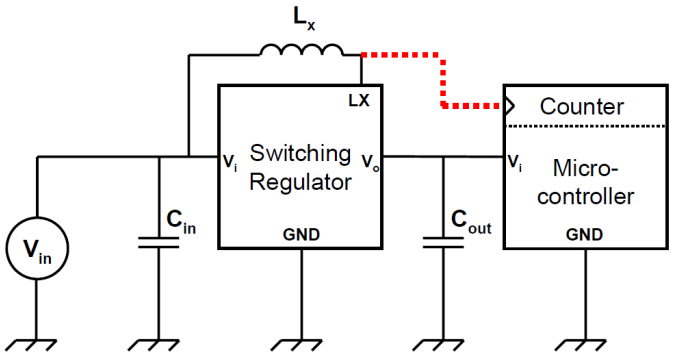
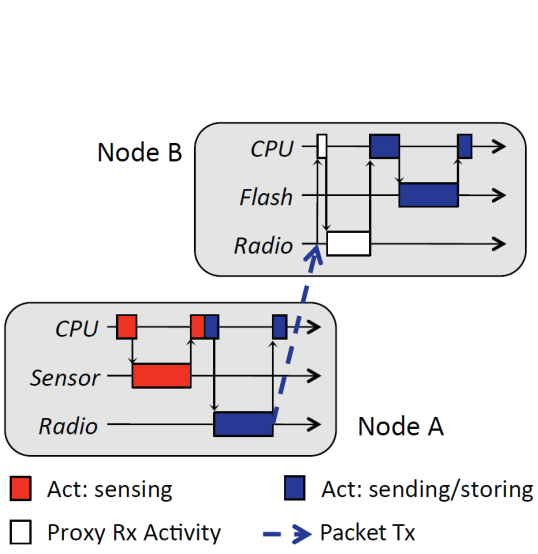


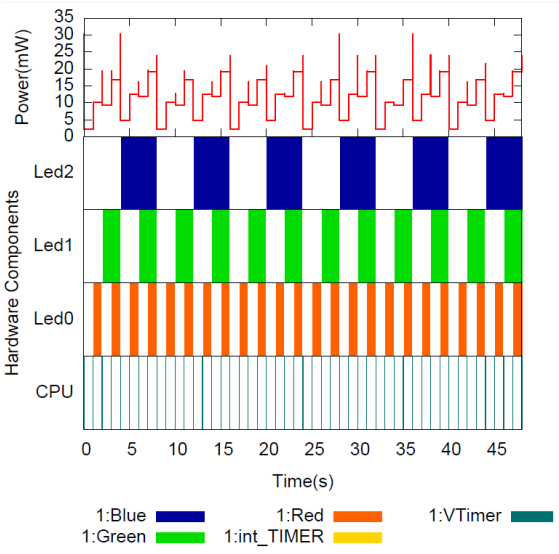
FIGURE 1.16: Circuit for iCount with a switching regulator, a microcontroller and a counter [118].

mA) whereas the resolution varies with the sample rate. The maximum sample rate reaches 66 kHz but it depends on the chosen resolution. For instance, for a resolution of 10 μA, the sampling rate decreases to 80 Hz. The maximum error is $\pm$15% over five orders of magnitude. For the experiments, the authors use a Moteiv Tmote [102] and a driver software is created especially for iCount. This driver is implemented in TinyOS and allows the node to execute some functions as start, pause or read values.

In [120], Quanto is presented as a time and energy profiler for embedded network devices. With the purpose of obtaining the energy measurements from a WSN node, the authors use iCount power meter included in the HydroWatch platform [121]. Quanto consists of several functions of code integrated in TinyOS. The OS includes different interfaces in order to track the energy in the node, as *PowerState* interface and *PowerStateTrack* interface. In PowerState



(A) Activity tracking in components of two nodes.



(B) Example of power draw measured by Quanto.

FIGURE 1.17: Activity tracking and power draw with Quanto [120].

interface, the program in microcontroller obtains from iCount the power values of the current state. Besides, the PowerStateTrack receives real-time events when a change of state occurs. Quanto carries out the activity tracking of each hardware component in the node, as in the example presented in Figure 1.17a. Once the activity of the different components in the node is known, Quanto determines the power consumption for each component and for each activity thanks to the measurements from iCount. The authors show the reliability of Quanto with a calibration. For that, they test the Blink application in TinyOS (blinking three LEDs) and compare the results with the current measurements obtained by a digital oscilloscope. The relative error between Quanto and oscilloscope measurements in this test is 0.83%. The power consumption measured by Quanto for this application is shown in Figure 1.17b. As observed in Figure 1.17b, Quanto is able to separate the global power measurements obtained by iCount in the power consumption of the different components in the node (CPU, LED0, LED1, LED2). The results are shown for a simple application blinking three LEDs, what does not mean that the error may be higher in a more complex application using other components as the RF module.

In [122], R. Zhou and G. Xing present Nemo, an *in situ* power meter for WSNs. It is a noninvasive, plug & play platform which is installed between the battery and the power supply input in the node. Nemo includes a shunt resistor switch composed of five resistors (0.1 Ohm, 1 Ohm, 10 Ohm, 100 Ohm and 470 Ohm) that are selected by four MOSFETs. The dynamic current range offered by this system is of 250 000:1, which is 2.5 times better than in iCount and 5.5 times better than in SPOT. The resolution depends on the measured current value because, when the system measures higher currents, the selected resistor in the shunt resistor switch has a higher value and, thus, the resolution decreases. However, by using the resistor with lowest value, the system achieves a minimal resolution of 0.013 μA. This allows to obtain a high accuracy in power measurements. The sampling rate can reach 8 kHz when a compression algorithm is used in the measurement data or 100 kHz, when this algorithm is not implemented. As shown in the Nemo architecture, Figure 1.18, the electric potential

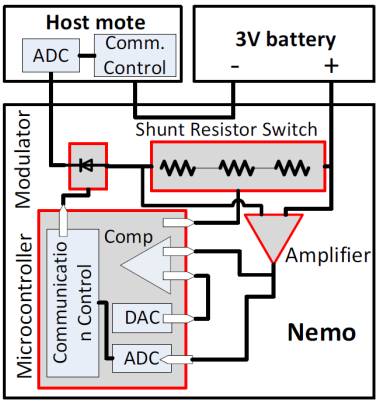


FIGURE 1.18: Architecture of the Nemo power meter [122].

difference in the terminals of the resistors is amplified and, after that, is measured by the ADC embedded in a microcontroller. These values are transmitted to the microcontroller integrated in the host node via a real-time, high-speed bi-directionnal communication based on a current/voltage modulation. The current load and the supply voltage from the battery are modulated and transferred thanks to power-line communication (PLC) technology [123]. After several tests, the average measurement error is 1.34%.

To be complete, we have to consider the energy consumption of Nemo system itself because this system is supplied by the same battery as the node. This is why, Nemo firmware contains a sleep mode that consumes 150 μA whereas the active mode reach up to 4.6 mA. Nemo can identify when the node under evaluation is in sleep mode; at this moment, Nemo continues to measure during some instants and, afterwards, goes to sleep mode. It keeps this last energy value until returning to active mode. Nemo wakes up immediately after the host mote and continues to measure the power consumption in the node. The authors offer an interesting study of the sleep current consumption of three TelosB when the temperature changes in the environment. This study shows the high measurement accuracy of Nemo for the low current measurements (tens of μA). We observe in this study that the temperature can clearly influence the energy consumption in a node of WSNs and, thus, the lifetime of this node.

1.4.4 Simulators - Testbeds - Power Meters

In this section, we present the characteristics of simulators, testbeds and power meters in WSNs.

Simulators are important to know approximately the behavior of a entire network, even with thousands of nodes. The tasks executed by each node can be modified relatively smoothly. In some simulators, the conditions in the physical channel are estimated and, moreover, the energy consumption is estimated taking into account the tasks that each node executes. However, simulators present some limitations. Among them, the lack of accuracy because the process is estimated and not real; the phenomenons that appear in the physical channel; and the energy values taken from datasheets of hardware components. For that, some researchers use the experimental testbeds and power meters.

Previously, we present some examples of testbeds which help to study the behavior of a WSN in experimental environment conditions near the real life. The energy consumption is estimated but they lack the real energy measurements in order to study the real behavior of the hardware components in the node.

Power meters are necessary to obtain real energy measurements from WSN nodes. The accuracy of these measurements depends on the hardware comprised in the power meter. With

power meters, it is possible to measure the impact of the channel conditions in the energy. Table 1.2 presents the advantages and the limitations of simulators, testbeds and power meters. We will see in the contents of this work that one of our main goals is to create an energy measurements platform. This platform is connected to a real node which is integrated in a real WSN and placed in a real environment. The energy consumption of this node is measured without affecting its proper functioning. This platform will be presented in Chapter 2.

TABLE 1.2: Comparison of simulators, testbeds and power meters.

	Simulators	Testbeds	Power meters
Advantages	- Easy development of software - High number of nodes	- Real hardware - Controlled environment - Tests of protocols	- Real hardware - Real environment - Real applications
Limitations	- Theoretical approach	- Not real environment	- Complex deployment

Generally, the power meters evaluate the energy in the whole node. In our work, we present a power meter which measures the current required by each electronic component separately in the node. This offers an important information about the behavior of each component according to the code programmed in the microcontroller. Another interesting contribution of our work is the automatic realization of an energy consumption model from the real energy measurements. This model will be used to be included in a simulator in order to simulate a WSN with a large number of nodes. This is why simulators, testbeds and power meters are important to achieve the energy optimization in WSNs.

Chapter 2

Energy Measurements Platform and Experimental Results

2.1 Introduction

Many WSN are developed around the world. Some WSN have been created to academical purposes and others are used in the industry. The academical WSN are used by the research community as testbeds in order to verify the performance of the new algorithms developed in the different protocol stack layers (routing protocols, MAC protocols, PHY layer algorithms). The industrial WSN are created to develop a real application which supplies a determined need in the society or in the world. In the present study, we create an energy monitoring platform to obtain the energy measurements from the real nodes installed in existing WSN. These energy measurements will be analyzed later to reach conclusions about the way to optimize the nodes as well as the network as a whole. The analysis of the energy consumption is carried out by taking into account both, hardware and software in the node. This platform is called *Synergie platform*.

2.2 Synergie Platform

A platform has been created in IRCICA laboratory to obtain and analyze the energy measurements in a circuit. This platform is called *Synergie: versatile system for the energy measurements in wireless sensor networks*. This system is composed of different parts. As depicted in Figure 2.1, the first part is a hardware power meter that measures the energy consumption from a real WSN node. This power meter recovers the values of the currents required from

the electronic components in the node separately. This analysis of component per component allows to verify more in detail the energetic behavior of each part in the node as well as this behavior according to the software programmed in microcontroller. Some indicators before some important actions are included in the program of microcontroller in order to relate these actions with the energy profiles in each component. These indicators are transmitted to the energy measurements platform through the Digital Input/Output (DIO) pins in microcontroller. This architecture and the obtaining of energy measurements is developed in Section 2.2.1.

Once the real measurements are stored as current values, we carry out the analysis of them. An energy consumption model based on Markov chains is created from the current measurements. The mathematical algorithms which realize this model are called *Automatic Energy Consumption Model*. This corresponds to the second part of the Synergie platform These algorithms are based on statistical processes and create a Markov model that estimates the lifetime of a WSN node. Then, the estimation of the lifetime is the third stage of the Synergie platform. This process is presented in three different ways, all of them are based on the Markov chains. We will explain this estimation below.

The fourth stage of Synergie platform corresponds to the optimization of hardware and software in a WSN node according to the information obtained from the energy model. This part is still not developed in an automatic way, then, the observations and the conclusions reached by the researcher from automatic energy consumption model results will use to optimize the hardware and the software in the node. A dynamic optimization of the code in the microcontroller of the node is a task to develop in a future work.

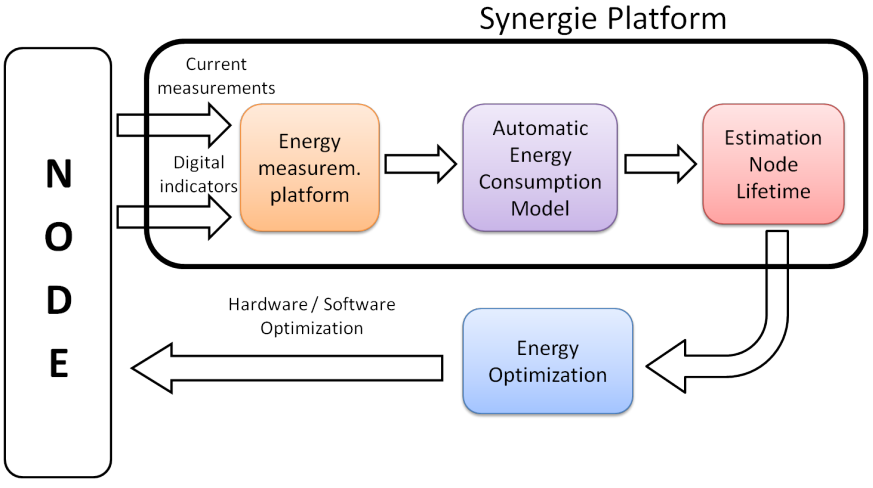


FIGURE 2.1: Outline of Synergie platform.

In next section, Section 2.2.1, we present the architecture of the energy measurements platform use in Synergie platform. Moreover, the first version of this power meter is also explained.

2.2.1 Hardware Architecture

The first part of the Synergie platform consists of a hardware platform which measures the energy consumed in a node of WSN, component by component and function by function. Synergie is able to recover in parallel the current level of several components in a circuit and to represent these levels in a time axis after the treatment of the data samples. In fact, thanks to these measurements in parallel, we can identify easily the different actions of the components in a circuit and, after that, we can change the program in the microcontroller to compare the energy behavior and the code programmed.

The method to measure the energy is the classical method using a shunt resistor before the supply input of an hardware component of the circuit. The difference of the voltage levels at the terminals of the resistor is amplified by an operational amplifier and the result is connected to the input pin of an ADC. This ADC is managed by a microcontroller which is programmed to take the values from the ADC inputs and to send them to a PC via a serial communication. Then, the PC treats these data through the MATLAB [124] software. The advantage of our system is that we take the values separately for several electronic components in the circuit. This method allows to know the energy behavior of each component for each instant. We are able to identify the different actions of each component in the circuit in order to optimize the behavior of the node in terms on energy consumption in a future work.

Some indicators have been inserted in the program of the microcontroller in the node to allow an accurate synchronization of the functions in the program. Each indicator is an instruction in the code before a relevant action, for instance a measure from a sensor, awakening the RF module or transmitting a data packet. Each one of these instructions gives an identification number to each action and offers an output signal with this identification number. Then, the microcontroller in the Synergie platform reads these signals and connect them with the energy values measured at the same instant. We have linked both microcontrollers, the one of the node and the other of the Synergie platform, by several of their DIO pins. The signal with the identification number of the current instruction of the program is transmitted from the microcontroller of the node to the one of the Synergie platform through this link of DIO. Thus, if Synergie platform has the energy consumption of each electronic component and the tracking of the actions on the node, we can obtain and analyze all of these data to know in detail the behavior of the node. Note that the energy consumption of the DIO is not taken into account in this work.

The general diagram of the connections between the evaluated node and the energy measurements system of Synergie platform is presented in Figure 2.2. The maximum number of electronic components under evaluation is represented by n, where n also corresponds to the maximum number of inputs in the ADC in Synergie platform (An). An interface with the

resistors through which the current in each component is measured is located between the node and the Synergie platform. The values of the resistors vary due to the different ranges of current used by these electronic components. For this reason, the resolution of the measurements in the ADC also changes according to these ranges of current (we define *range of currents* as the difference between the maximum and the minimum current used by the component). This means that the resolution for components with a large range of currents will be worse than for components with shorter range of currents. Moreover, we also include the connections of the indicators in Figure 2.2. They come from the DIOs of the microcontroller on the node and go to the DIOs of the microcontroller on the Synergie platform. The number of indicators and, then, the maximum number of functions in code that we can track, is represented as 2^m, where m is the number of DIOs connected between microcontrollers. The microcontroller on the Synergie platform reads the values from the microcontroller on the node every time during the experiment. These values of the indicators along with the current measurements from each component on the node are stored in a external memory placed in Synergie platform or they are sent directly through a serial communication between the Synergie platform and a computer. The computer recovers and analyzes all the information to calculate the current values for each electronic component in the node and the energy consumption.

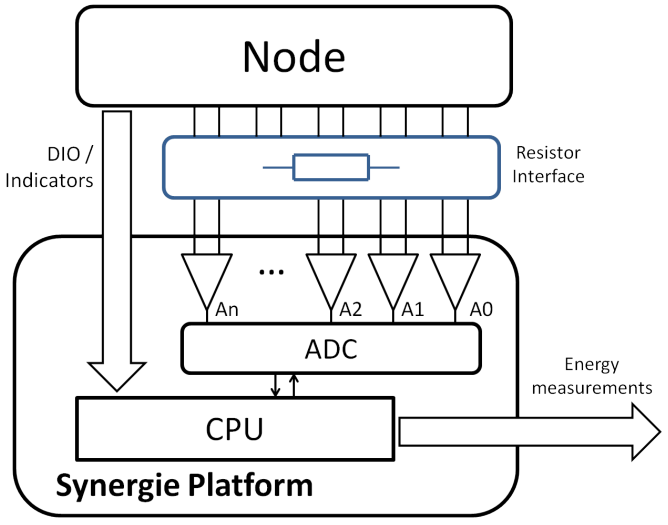


FIGURE 2.2: Diagram of connections between the node and the Synergie platform.

Then, the information acquired by Synergie platform consists in n voltage differences measured by the ADC inputs and 2^m possible indicators acquired by the m DIOs used in the microcontroller. The frame of the serial communication must be the same every time to facilitate the storing and the processing of the data in the PC. This frame is shown in Figure 2.3. The first part of the frame corresponds to the value of the indicator whereas, just after the element, the values acquired in A1,...,An are situated in the frame. In the last position, we add a delimiter to know when the frame is finished. Note that this frame must be as simplest

and shortest as possible to transmit the information as fast as possible through the serial communication.

FIGURE 2.3: Frame sent by Synergie platform to a computer via a serial communication.

The Synergie platform has been created with commercial low-cost electronic components, easy to implement and to use. The heart of the Synergie platform is composed by an Arduino Nano v3.0 with a microcontroller ATmega328P from Atmel [125]. We have used the ZXCT 1086 from Diodes Incorporated [126] as operational amplifiers to obtain the electric potential difference between the terminals of the resistors. An external storage unit, a SD Card from Sandisk, has been added to store the energy measurements in case of not using the serial communication. In this version of the Synergie platform, five operational amplifiers have been included, thus, we can take the energy measurements in parallel from up to five electronic components on the node.

A node has been designed to test the platform. This node is composed of commercial off-the-shelf (COTS) electronic components. In these experiments, we are not interested in consuming as less as possible in energy on the node to the evaluation but to find the elements that optimize this energy consumption. Among these elements, there is the fact to use a new component to change the behavior of the other components; or to choose a new embedded software in the microcontroller in order to minimize the utilization of some actions; or to change the disposition of the nodes in the network.

The electronic components used in the node have been an Arduino Nano v3.0 with a microcontroller ATmega328P, a XBee Series 1 from Digi [127] as transmitting unit, an accelerometer ADXL345 from Analog Devices [128] as sensor, a SD Card from Sandisk as external storage unit and a TR3000 from Murata Electronics [129] as low-power transmitting unit. The XBee Series 1 radio module follows an established standard, the IEEE 802.15.4 standard [130]. This protocol uses the Offset Quadratic Phase-Shift Key (OQPSK) modulation, has a determined packet frame with 100 bytes as longest payload of information and follows a determined Medium Access Control (MAC) protocol. However, the TR3000 transceiver does not have any protocol integrated. It is a low-power RF transceiver which uses an On-Off Key (OOK) modulation but the type of packet frame and the MAC protocol must be determined by the user. Then, this RF module is so power-efficient because the frame, the packaging, the synchronization between transmitter and receiver, the calculation of CRC, the MAC protocol and other tasks must be made by the microcontroller and programmed by the user. This allows

plenty of freedom to the user but complicate the task of the communication between nodes in the network.

The connection between the node and the Synergie platform is shown in Figure 2.4. In this first version, the measurement platform and the node are plugged by several connectors (connectors in green color in the middle of the image). The independent resistor interface does not exist in this first version because the resistors are integrated in the same circuit, just before the input supply of each electronic component. The number of connectors between both circuits corresponds to two connectors for each component (terminals + and - in the resistors) and, also, the connectors intended to the indicators between DIO pins. In this case, we use four connectors for the indicators ($m = 4$), then, up to $2^4 = 16$ indicators can be identified. The supply voltage in node (3.3 V) is independent of the supply voltage in measurement platform.

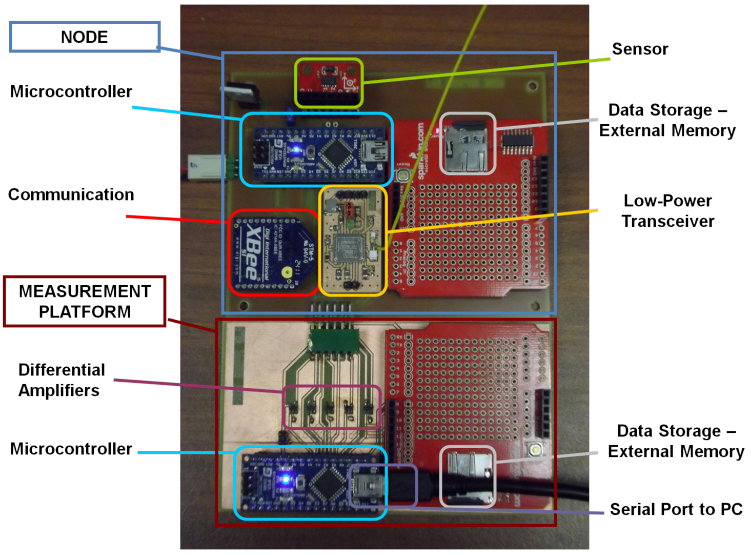


FIGURE 2.4: Node and measurement platform (Synergie) used for the experiments.

The gain of the ZXCT 1086 amplifiers is 50. This gain has been chosen according to the range of currents used by the components on the node. The ADC is included in the ATmega328P. It is an 10-bit ADC with eight possible inputs and a maximum symbol rate of 15 kSps. The chosen value for the resistors $(R_1, \ldots, R_n)$ also depends on the range of the currents in the components, mainly, it depends on the maximum value of the current coming from the datasheet. Then, the choice of the value of these resistors follows the criteria expressed in (2.1).

$$R_n = \frac{V_{\text{out}_{\text{ref}}}}{I_{\text{max}_n} \cdot G} \tag{2.1}$$

where R_n is the resistor situated with the component n; $V_{\text{out}_{\text{ref}}}$ corresponds to the reference output voltage used in the ADC of the measurements platform (in this case, the ADC

is included in the microcontroller); $I_{\max_n}$ is the maximum current value extracted from the datasheet of the component n; and G is the gain of the operational amplifier.

Once the values of the resistors have been chosen, the system is ready to work. The microcontroller of the Synergie platform is programmed to recover the data from the ADC inputs and to send them to an external computer via the serial communication. In the external computer, an algorithm has been created using MATLAB [124] to process these data.

2.2.2 Experiments with XBee Transceiver

An example of the current values recovered and treated by the Synergie platform from the components of the node is presented in Figure 2.5. The energy behavior of each component can be clearly observed. The blue plot corresponds to the current measurements from the microcontroller (Arduino Nano v3.0), the red plot is for the XBee Series 1, the green plot (with low current values) corresponds to the accelerometer and the black plot shows the current values for the SD card. The embedded software executed in the microcontroller describes a typical periodic data collection application in WSN. It consists in the recovery of some data from the sensor and storing them in memory; after some time, these data is read from memory and sent via the RF transceiver to the coordinator node of the network. This network follows a star topology with the end device nodes and a coordinator or sink node which recovers the data from the other nodes. The node under evaluation corresponds to an end device node of the network.

Due to the energy constraint, the utilization of the sleep mode in the components of a node of WSN is necessary. In Figure 2.5, we can easily observe when the components are in the sleep modes, corresponding to the lowest current levels for each component. The lowest current

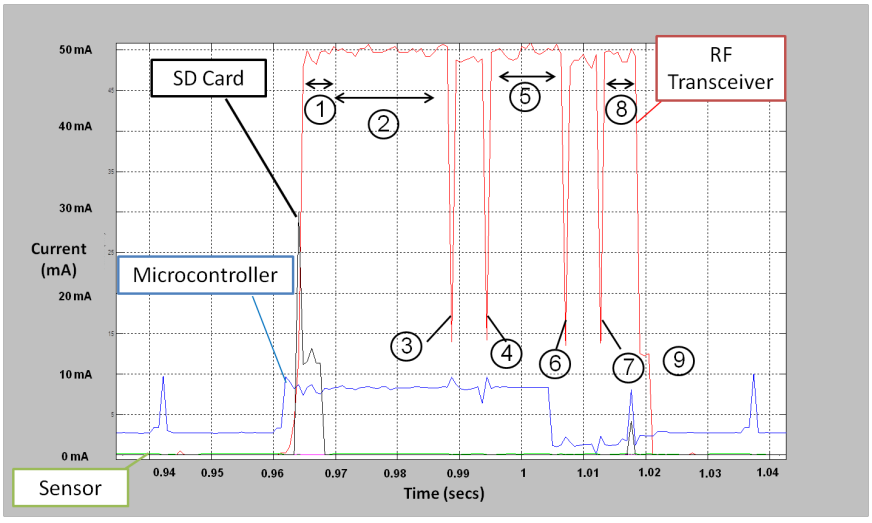


FIGURE 2.5: Current values in time for four components in a node.

levels of the transceiver and of the SD card are almost zero whereas the lowest current level of the microcontroller is not less than 2.5 mA. This shows the poor energy efficiency of the Arduino platform because 2.5 mA of current in a sleep mode is a really high value. In the software, the microcontroller stays in sleep mode until the overflow of its Watchdog timer (WDT). When it occurs, the microcontroller requests a value to the sensor and this value is stored in the Random Access Memory (RAM) of the microcontroller. After some values stored in the RAM, the microcontroller takes all these values from the RAM and writes them in the SD card in order to free the memory space in RAM. After several iterations of this process, all data stored in SD card are read by the microcontroller and sent by the XBee transceiver to the sink node of the network.

This Figure 2.5 represents a zoom on the part consisting in transmitting the information to the coordinator node. We can observe how, before 0.94 s, all the components are in sleep mode. Then, the WDT of the microcontroller wakes the microcontroller up to request a value from the sensor (it corresponds to the small peak in blue plot). Immediately thereafter, the microcontroller returns to the sleep mode. A few milliseconds later, the microcontroller wakes up again to read all the values stored in SD card (see the peak in black plot) and to wake the transceiver up. After that, the time to initialize the transceiver is shown in number ① of Figure 2.5. Number ② corresponds to the time for the first packet to be sent via the serial communication between the transceiver and the microcontroller. The maximum size of the packets is established by the IEEE 802.15.4 standard and is equal to 100 bytes as maximum payload of the packet. In this example, we use this maximum packet size. Number ③ represents the beginning of the transmission of the data packet. In number ④, the transmission is done and the acknowledgment frame (ACK) sent by the receiver is received. Numbers ⑤, ⑥ and ⑦ correspond to the same functions as the numbers ②, ③ and ④, respectively, but for a new packet because in this example the microcontroller reads 200 bytes of data from the external memory. The transceiver in number ⑧ listens to the channel for the last time before going to sleep in number ⑨. Note that the microcontroller, in the blue plot, goes to sleep mode once the last information has been correctly received for the transceiver. This is because the XBee transceiver has a certain degree of intelligence and can store into the memory the order from the microcontroller of *go-to-sleep* mode. The transceiver will accomplish the sleep period after finishing the transmission. We can observe that these measurements are done for four electronic components in parallel.

2.2.3 Experiments with TR3000 Transceiver

Another example of current measurements recovered by Synergie platform is shown in Figure 2.6. In this example, we evaluate the current consumption in a microcontroller (Arduino Nano v3.0 platform), a sensor (ADXL345 accelerometer) and a low-power transceiver

(TR3000). The TR3000 is a transceiver who has a fixed current level for reception mode (3 mA) but a variable current level for transmission mode (in this example, 2 mA). Its power output can be varied with a potentiometer installed before the TX input of the transceiver and, thus, the range to transmit and the current level vary in the same way. In the example, the microcontroller remains active for the whole experimentation period, the sensor continues with its characteristic low current level and the sleep, transmit and receive modes are alternated in the low-power transceiver. In this experimentation, the current level in transmit mode of the transceiver is 2 mA whereas it is 3 mA in reception mode. In parallel, we can observe the different indicators inserted in the embedded software. The indicator 0 corresponds to the sleep mode of the TR3000, the indicator 1 appears in the transmit mode whereas the indicator 2 informs about when the reception mode is active. These two images show a good match between measurements and indicators, what allows to analyze and to change different aspects in the node to optimize the energy consumption.

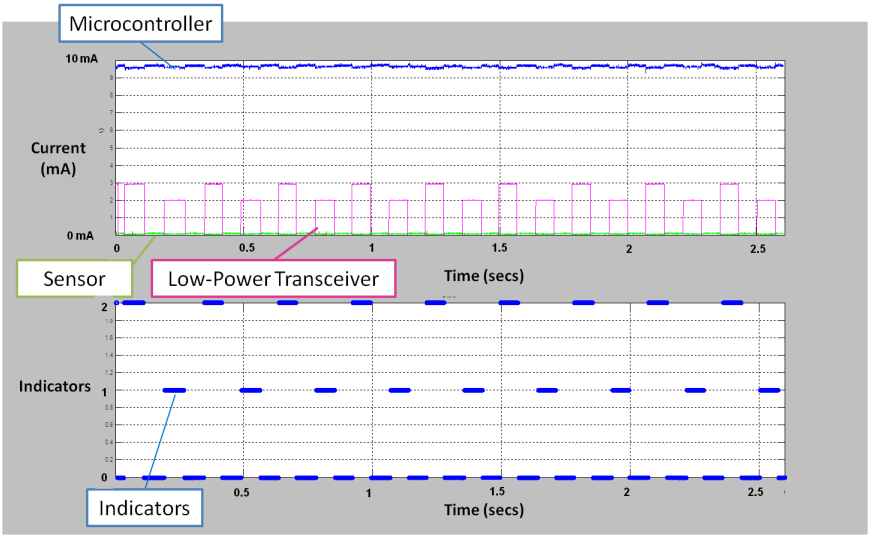


FIGURE 2.6: Current values and indicators measured by Synergie platform.

2.2.4 LoRa Node

The Synergie platform is able to evaluate a large number of types of nodes. In order to measure the current required by each electronic component in these nodes, the process is always the same. An interface with resistors is placed between the node under evaluation and the energy measurements platform. If the node is created specifically to these tests, the traces between the power source and the supply input of a component are connected to two pins in the circuit. These pins are directly connected to the resistor interface. On the other hand, if the node already exists, the power supply trace of a component is cut and two wires are soldered on each terminal of these traces. Each wire is connected to a terminal of the corresponding resistor. The resistor interface has been created separately in order to have a

generic interface where the values of the resistors are changed according to the maximal current value in the hardware component under evaluation.

Figure 2.7 shows a new node of WSN. This node is composed of a transceiver that uses the LoRa technology citeLoRa, a microcontroller, in this case an ATmega328P placed in an Arduino v3.0 platform, and sensors. This node is connected to the resistor interface by two types of connectors: an analog part, where the power supply trace of each component in the circuit is linked to the terminals of the resistors; and a digital part, where four traces recover the digital signals from four DIO in microcontroller. These digital signals offer information about the functions in the code in order to link the software with the current values. These traces with the digital signals are directly connected to the energy measurements platform.

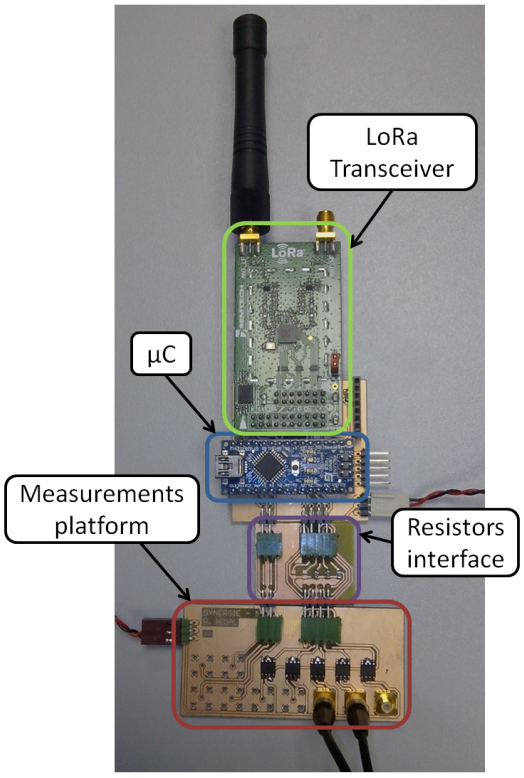


FIGURE 2.7: LoRa node connected to the measurements platform used by NI test bench.

The measurements platform presented in Figure 2.7 has been created to be used with NI test bench, another power meter presented below, in Section 2.5.1. This is why a SMB connector is prepared to be connected to each analog and digital input in this test bench. In this case, we soldered three SMB connectors for three analog inputs but, in total, we could use up to five analog inputs to evaluate five electronic components and four digital inputs. The differential amplifiers chosen for this test bench are the Analog Devices AD8216 [131] amplifiers with a bandwidth of 3 MHz and a gain of 3 according to its datasheet. Several tests has been carried out to verify this gain of 3 and our results have confirmed the correct value indicated in the datasheet-.

In this node, we use the Semtech SX1276 [132] transceiver which includes the LoRa communications technology. This technology belongs to the Low-Power Wide Area Network (LPWAN) technologies which are characterized by a long range, a low power and a slow data rate. The long range parameter avoids the multi-hop topology and, thus, the network layer and the MAC layer issues. If the communication is done only between end device and sink node, the transmissions of the packet coming from other nodes in the network disappear as well as the adaptive routing techniques to find the most efficiency way until the sink. Other phenomenons as the idle listening, the overhearing and the overhead in all the nodes decrease considerably or disappear. The probability of collisions is also reduced because of the diminution of exchanges of packets between nodes. However, if the data rate decreases significantly, the on-air time of the packets increases in the same way and, then, the interference appears with the occupation of the physical channel. All these elements should be analyzed by taking into account the energetic behavior of the transceiver and the microcontroller as well as the energy according to the embedded software. This analysis will be carried out as future work. Moreover, the comparison between the throughput of a multi-hop network using for instance the IEEE 802.15.4 technology and a LPWAN will become a really interesting study, mainly, if we compare the energy efficiency between them for different applications.

2.2.5 Battery Model

From all the experiments, statistical data is obtained: minimal, maximal and mean value of current for each component as well as for the whole circuit. From these current values and with the voltage value used in circuit, we calculate the power values. Moreover, the total energy consumed in the experiment is calculated using (2.2), but we use the discrete form in (2.3) because the values obtained correspond to the samples of the measurements.

$$E = \int_0^T W(t)\mathrm{d}t = \int_0^T V \cdot I(t)\mathrm{d}t \tag{2.2}$$

$$E = \sum_{i=1}^{N} W_i \Delta t = \sum_{i=1}^{N} V \cdot I_i \Delta t \tag{2.3}$$

where E is the energy consumption in the experiment from the initial instant to the end of the period of experimentation (T). $W(t)$ and $I(t)$ represent the functions of power and current, respectively. i represents each sample whereas N is the total number of samples in the experiment. W_i and I_i represent the power and the current value corresponding to each sample and Δt is the time of the sample which is fixed for every experiment.

The total energy consumed and the accumulated energy are calculated for some periods of time in the experiment. The main goal for measuring the accumulated energy in the tests is to

know the periods when the progression of this energy consumption is more or less important. Thanks to the research study of Chen and Rincon-Mora [133] about a polymer Li-ion battery, we have integrated the equation of the open-circuit voltage (V_{OC}) of this battery according to the State-Of-Charge (SOC), represented in (2.4). The representation of the discharge of this battery is shown in Figure 2.8 where SOC = 1 represents the battery in full charge and, consequently, SOC = 0 corresponds to the empty battery. This battery model has been used in the rest of the work, mainly, to estimate the node lifetime.

$$V_{OC}(\text{SOC}(t)) = -1.031 \cdot e^{-35 \cdot \text{SOC}(t)} + 3.685 + 0.2156 \cdot \text{SOC}(t) - 0.1178 \cdot \text{SOC}^2(t) + 0.3201 \cdot \text{SOC}^3(t) \tag{2.4}$$

$$\text{SOC}(t) = \frac{E_{bat}(t)}{E_{init}} = \frac{E_{init} - E_{cons}(t)}{E_{init}} = 1 - \frac{E_{cons}(t)}{E_{init}} \tag{2.5}$$

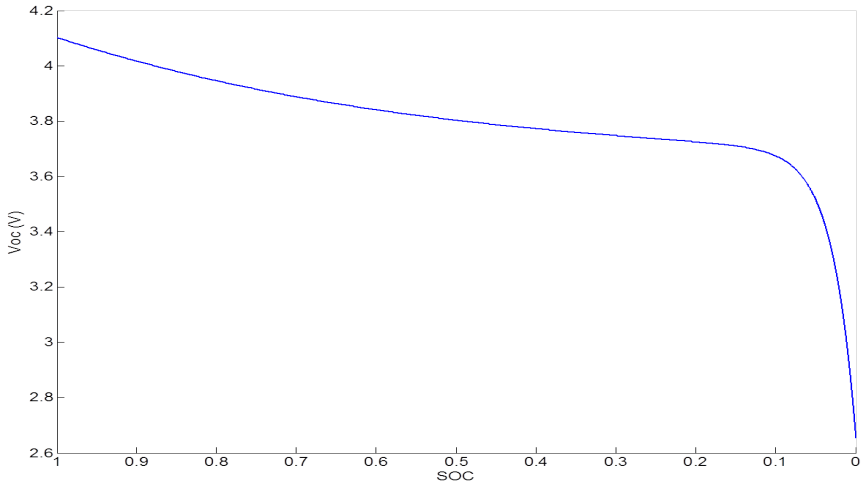


FIGURE 2.8: Voltage in battery discharge.

The SOC in each instant (SOC(t)) is calculated using (2.5). The energy consumed ($E_{cons}(t)$) is calculated from a real node for each instant t. Moreover, the initial energy stored in the battery (E_{init}) is known (850-mAh). We calculate the energy remaining in battery ($E_{bat}(t)$) for each instant from these data (energy consumed and initial energy in battery) and, as a result we have the SOC. From (2.5) and (2.4), we calculate the value of V_{OC} in battery and, if we consider that a circuit is not operable below a certain voltage (for instance, in some cases, below 2.8 V), we can estimate its lifetime.

In next section, we present a comparison of the energy measurement platform of the Synergie platform with a power analyzer.

2.2.6 Comparison Synergie - Power Analyzer

A comparison of the hardware device created in IRCICA laboratory with another instrument is necessary to know the degree of reliability of this platform. We use a Keysight N6705A DC Power Analyzer [134] to carry out this comparison. This power analyzer offers the supply voltage and the ground to the device under evaluation in order to complete the circuit. Then, the analyzer knows the current required and the voltage used by the device all the time. We can use up to four slots in analyzer as four power supply sources with a current and voltage sensibility of 18 bits and a minimum sample period of 20 μs. In data logger mode, it can store the current, voltage and power values for up to four devices in parallel from 20 μs until 60 seconds of time. In this test, we use three slots to analyze the current in microcontroller, radio module and sensor because these devices are the most important in the node. We carry out a test of 30 seconds with a sample period of 61 μs; 30 seconds is enough to measure several cycles of our program and 61 μs is much better resolution than in the first version of measurements platform in Synergie, shown previously. The scheme of this experiment is shown in Figure 2.9. The real node used for this test is presented in Figure 2.10a whereas the connection between the N6705A DC analyzer and this node is shown in Figure 2.10b.

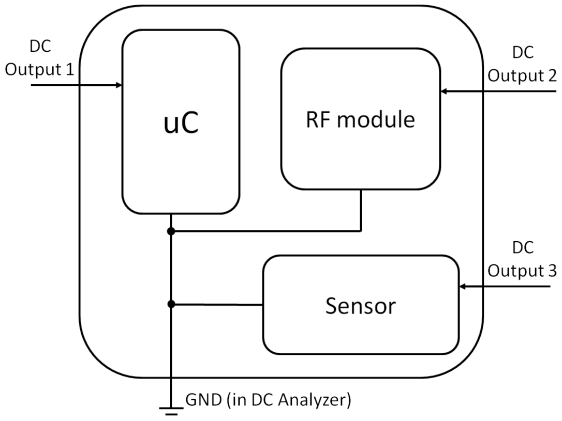
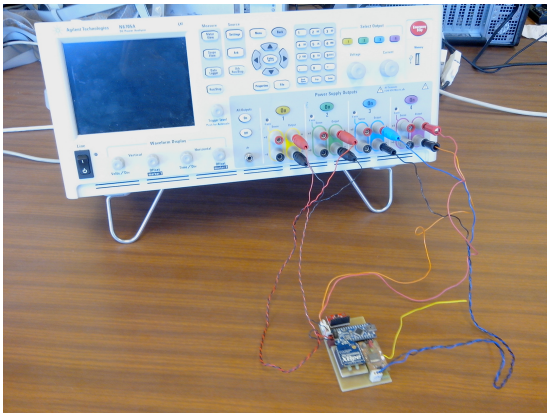


FIGURE 2.9: Schema of the connection N6705A - Node.

The hardware components included in this experimental node are the same microcontroller, RF module and sensor as in node of Section 2.2.1, *i.e.* ATmega328P, XBee transceiver and ADXL345. The embedded software consists in sending a 100-byte payload packet to the coordinator node every 125 ms. Between two transmissions, the microcontroller and the transceiver stay in sleep mode. Figure 2.11a shows a zoom in a transmission period of the current values obtained in the power analyzer for each component where blue plot corresponds to the microcontroller, red plot to transceiver and green plot to the sensor. After several tests, we observe a high noise level obtained in the power analyzer. The sample rate is clearly higher in this acquisition system than in our hardware platform explained in previous section, but the noise level difficult the analysis of the currents. This is why, we calculate the average value in a window of 20 samples and a step of 1 sample for this current values. The

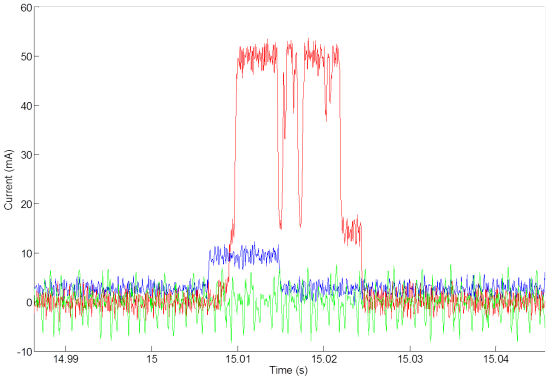
(A) XBee node used in power analyzer tests



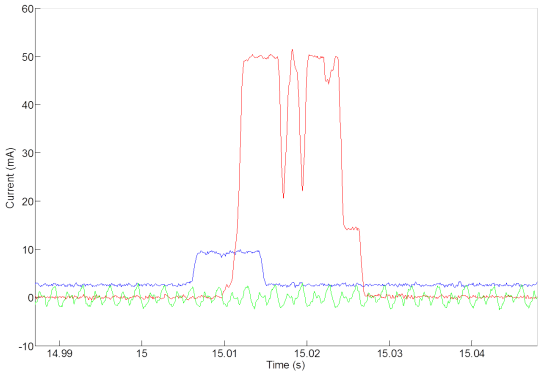
(B) N6705A DC power analyzer with XBee node

FIGURE 2.10: Node and power analyzer for the comparison.

resulting chart is presented in Figure 2.11b. We observe the variation of the values because of the noise in the measure process is clearly lowered. In accelerometer current values (green plot), the variation is still quite significant since this sensor stays in active mode during all the tests and, then, this current plot should stabilize in a value near 140 μs. If we compare current charts in Figure 2.11b with the current values obtained by the measurements platform presented previously in Figure 2.5, we observe that the current values coincide correctly. The



(A) Currents from the N6705A analyzer



(B) Interpolation of currents from the N6705A analyzer

FIGURE 2.11: Currents and interpolation obtained from N6705A DC power analyzer.

sleep mode in microcontroller reaches a current value of near 2.5 mA whereas the active increase up to 9 mA. In the transceiver, the RX part at the beginning of the active period reaches some values between 49 and 50 mA as well as observed in measurements of Figure 2.5. Then, it appears two consecutive inverse peaks until 15 mA when the transceiver sends the data packet. In both series of measurements, Synergie platform and N6705A analyzer, we observe the current behavior of the radio module just before going back to sleep mode where it consumes about 12 mA during 1.4 ms. Thanks to the high sample rate of the N6705A module,

new behaviors in the radio module are observed as in active mode after the transmission where the current decreases until the 42 mA during 1 ms. This version of the measurements platform of Synergie is not able to detect clearly. Even if the N6705A analyzer offers a faster sampling rate, the hardware device of Synergie platform obtains the energy measurements with a high precision, as shown in the results.

2.3 Experiments and Results

Different experiments have been carried out with two types of nodes: a node with XBee transceiver as radio module and the WSN430 node. In this Section, we present these experiments and the obtained results.

2.3.1 XBee Node

The *XBee node* has been presented in Section 2.2.1. This node is composed of an ATmega328P as process unit, a XBee Series 1 transceiver as transmitting unit, an ADXL345 accelerometer as sensor unit, a SD card as storage unit and a TR3000 as low-power transmitting unit. The XBee Series 1 is characterized by including the IEEE 802.15.4 technology that uses the Carrier Sense Multiple Access (CSMA) [135] MAC layer protocol. Some parameters are configured:

- the baud rate in serial communication between microcontroller and transceiver;
- the power output level;
- the IEEE 802.15.4 channel used;
- the identification network address;
- the destination address;
- the number of retransmissions of a single packet;
- the type of sleep mode;
- among others.

In the first experiment, we compare the energy consumption when the size of the packets transmitted by the node changes. This experiment has been realized in a laboratory environment with two nodes: an end-device node (the node under evaluation) and the sink node of the network. The node is composed in this case of the microcontroller and the XBee transceiver. The XBee module is configured with the Pin Hibernate sleep mode, then,

this module remains in deep sleep mode until the arrival of an interruption from microcontroller. This microcontroller stays in sleep mode a time according to the period selected in the Watchdog timer (WDT), in this case, $t_{\mathrm{WDT}} = 16$ ms. After this period of t_{WDT}, the microcontroller takes a value from the accelerometer and stores it in Random Access Memory (RAM). Then, the microcontroller wakes the transceiver up to transmit all the data stored in RAM (NB_DATA). The energy consumption per bit is compared in Figure 2.12 according to the quantity of values of NB_DATA = [10, 100, 550] bytes stored in RAM and, after that, transmitted. The maximum value of normalized energy per bit (value 1 in normalization) corresponds in this test to 70 μJ. In this result, we consider the energy of the three components (microcontroller, transceiver and accelerometer) for the actions of waking up, taking a value from sensor, storing the value in RAM and transmitting this value to the sink node. Moreover, we

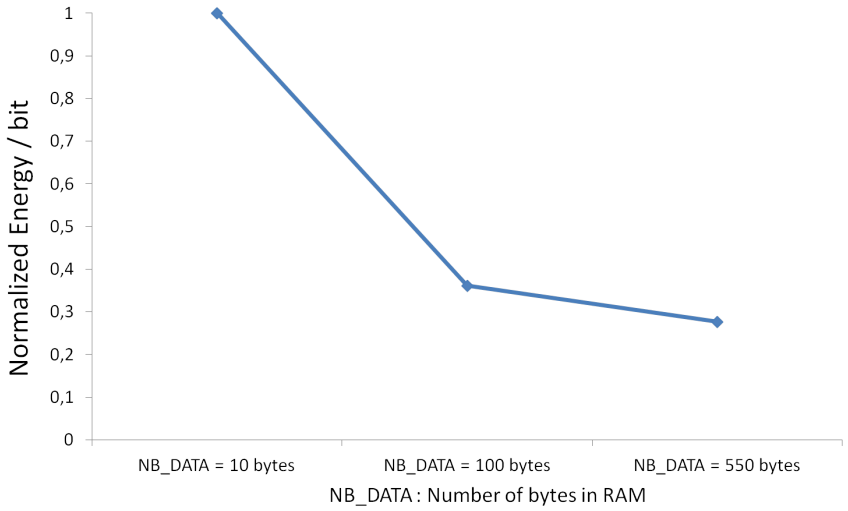


FIGURE 2.12: Normalized energy consumption per bit transmitted according to the quantity of data stored in RAM.

remind that the hardware components used are not the most energy-efficient because our interest deals with comparing different manners to use commercial components, not with using the lowest power components. This is why, the resulting energy per bit is so high in this test. We observe in Figure 2.12 an important decrease between transmitting 10 bytes and 100 bytes. This is because in first case, the transceiver wakes up and sends a packet every 10 bytes recovered from sensor, then, the system spends energy to wake the transceiver up, to communicates the data from RAM to FIFO in transceiver and to sends the data. With 100 bytes, this process is less frequent and, therefore, the node saves energy if the maximum length of the payload in an IEEE 802.15.4 frame is packaged and sent. In the case of NB_DATA = 550 bytes, we observe another decrease of energy consumption per bit but more softly because we need to divide the 550 bytes of data in six packets with their headers and their confirmations of ACK frames. This process of packaging and communication with sink node wastes some energy.

In the second experiment, we include the flash memory SD card as external storage unit in the node. The results are presented in Figure 2.13. In the first instance, we present the results without using the SD card, as in previous experiment. This case is shown in Figure 2.13 as *1-> Without External Memory*. Next values in abscissa axis of the graph represents the number of times that NB_DATA bytes are stored in the external flash memory. These values are presented as NB_PAC = [2, 3, 4, 5, 6, 7, 8, 9, 10]. We show the difference of energy consumption per bit for the three cases: 10, 100 and 550 bytes in RAM. We observe that in NB_DATA = 10 bytes, the energy consumption remains much higher than the others, but we notice a considerable decrease from the first case, without using the external memory, and last case, with 10 times stored 10 bytes, *i.e.* 100 bytes transmitted in total. The difference of energy can reach the 40%. In NB_DATA = 100 and 550 bytes, a difference between NB_PAC = 1 and NB_PAC = 10 exists but it is not so remarkable.

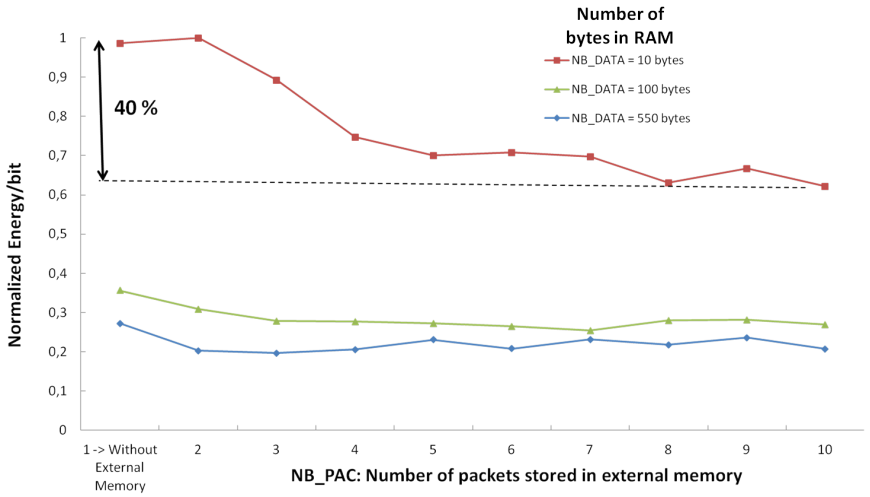


FIGURE 2.13: Normalized energy consumption per bit with data stored in external flash memory.

2.3.2 WSN430

The WSN430 [136] is a node of WSN created by Inria [137]. This node has been integrated in SENSLab testbed [89], a part of FIT IoT-lab [90] platform, a WSN testbed created for research purposes. This WSN430 node is composed of a MSP430F1611 [138] microcontroller, a CC2420 [110] transceiver, a flash memory as external storage unit, an EEPROM serial number and different sensors as a temperature, a sound and an ambient light sensors. The Texas Instruments CC2420 transceiver is configured according to the IEEE 802.15.4 standard and it transmits the packets of information in the 2.4-GHz band. The functions executed by the MSP430F1611 microcontroller in WSN430 can be managed by different operating systems as Contiki, TinyOS, FreeRTOS or Riot.

In this case, the WSN430 node is an existing device created by another research laboratory, then, the process to connect the Synergie platform according to the diagram of Figure 2.2 is different. Firstly, we have to identify the supply inputs in each hardware component that we want to evaluate. For that, we study the printed circuit board (PCB) layout and we cut the via which corresponds to the supply input. Two wires, one of each side of the cut line, are linked to the resistor interfaces. The values of the resistors are chosen according to the maximum current value of the components under evaluation. Then, the resistor interface is connected to the Synergie platform, as shown in Figure 2.14

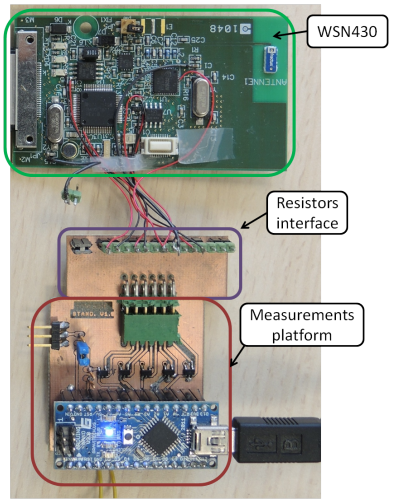


FIGURE 2.14: WSN430 node linked to the energy measurement platform.

For this experiment, we embedded Contiki OS [139] in the WSN430 node. This program consists in waking the transceiver up every 125 ms in order to listen the channel and every eight iterations, *i.e.* every 1 second, the transceiver sends a 19-byte payload packet to the coordinator node. The current profiles of this process in microcontroller and radio module are presented in Figure 2.15. We observe a maximum current value in CC2420 radio module equal to about 16 mA whereas the microcontroller reaches maximum values of 4 mA. The difference of current levels between a sleep mode and an idle listening or a transmitting mode is considerable. This is why the node should remains in sleep mode the most of its time, but it depends also on the desirable performance of the application. Table 2.1 shows the time and the average current values in the transceiver for each operating mode from these results. We calculate that in transceiver, the sleep mode spends the 0.5% of the energy even if this component remains in this mode the 88% of the time. The idle listening mode spends the 37.3% of the energy even if no packet is received whereas the energy consumed in transmit mode is 62.2% of the total energy.

In Section 2.4.2, the energy consumed in the WSN430 node is studied in detail for an application with external interference in the physical channel.

TABLE 2.1: Values and percentage values of time and energy for the operating modes in WSN430 node.

Modes	Time (s)	Av. current (mA)	Iter. in a period	Perc. Time	Perc. Energy
Sleep	0.123	0.0053	8	88 %	0.5 %
Listen	0.0064	9.1845	7	4 %	37.3 %
Transmit	0.0894	7.6631	1	8 %	62.2 %

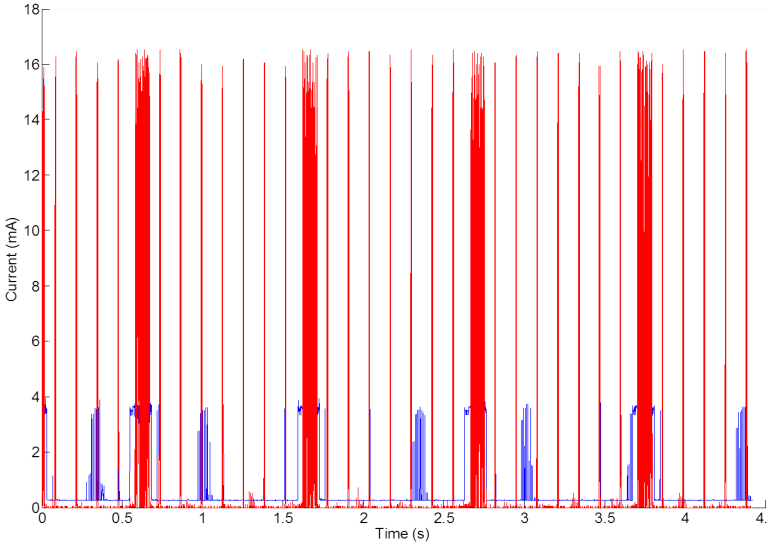


FIGURE 2.15: Current values in microcontroller (MSP430 in blue) and RF module (CC2420 in red) in WSN430 node.

2.4 Impact of Interference on Energy Consumption

The impact of the interference on the Quality-of-Service (QoS) of the communications between nodes of a WSN is of an important interest to the research community. The QoS means the overall performance of the communications that is evaluated according to some aspects as the error rate, the bit rate, the throughput or the transmission delay. The deterioration of the QoS may imply a longer activity in the node to try to send correctly the packets and, then, an increase of the energy consumption in this node.

This study focuses on the impact of interference in 2.4-GHz ISM band on the energy consumption of WSN nodes. The coexistence of several wireless communication technologies in the 2.4-GHz band as Wi-Fi (IEEE 802.11), IEEE 802.15.4, Bluetooth, wireless cameras, among others, represents a major problem in WSN. Other sources of interference as microwave oven in a same environment can also prevent the communication. Figure 2.16, extracted from [140], presents clearly the different wireless technologies which coexist in 2.4-GHz band. The carrier frequencies used by these technologies coincide with all or a part of the 16 channels (from 11 to 26) of IEEE 802.15.4 technology in this frequency band. We observe that

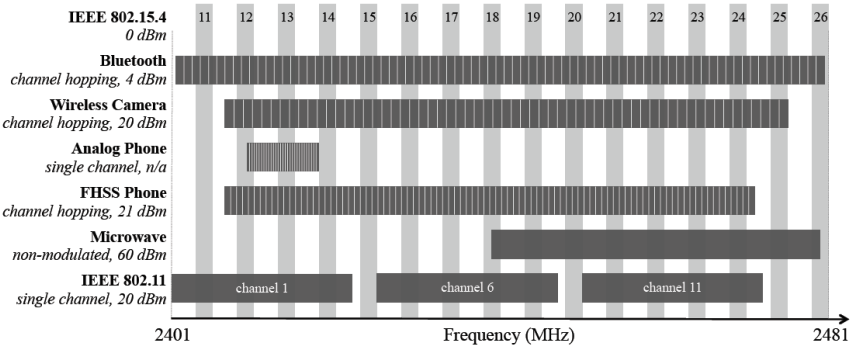


FIGURE 2.16: Wireless communication technologies in 2.4-GHz band [140].

the microwave oven can reach an output power of 60 dBm whereas the Wi-Fi (IEEE 802.11) technology reaches 20 dBm and occupies an important part of the band with its three main channels (1, 6 and 11).

In [141] and [140], this problem is studied by calculating the packet reception rate (PRR) according to the interference level and the technology selected. In [142], the impact of the interference from Wi-Fi, Bluetooth and microwave ovens on the PER of ZigBee (IEEE 802.15.4) transmitters is studied. The case of buildings was proposed. Besides, M. Petrova *et al.* present in [143] an interesting work about the impact of the interference caused by IEEE 802.11g/n technologies in IEEE 802.15.4 networks. In this study, the authors calculate the PRR according to different parameters as the selected IEEE 802.15.4 channel, distance or angle between the nodes and the interference makers. However, in none of these works, the study of the relation between the impact of interference and the energy consumption is done.

The authors of [144] propose a solution to measure the interference level in order to decrease its impact on QoS and energy consumption of WSN nodes. However, the quantitative evaluation of the impact of interference on energy consumption of wireless communication is not provided. The energy consumption of interference measurements part is only evaluated theoretically, using a model from [145].

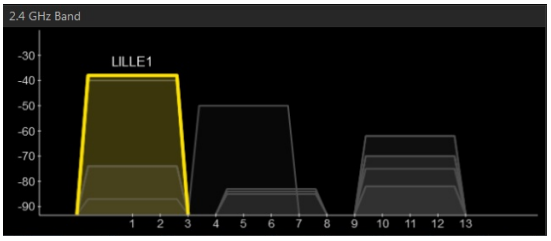
In [146], the increase of energy consumption due to interference is illustrated by calculating the excess energy through a home automation scenario that features a simple retransmission without any radio duty cycle (RDC) protocol, which is important to implement in order to increase the battery life. Moreover, the energy consumption of multiple retransmissions is simply estimated as a product of the energy of single transmission by the number of transmissions.

None of these works neither studied experimentally the energy consumption of the node nor combined the energy and interference measurements. We present an experimental study of the interference from real devices in relation with the energy consumed by the nodes of WSN when the retransmissions and the packet loss occur.

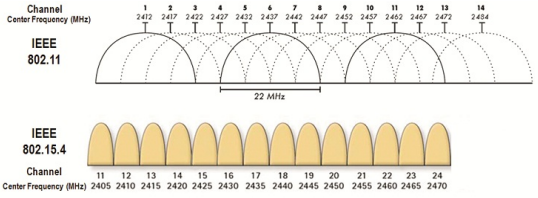
2.4.1 First Results

2.4.1.1 Experiment 1

The first experiment to show the impact of the interference on the energy consumption of a WSN node has been carried out with a XBee node in the IRCICA laboratory. The embedded software of this node consists in sending a packet of maximum size (100 byte in payload) every 1.6 seconds to the coordinator node. This coordinator is located at 3.5 meters from the end device node. A Wi-Fi access point is placed in the same laboratory at 10 meters from the node under evaluation. This Wi-Fi device uses the channel 1 of the IEEE 802.11 standard according to the information offered by inSSIDer Home [147] software application, as shown in Figure 2.17a. This channel, as any IEEE 802.11 channel in 2.4-GHz band, occupies a bandwidth of 22 MHz centered in 2.412 GHz. On the other side, the bandwidth of the IEEE 802.15.4 channels is 2 MHz and each channel has a distance of 5 MHz with its consecutive channels. Then, the IEEE 802.15.4 channels that are perturbed by the IEEE 802.11 channel 1 are the channels 11, 12, 13 and 14, as depicted in Figure 2.17b extracted from [148]. In this experiment, the XBee transceiver is configured in channel 12.



(A) Wi-Fi channels (2.4-GHz band) in IRCICA laboratory with inSSIDer Home software



(B) Wi-Fi and IEEE 802.15.4 channels share the same frequency band [148]

FIGURE 2.17: Wi-Fi and IEEE 802.15.4 channels in 2.4-GHz band

In the present experiment, the first 10 seconds remains without external interference. After that, we add some interference with a smartphone placed at 30 cm from the node and connected to the Wi-Fi access point. While the smartphone download and upload data, we observe visually that the coordinator node does not receive all the packets sent by the end device node.

Figure 2.18 represents the currents in microcontroller (in blue), XBee transceiver (in red) and sensor (in green) for a 60-seconds test with interference in the wireless channel. We observe the part number ①, where the interference does not exist. The system works properly in this 10 first seconds: the radio module wakes up every 1.6 seconds, sends a packet and goes to sleep mode again. In part number ②, the interference appears and the perturbations in current values start. We observe that, in ②, the XBee module tries to send correctly the data packet. According to the IEEE 802.15.4, this means to send a data packet and to wait

for the acknowledgment (ACK) frame coming from the coordinator node. If the transmitter node does not receive the ACK frame, the same data packet is sent again after a timeout period. This process is repeated until the ACK frame is well received or the packet has been sent a maximum number of times. The part number ③ in Figure 2.18, represents a period where interference exists but its level becomes lower than in part ②; we have decreased this interference level in a controlled manner. Then, in ③, some packets are well sent in a first time and others need some retransmissions.

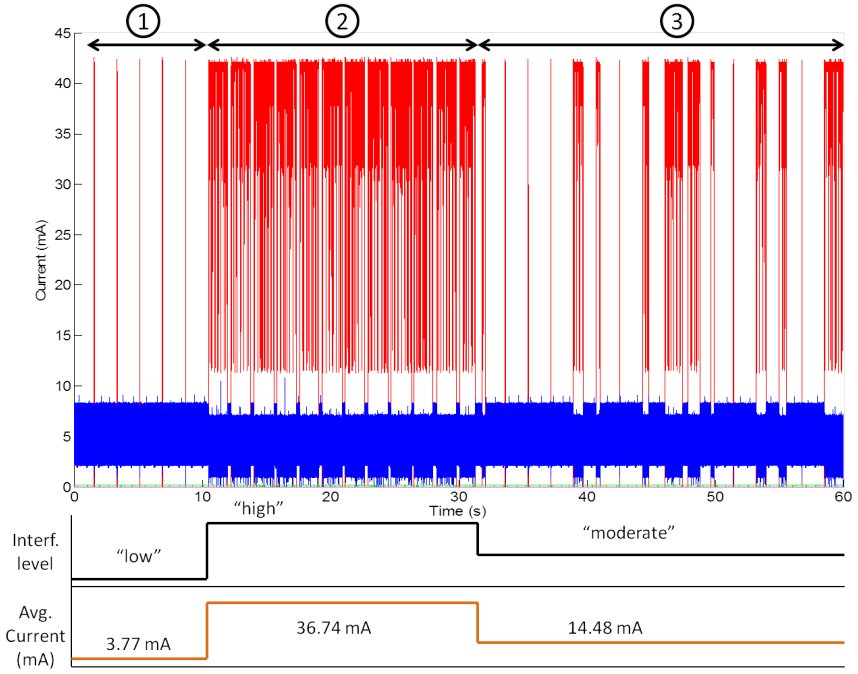


FIGURE 2.18: Current values in XBee node with interference.

The average current for all the circuit in ① is calculated with 3.77 mA while, in ②, reaches the 36.74 mA, *i.e.* almost 10 times more than in ①. In ③, this average current is 14.48 mA, 3.8 times greater than in the case without interference.

2.4.1.2 Experiment 2

In this experiment, we use the same XBee node to compare the energy consumption for different lengths of transmitted packets and for different interference levels in channel. The payload lengths of the packets in the tests are NB_DATA = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110] bytes.

Firstly, the experiment is made in a laboratory environment without interfering nodes which can complicate the transmission of packets between nodes. Later, another XBee node that uses the same IEEE 802.15.4 channel, *i.e.* the same frequency, as the node under evaluation is installed in the same medium. This interfering node sends a 100-byte payload packet

every 100 ms to another coordinator node in a different network. Then, we have two networks that use the same communication technology and the same frequency channel in the same environment. This scenario is closer to reality since, in the future world of IoT, several technologies that use the same carrier frequency to send the packets will coexist in the same physical environment. After that, we add another interfering XBee node. This node is a clone of the first interfering node. This node works in the second network and sends one 100-byte payload packet to the coordinator every 100 ms. We notice that both interfering nodes are not synchronized and, then, it is quite possible that they send a packet in different instants. This fact increases the probability of collisions in the medium and implies extra energy consumption. A third interfering XBee node with the same functions is finally installed in the same environment.

The results are shown in Figure 2.19. The ordinate axis represents the normalized accumulated energy consumption. This normalization is made respect to the highest value in all the cases. The No interf. plot represents the energy consumed for the node when no interfering nodes are present in the medium. We observe that this case is the more energy-efficient. The *Interf 1 Node* chart shows the energy when only an interfering node is added. We observe that the energy consumption does not increase much for one interfering node. The increase of the consumption when two interfering nodes (*Interf 2 Nodes*) are installed is more important whereas, for three interfering nodes (*Interf 3 Nodes*), the energy increases dramatically. In Figure 2.20, we show the increase of the normalized energy consumed by the node with the different interfering nodes with respect to the case without interference.

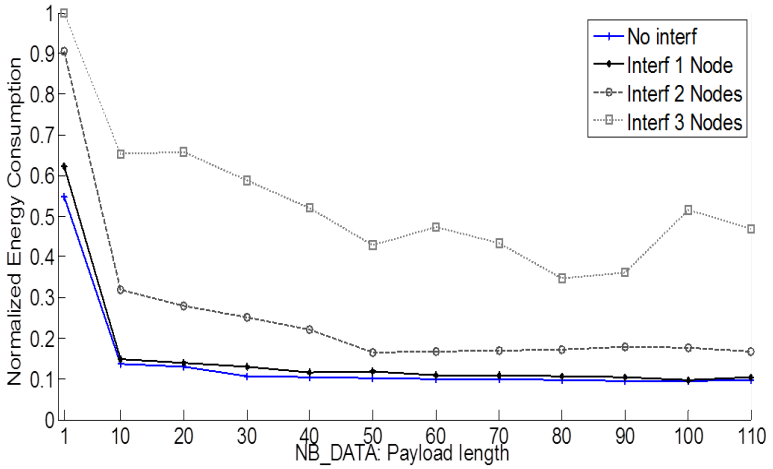


FIGURE 2.19: Normalized energy consumption without and within interference.

Thanks to this experiment, we show two aspects. The first one is that the energy consumption decreases if the size of the packet is optimized, *i.e.* the packet size is near its maximum value (100 bytes for this communication standard). The extra energy consumption in each transmission exists due to two causes: the initialization time of the transceiver when it reaches the active mode; and the header of 28 bytes in each packet. For instance, in the experiment of

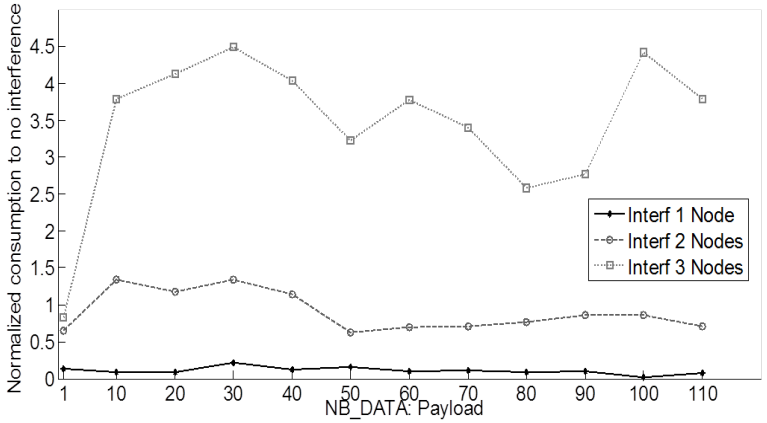


FIGURE 2.20: Difference between energy consumption without and within interference.

1-byte packets, the energy consumption is very high because the header (28 bytes) is much longer than the real information in the payload but, moreover, for the initialization period of each 1-byte packet. On the other hand, the second parameter shown in this experiment is this impact of the electromagnetic interference on the energy consumed by the circuit. In Figure 2.20, we observe that in 1-node interference case, this increment of energy is a stable value of approximately 0.1 times greater than the experiment without interference for all the packet lengths. For 2-nodes interference case, this increase of energy is more changeable because, for smaller packets, this difference reaches 1.4 times greater, while, for longer packets, it remains 0.8 times. In 3-nodes case, the interference levels are totally unstable, then, we obtain greater variations that can reach the 4.5 times more of energy with respect to the experiment without interference.

Next Sections 2.4.2 and 2.4.3 present a more accurate study of the impact of interference on the energy consumption.

2.4.2 IEEE 802.15.4 Interference

This new study of the interference has as main goal to assess the energy loss when the retransmissions of a packet are done. Moreover, after obtaining and analyzing the results, we estimate the average energy consumed by each packet transmitted as well as the lifetime of the node according to the interference level in the physical channel.

2.4.2.1 Experimental Setup

Two sets of devices have been used to complete the experiments of this work: one for the transmission part (TX) and the other for the reception part (RX). Both sets are composed of three different devices each one, as shown in Figure 2.21:

1. *WSN430 node* [136], as the node under evaluation.

2. *TelosB mote* [102] is used to measure the interference in the channel. This device has been chosen to this test because it has a similar hardware structure to WSN430. In fact, WSN430 has been developed based on TelosB. It is programmed with a Contiki code that reads Received Signal Strength Indicator (RSSI) values at the center frequency of the channel used by the WSN430 nodes. This option is implemented in the CC2420 chip, which provides average values of electromagnetic energy measured during 128 μs intervals. The metric distance between TelosB and WSN430 node is 10 centimeters (cm) approximately.

3. *Synergie platform*, as the energy measurements platform. The node under evaluation and the measurement platform are connected by the resistor interface which makes the measurement unit compatible with different WSN nodes. The node is also connected to the Synergie platform via two General Purpose Input-Output (GPIO) lines in order to synchronize energy measurement data with different states of the device (*e.g.* beginning of the retransmission, dropped packet). GPIO lines provide this information directly from the embedded code of the microcontroller of WSN430. The measured values and GPIO codes are transmitted to a computer through a serial link. Then, the synchronized energy measurements from the hardware are analyzed to obtain the current required by each component independently and to assess the energy consumption.

In this work, we analyze the average values of RSSI and energy consumption for each application layer packet (including retransmissions). These values are calculated between the end of previous packet and the end of current packet which are delimited using GPIO signals. The timestamp values are used to synchronize the packets in the measurements.

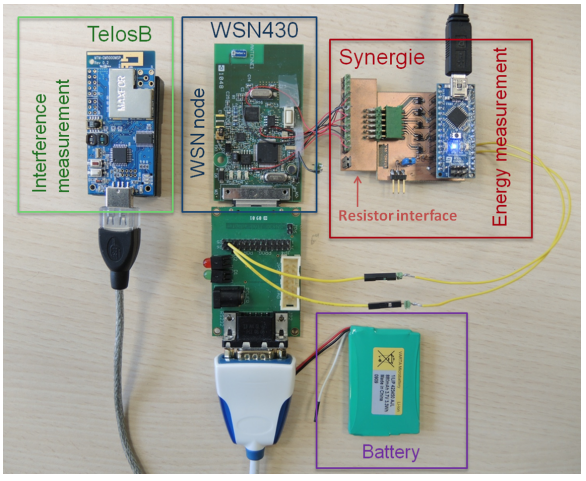


FIGURE 2.21: Devices used in the Interference - Energy experiment.

We have carried out a unicast communication between two Contiki-driven WSN430 nodes: one sender (TX) and one receiver (RX). Both nodes use Rime protocol [149] on a network layer, X-MAC [59] Radio Duty Cycle (RDC) management strategy and CSMA MAC algorithm as parameters of Contiki OS communication stack. In physical layer, the data are included in IEEE 802.15.4 packets. These protocols have been included in this experiment since they are commonly used in real networks. The packets are generated every second by the transmitting node. The size of the payload is 19 bytes. 6 bytes of a Rime network layer header and 15 bytes of IEEE 802.15.4 data link layer header (for 16-bit addressing) are added. Thus, the total size of each packet is 40 bytes. Packets to send are placed in a buffer which can store up to 24 packets in our case.

Following the X-MAC protocol, a transmitting node wakes up when it has a packet to transmit. Then, it sends a sequence of beacons (X-MAC preamble) containing the MAC address of the destination node in order to establish a connection. Once the receiver node is awaken, it decodes the MAC address from the beacon and if it is the packet destination, it sends an X-MAC early acknowledgment (ACK) packet back to the TX node. After reception of the X-MAC early ACK message, the transmitting node starts to send a packet with data, which should be acknowledged by the receiver in case of correct reception. Otherwise, it repeats the procedure of sending the same packet starting from X-MAC connection establishment after a random back-off time. In is important to note that the size of a data ACK message is 11 bytes [150], which is about 4 times smaller that a data packet in our case. According to the IEEE 802.15.4 protocol, transmitting node can have up to three tries to send each packet. If after three transmissions there is still no data ACK message received, the current data packet is discarded and deleted from the buffer. A 125 ms duty cycle is set as a parameter of X-MAC protocol. Then, the receiving node wakes up every 125 ms to check whether there is any packet to be received on the channel.

The experimental measurements have been carried out using two different scenarios. In first case, called *ideal case*, the devices where placed in an anechoic chamber, while the second case, called *real case*, the experiment was done in IRCICA laboratory. In both cases, we use as interfering source three XBee nodes working in the same channel as the node under evaluation. These interfering nodes send a 100-payload packet every 10 ms and are switched on progressively in order to reach different interference level during the experiment. In ideal case, the XBee nodes are placed at a distance of 20 cm from the experimental node while, in real case, these interfering nodes are located further, in different places of the laboratory, as in a real network.

During the experiments, we have collected the packet statistics information as number of retransmissions (on TX side) and number of received duplicated packets (on RX side) for each generated application layer packet. The experimental results of anechoic chamber scenario

are presented as an example in Table 2.2. Each row in this table represents the number of physical transmissions of a single application layer packet on the TX side, with a maximum of three transmissions according to the IEEE 802.15.4:

- *Type of packet '1'* corresponds to the situation when a single packet is sent once, it is well received by RX and the ACK sent by RX has been well received by TX. This is the ideal situation, normally for the lowest interference levels.
- *Packet type '2'* corresponds to two transmissions of a single packet, whether a data packet or an ACK packet is lost.
- The same logic also applies for *type packet '3'*, TX sends three times the same single packet and TX receives the ACK only after the third transmission which indicates that the packet has been well received.
- In *type of packet '4'*, three transmissions of a packet have been done but TX has not received the ACK packet to confirm the well reception. In this case, two options are possible: either one or several data frames are received by RX but the ACK messages are lost or no data frame is well received by RX.

On the other hand, each column in Table 2.2 corresponds to the number of times when a single packet is received by RX:

- *Packet type '1'* describes the situation when a packet is received once, the best situation.
- *Packet type '2'* occurs when a single packet arrives twice to RX due to the loss of an ACK.
- *Packet type '3'* represents one packet received and two duplicates of the same packet.
- *Packet type '4'* corresponds to "no packet received".

The detection of these retransmissions, duplicates and loss of packets, requires an accurate synchronization. This synchronization between TX and RX has been possible thanks to the

TABLE 2.2: Number of transmissions per packet sent on TX side and number of received duplicated packets on RX side by the RF modules. (Example)

		'1' 1 RX	'2' 2 RX	'3' 3 RX	'4' 0 RX
'1'	1 TX	745			
'2'	2 TX	203	24		
'3'	3 TX (success)	131	20	1	
'4'	3 TX (fail)	77	10	0	330

timestamps created by the software ExtraPuTTY [151] as well as the signals generated by the GPIO lines connected between WSN430 node and Synergie platform. Then, synchronized values enable to combine the type of packet on TX and RX sides.

We observe in Table 2.2 that the most of the transmitted packets correspond to the combination '1'-'1', where the used format is 'TX'-'RX', *i.e.* type of packet on TX and in RX sides, respectively. Then, '1'-'1' represents the ideal situation, where a single packet sent by TX is correctly received by RX after first trial and the corresponding ACK is received by TX. Combination '2'-'1' happens when TX transmits a single packet twice and it is received by RX only once, thus, one data frame is lost in the channel, as shown in Figure 2.22. Besides, '2'-'2' means that a single packet was transmitted twice and received twice also, thus, the first transmission has been completed but the ACK sent from RX was lost. This situation appears rarely because of the short length of the ACK frames. The situation '3'-'1' occurs when three transmissions of a single packet are done and only one frame arrives properly to RX. Also, in '3'-'2', three transmissions are present, two frames are well received and one ACK is lost. In '3'-'3', TX sends three times the same packet and it is correctly received the three times, but two ACK packets are damaged or lost in the channel. We observe as well that the event '3'-'2' occurs less often than '3'-'1' whereas '3'-'3' appeared only once in this example. Indeed, when the channel occupancy is increased significantly due to the interfering signals, the probability to lose a data frame is higher than to lose an ACK, by the reason of difference of size, *i.e* on-air

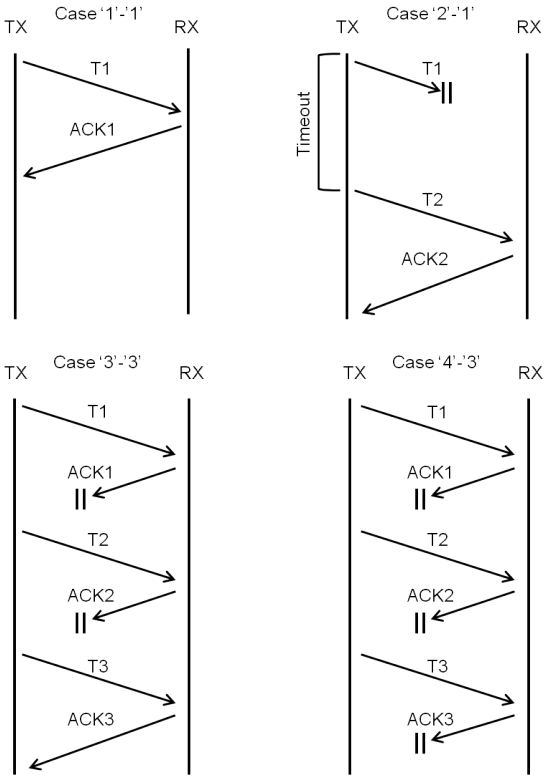


FIGURE 2.22: Examples of combinations of type packets TX-RX.

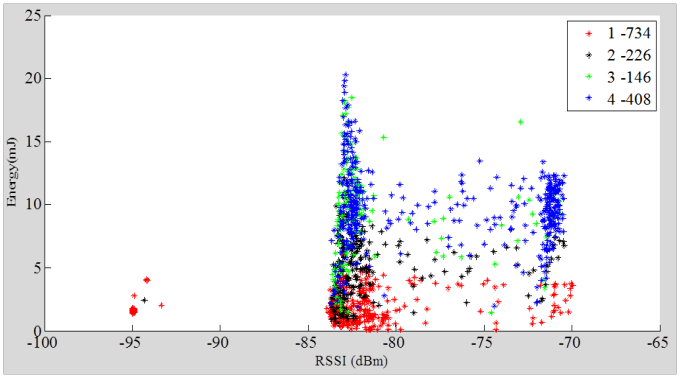
time. However, in '4'-'1', TX sends three frames of a same single packet, one data frame is received by RX but the corresponding ACK does not reach TX node, then, according to TX, this packet is considered as lost. The same situation occurs in '4'-'2' but, in this case, two frames are successfully received by RX and all of two ACK messages are lost. As the case of '3'-'3', the combination '4'-'3' is unlike to appear since the probability to lose three consecutive ACK messages while receiving three duplicated data frames is very small. Therefore, the case of '4'-'3' was never appeared in this experiment. Finally, '4'-'4' corresponds to the situation when the interference in the channel makes the communication impossible. No data packet are received by RX and, then, ACK packets are not sent.

We observe that Table 2.2 corresponds logically to a triangular matrix, because the number of receptions cannot be greater than the number of transmissions.

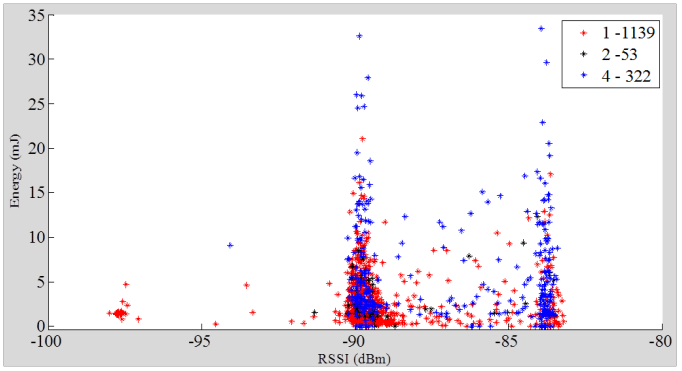
The results and the analysis of these results are shown more accurately in next Section.

2.4.2.2 Experimental Results

We present the Figure 2.23 whose points represent the synchronized measurements of energy consumption of the radio module, interference level and packet transmission statistics. Figure 2.23a corresponds to the energy consumed by the transmitter's radio as a function of measured RSSI for each application layer packet. The different packet types of transmissions (ACK received/not received) explained in Table 2.2 are presented by different colors (red, black, green and blue for packet types '1', '2', '3' and '4', respectively). The number of packets for each type is also presented in the legend of Figures 2.23. In these results, we can notice three distinguished sets of points. The first set is situated in the zone of low interference (around -95 dBm) and low energy consumption. This set contains the packets successfully received after one transmission and only one packet retransmitted once. These packets were transmitted during the first part of the experiment, carried out without any interference thanks to anechoic chamber ideal characteristics, which explains the high first transmission success rate. One-retransmission packet (packet type '2') can be caused by internal problems in the receiver node. Second set of points is situated around interference level of -83 dBm. So, the gap between these two sets of points is bigger than 12 dB. No packets are situated in this RSSI values interval because the first interfering XBee node is switched on suddenly. The second set of points corresponds to the case of one to two interfering nodes. In this case, due to the increased level of interference, we can observe important number of packets retransmitted once or twice (types '2' and '3' respectively) as well as dropped packets (type '4'). Average energy consumption is also significantly increased. The third set of points is situated near the interference level of -72 dBm. This set of points is related to the case of three interfering nodes. In this case we can observe important number of dropped packets prevailing



(A) TX side



(B) RX side

FIGURE 2.23: Energy consumption vs RSSI per packet type in ideal case. A) in TX side and B) in RX side with red, black, green and blue points for packet types '1', '2', '3' and '4', resp.

over other types of packets. Also, we observe that some packets are placed between the second and the third set. This fact is due to the synchronization and interference between the three XBee nodes, because the effect of some packets sent by these devices may be canceled by other packets and the RSSI level can vary.

Similar graph, presented in Figure 2.23b, is obtained from measurements on RX side. In this graph different numbers of receptions (duplicate packets) for every packet, as seen in Table 2.3, are represented by different colors. As in the TX case, three sets of points can be observed. First set is represented by packets received once (without duplication). Second and third sets of points contain important number of never-received packets (type of packet '4'). The average interference level corresponding to these sets of points is lower than in TX case (-97;-90;-83 dBm vs -95;-83;-72 dBm respectively) due to an increased distance between transmitter, interference sources and interference measuring part at the RX side. We observe in Figure 2.23b some packets type '1' (red points) that consume as well as the packets type '4', this fact is due to the extended preamble of some packets (some parts of the preamble are lost) because of the interference in the channel.

We calculate the average energy consumption for every type of packet ('1', '2', '3' or '4') on TX side. These values are shown in Table 2.3.

TABLE 2.3: Results of energy, number of packets and total energy per type of packet on TX side. Ideal case.

Packet Type	'1'	'2'	'3'	'4'
Avg. Energy (mJ)	1.82	5.26	9.36	9.77
No. packets (%)	48.5%	14.9%	9.7%	26.9%
Total energy (%)	17.0%	15.1%	17.3%	50.6%

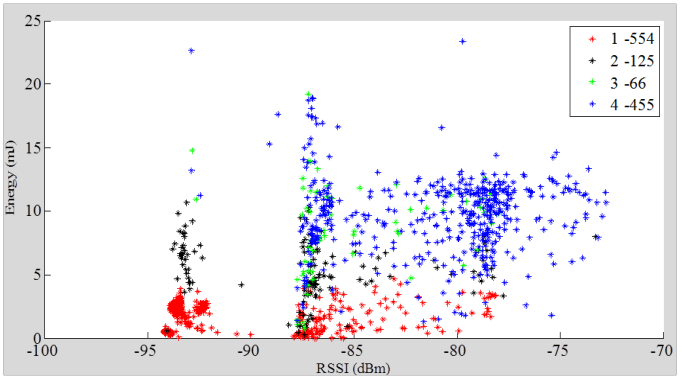
According to the results in Table 2.3, the average energy consumption for a packet retransmitted once is more than double, almost three times greater than the energy of a single transmission. The reason of this energy overhead is the X-MAC connection establishment process. Likewise, in average, the energy needed to send a packet three times is about five times greater than in case of a single transmission. The average consumption of a case of three transmissions without success (no ACK received, packet type '4') is slightly greater than one of successful case (packet type '3') due to the additional waiting time before timeout. Thus, the energy consumption increases in a nonlinear way with the increase of number of transmissions.

In RX side, the results are different. We remind that the packet types in RX is related to the ACK frames lost, what is highly improbable because of their small size. This is why packet type '3' (i.e. three times received the same packet with two ACK frames lost) does not appears in this experiment. Table 2.4 contains the information of average energy, number of packets and total energy. This total energy in TX and in RX side is calculated from average energy and number of packets.

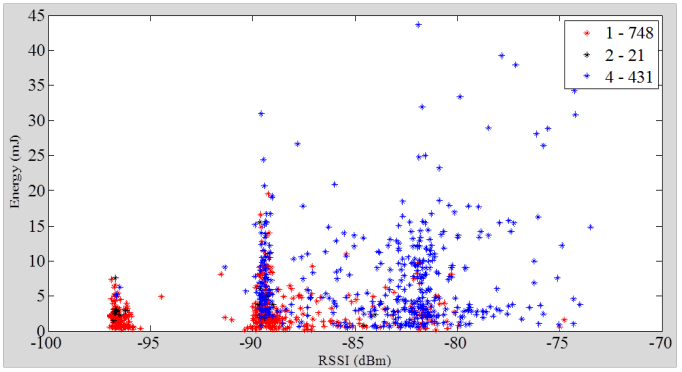
TABLE 2.4: Results of energy, number of packets and total energy per type of packet on RX side. Ideal case.

Packet Type	'1'	'2'	'4'
Avg. Energy (mJ)	2.35	3.19	5.86
No. packets (%)	75.2%	3.5%	21.3%
Total energy (%)	56.6%	3.6%	39.8%

In real case, *i.e.* the experiment in laboratory environment, similar graphs are provided. Figures 2.24 represents different types of packets as a function of energy consumption and measured RSSI on TX and RX sides. As for the ideal case, we can clearly notice three sets of points. Since the conditions of the experiment are different, these sets of points are shifted in RSSI axis in comparison with the set of points in the ideal case. We also notice a more significant variance of RSSI values than in the ideal case due to presence of signal reflections and external sources of interference. The same explanation also holds for the presence of retransmitted and lost packets in the lowest interference zone, where no XBee module is active. In this case, this zone corresponds to the set of points around −93 dBm RSSI level in Figure 2.24a,



(A) TX side



(B) RX side

FIGURE 2.24: Energy consumption vs RSSI per packet type in real case. A) in TX side and B) in RX side with red, black, green and blue points for packet types '1', '2', '3' and '4', resp.

measured by the CC2420 of the TelosB mote. At RX side, Figure 2.24b, we can also observe an increased variance of energy and RSSI. Three different sets of points are differentiated, but we can observe a more dispersed scatter of points. However, the gap between the second and the third set, around −89 dBm and −83 dBm respectively, is not as clear as in the ideal case. This fact is also due to the external sources of interference.

In this case, the statistics of average energy consumption, number of packets and, thus, total energy consumption in TX side per packet type is explained in Table 2.5. We observe that the average energy per packet in real case is similar to the average energy in ideal case, seen in Table 2.3. The value that differentiates the results obtained in ideal case and those obtained in real case is the quantity of dropped packets (type '4'). This is why in ideal case the dropped packets consume near the half of the energy in the node (50.6%) whereas, in real case, this value is up to 65.1%.

We calculate the distribution of the packet type according to the RSSI levels on TX side. Figure 2.25 represents this distribution in RSSI windows of 5 dB from -95 dBm until -70 dBm. The percentage values of the distribution in each bar are indicated on the same figure. We observe that the [−90, −85] dBm window does not exist in this case because, in ideal case experiment, there is not any point in this window of interference.

TABLE 2.5: Results of energy and number of packets per type of packet on TX side. Real case.

Packet Type	'1'	'2'	'3'	'4'
Avg. Energy (mJ)	2.01	5.40	8.76	9.71
No. packets (%)	46.2%	10.4%	5.5%	37.9%
Total energy (%)	16.4%	9.9%	8.6%	65.1%

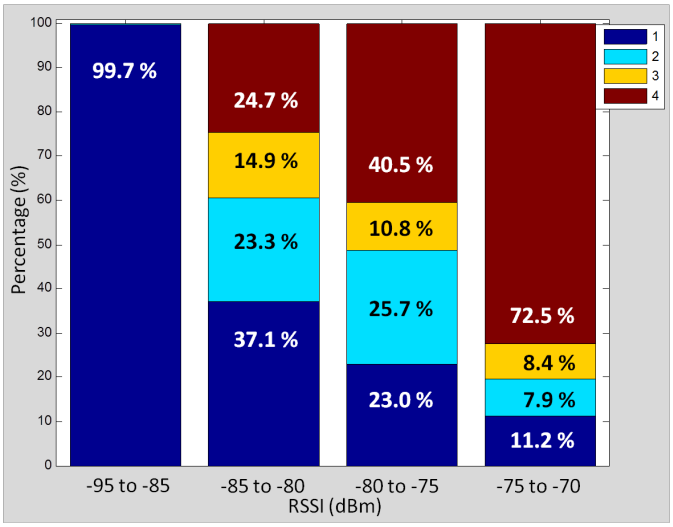


FIGURE 2.25: Packet distribution per RSSI window on TX - Ideal case with dark blue, light blue, yellow and brown slices corresponding to packet types '1', '2', '3' and '4', respectively.

In Figure 2.25, we observe how the packets of type '1' (in dark blue) located in the $[-95, -85]$ dBm window correspond to almost 100% of the packets. This is coherent because the experiment was completed in the anechoic chamber. In $[-85, -80]$ dBm window, the percentage of packets '1' has dramatically decreased but it continues to be the predominant type of packet. Also, the number of the other types of packets has increased significantly, especially types '2' and '4' with a relative frequency of 23.3% and 24.7%, respectively. In $[-80, -75]$ dBm window, packets '1' have decreased to 23% whereas type '2' and, mainly, '4' continues to increase. $[-75, -70]$dBm window corresponds to the zones where the interference becomes higher due to the activity of the three XBee modules.

In the $[-75, -70]$dBm window, the dominant packet is '4' with the 72.5%, which means that three of four packets sent with this level of interference will be lost. We also observe the low percentage of packet type '3' in all the windows. This constitutes an important issue because the probability to receive a packet successfully after three tries is low while the energy consumed by this type of packet is too high. At this point, we can conclude that having transmission limit set to three is not efficient in case of high interference, because the relative frequency of success after third transmission is very small while the relative frequency of fail is high (7.9% and 72.5%, respectively) for a window of $[-75, -70]$dBm. So, decreasing the maximal number of transmissions can help to save energy.

On the other hand, in real case, the distribution of the different types of packets is possible for 5 different interference zones from −95 dBm to −70 dBm with the step of 5 dB because there are packets situated in all the five windows of RSSI level, then, the distribution is also done in the slice [−90, −85] dBm. Figure 2.26 shows this distribution.

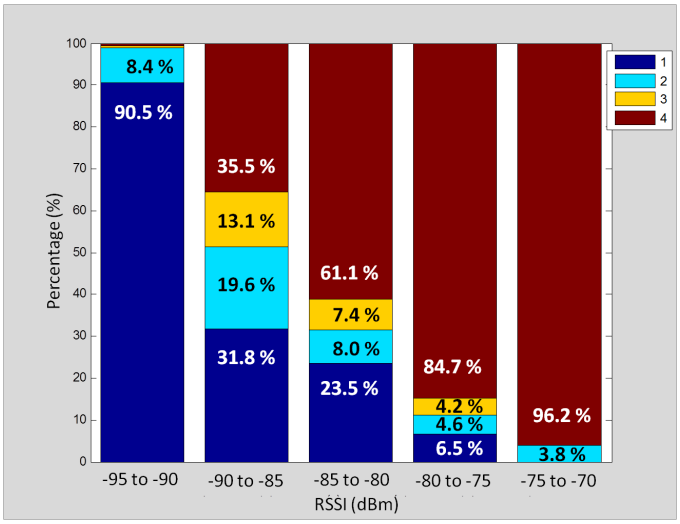


FIGURE 2.26: Packet distribution per RSSI window on TX - Real case.

We observe different behaviors of the system in each window. In the zone with less interference ([−95, −90] dBm), the percentage of successful transmissions is 99.3% (packets '1', '2' and '3') whereas the percentage of having a successful transmission with only one try (packet '1') is 90.5%. Due to the external sources of interference, this percentage is less than in the ideal case, where it reaches 99.7%. The percentage of packets '1' decreases on an exponential way until it reaches 0% in the highest interference zone. At the same time, the percentage of packets '4' increases towards the zone with more interference. The amount of packet '4' increases slightly while the decrease of packet '1' is more significant because of the presence of the types '2' and '3' in the windows between −90 dBm and −75 dBm of RSSI. Other aspect to notice is that packet type '3' represents the packet with the smallest percentage of appearance in all the windows of RSSI, as in ideal case. Also, it is important to take in consideration that the percentage of the packet type '4' in windows with the maximum of interference, [−80, −75] dBm and [−75, −70] dBm, are 84.7% and 96.2%, respectively. This means that the probability to transmit a packet successfully is very low. Then, we should avoid sending the packets when the interference reaches these levels in order to improve the energy efficiency.

2.4.2.3 Average Energy and Estimation of Node Lifetime

The average energy consumption in each slice of RSSI has been calculated from the average energy consumption per type of packet and from the packet distribution per slice of RSSI, as

shown in (2.6).

$$E_{\text{RSSI}_k} = \sum_{n=1}^{N} p_n \cdot E_{\text{avg}}(n) \tag{2.6}$$

where E_{RSSI_k} represents the average energy consumption per window k of RSSI; N is the total number of different types of packets, then, here $N = 4$; p_n corresponds to the relative frequency of the type of packet n. $E_{\text{avg}}(n)$ is the average energy consumption per type of packet.

The results obtained for the ideal case are depicted in Table 2.6. After these results, we demonstrate here that the average energy consumed by a transmitted packet varies according to the level of interference in the channel. Then, a packet transmitted with an interference level situated in the $[-75, -70]$ dBm window will consume 4.6 times more in average than a packet sent when the RSSI value does not exceed -85 dBm.

Then, we calculate the maximum number of packets (with 19-byte payload) transmitted according to the interference level until the total discharge of the battery in the node. For that, we have chosen the real 850-mAh TCL PL-383562 polymer Li-ion battery presented in Section 2.2.1. We consider that below 2.8 V, the circuit does not work. This point arrives near the 5% of SOC, then, the maximum amount of energy in battery used to supply circuit corresponds to about 95% of the total energy stored in battery. Table 2.7 shows the average time spent by TX node per type of packet. This average time is obtained from experimental results. We observe that the time for the packets of type '1' and '2' is less than 1 second. This is because the packets stored in the buffer are sent as quick as possible. These packets are temporary stored in buffer after transmission failures, then, once the transmission is possible, TX node tries to empty the buffer.

Based on the energy stored in battery, the maximum number of packets transmitted per slice of RSSI and the average time per packet, it is possible to estimate the lifetime of the node per level of interference. These results are shown in Figure 2.27. In this Figure, the lifetime of the node decreases dramatically from 46.8 days in the $[-95, -85]$ dBm window to 20.2 days in

TABLE 2.6: Average energy and maximum number of transmitted packets per RSSI window - Ideal case.

RSSI (dBm)	-95 to -85	-85 to -80	-80 to -75	-75 to -70
Avg. energy (mJ)	1.84	5.71	6.74	8.47
Num. of packets ($\times 10^6$)	5.23	1.68	1.42	1.13

TABLE 2.7: Average time per type of packet - Ideal case.

Type of packet	'1'	'2'	'3'	'4'
Time (ms)	774.0	908.8	1428.6	1327.2

[−85, −80] dBm window, then, more than by half. The lifetime continues to decrease up to 16.2 days if the interference level in the channel stays in [−75, −70]dBm window.

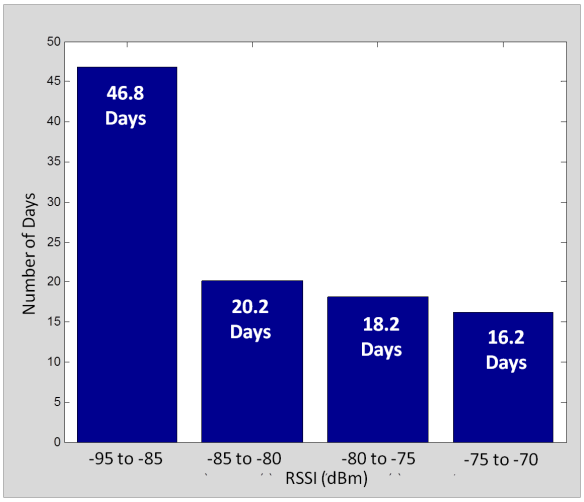


FIGURE 2.27: Lifetime node according to the interference level - Ideal case.

On the other hand, in real case, we have the Table 2.8 which includes the average energy consumption of the TX side radio module for each interference windows.

These values have been obtained from the distribution of the different types of packets per slice of RSSI and the average energy corresponding to each type of packet for the real case. We observe that the average energy consumed by a transmitted packet grows with the interference level, mainly due to the dropped packets (packet type '4'). We can notice that the energy consumption of the radio module in the highest interference zone is about four times bigger than the consumption in the lowest interference zone. This becomes an important indicator of the impact of the interference on the energy consumption and on the performance of the system.

We calculate the number of packets that TX can send with a 850-mAh battery supplying the

TABLE 2.8: Average energy and maximum number of transmitted packets per RSSI window - Real case.

RSSI (dBm)	-95 to -90	-90 to -85	-85 to -80	-80 to -75	-75 to -70
Avg. energy (mJ)	2.37	6.29	7.48	8.97	9.54
Num. of packets ($\times 10^6$)	4.04	1.52	1.28	1.07	1.01

TABLE 2.9: Average time per type of packet - Real case.

Type of packet	'1'	'2'	'3'	'4'
Time (ms)	704.6	971.6	1371.7	1320.5

node, as in ideal case. The results of the number of packets per slice of RSSI are also presented in Table 2.8. We observe that four times more of packets could be sent in conditions of low interference in comparison with the highest interference conditions. In Table 2.9, the average time of each type of packet is shown. This average time is calculated directly from the experiment measurements, as in the ideal case.

Finally, the lifetime of the node according to the interference level is represented in Figure 2.28. The maximum possible lifetime is in this case 34.4 days whereas the operation of TX node only in the zone of the highest interference level can lead to only 15.2 days, which is less than the half. Also, if we compare both, ideal and real cases, lifetime of the node is shorter in all the RSSI slices of the real case due to the interference caused by the external sources.

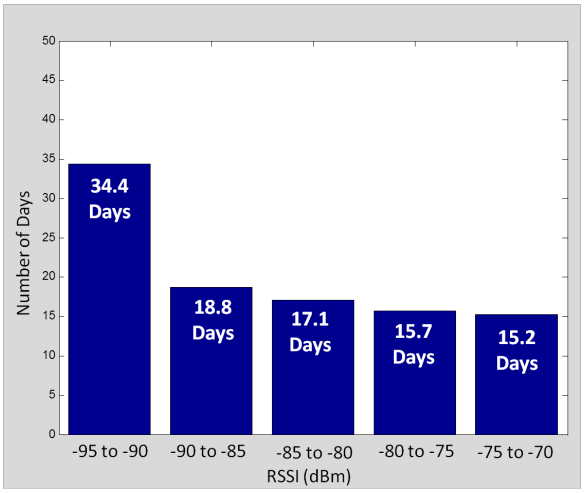


FIGURE 2.28: Lifetime node according to the interference level. Real case.

Next section presents the results obtained for similar experiments with other interfering sources such as Wi-Fi technology.

2.4.3 Wi-Fi Interference

The tests with Wi-Fi interference have been carried out in a laboratory environment. For that, we use the same experimental setup as in the previous Section, with the same devices shown in Figure 2.21. The TX node sends a 19-byte payload packet every second and waits for the ACK frame coming from RX node. In this case, the interference source is a laptop placed at 30 cm from the TX node. This computer uses its Wi-Fi module to communicates with the Wi-Fi access point 10 meters away. In order to send the IEEE 802.11 packets in a controlled manner, we use the D-ITG [152] (Distributed Internet Traffic Generator) tool where it is possible to choose the length of packet and the number of packets per second to be sent.

The IEEE 802.11 frame format [153] is composed of a fixed header of 34 bytes and maximum payload of 2312 bytes. In this test, we choose a payload of 250 bytes with a data rate of 54

Mbps, however, the data rate in header is different than in payload, 6 Mbps. Then, the on-air time is calculated through the expression in (2.7). We have a duration equal to $t_{\text{paq1}} = 82.37\mu$s that a Wi-Fi frame stays on the air. The on-air time of an IEEE 802.15.4 frame is also calculated in (2.8). In this case, the total length of the frame is 40 bytes and the data rate is $R = 250$ kbps. Then, this on-air time is $t_{\text{paq2}} = 1.28$ ms, *i.e.* $\frac{t_{\text{paq2}}}{t_{\text{paq1}}} = 15.5$ times greater than the on-air time for IEEE 802.11. With this information, we can calculate that an IEEE 802.11 frame occupies $\frac{40 \text{ bytes}}{15.5} = 2.57$ bytes of IEEE 802.15.4 frame or, thus, 2.57×8 bits $= 20.59$ bits. This is enough to corrupt an IEEE 802.15.4 packet.

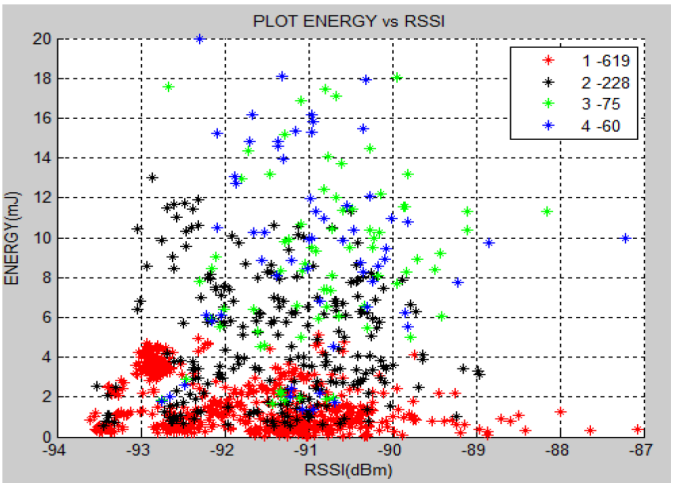
$$t_{\text{paq1}} = \frac{(L_{\text{header}} \times 8 \text{ bits})}{R_{\text{header}}} + \frac{L_{\text{payload}}) \times 8 \text{ bits}}{R} = \frac{34 \times 8}{6 \cdot 10^6} + \frac{250 \times 8}{54 \cdot 10^6} = 82.37\mu\text{s} \tag{2.7}$$

$$t_{\text{paq2}} = \frac{(L_{\text{header}} + L_{\text{payload}}) \times 8 \text{ bits}}{R} = \frac{40 \times 8}{250 \cdot 10^3} = 1.28\text{ms} \tag{2.8}$$

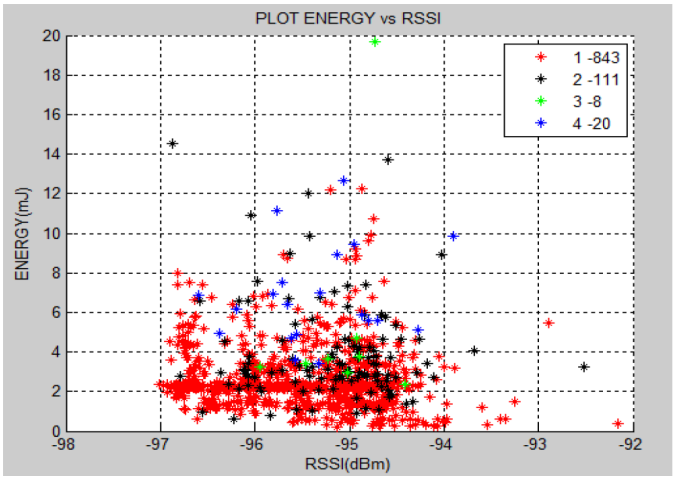
In this experiment, the channel starts without interference from the laptop during 2 minutes 30 seconds. Then, we decrease the interval of sending the IEEE 802.11 packets every 2'30" in 100, 300, 500, 700, 900 packets per second. The analysis of the packets sent by the TX is represented in Figure 2.29a while the analysis in RX is depicted in Figure 2.29b. We observe that, in both cases, the energy consumed by the packets according to their packet type is coherent, since the packets '1' (in red) remains the less energy-hungry. Then, the occurrence of packets '2', '3' and '4' decreases whereas the average energy consumed by these packet types increases respectively. In abscissa axis the situation is different, the range between the lowest and the highest RSSI values is small (around 7 dB), and the measurements are not coherent. This is due to the RSSI measurements of CC2420 transceiver included in TelosB. These measurements are carried out every 128 μs, whereas the IEEE 802.11 packet on-air time is shorter, $t_{\text{paq1}} = 82.37\mu$s. Then, CC2420 radio module and, generally, the IEEE 802.15.4 transceivers are not able to measure if the channel is occupied by other electromagnetic sources, as IEEE 802.11 devices or the microwave ovens. This is a serious problem in WSN applications where many packets may be corrupted because of these interference sources and the WSN nodes cannot detect these perturbations in the channel.

The average energy, the number of packets and the total energy values par packet type in this experiment are presented in Table 2.10. We observe that the average energy in this case is similar to the average energy per packet type obtained in precedent tests with IEEE 802.15.4 interference.

We conclude with these results that an application layer packet transmitted one, two or three times will consume in the same way for all the types of interference. The value that changes is the number of packets for each packet type and, therefore, the total energy consumed.



(A) TX side



(B) RX side

FIGURE 2.29: Energy consumption vs RSSI per packet type with Wi-Fi interference.

TABLE 2.10: Results of energy, number of packets and total energy per type of packet on TX side. Ideal case.

Packet Type	'1'	'2'	'3'	'4'
Avg. Energy (mJ)	1.60	4.76	8.86	9.65
No. packets (%)	63.0%	23.2%	7.6%	6.1%
Total energy (%)	29.9%	32.6%	20.0%	17.4%

The study of the distribution of packet types and the node lifetime per interference window is not accurate in this case because of the low precision in RSSI measurements. This is why, this node lifetime according to the interference level caused by a Wi-Fi device will be study more precisely in a future work. These perspectives are explained in Section 2.5.2.

2.5 Perspectives

2.5.1 Synergie Platform - NI Test Bench

As shown in Figure 2.2, the architecture of the hardware part of Synergie platform remains the same for any power meter used. Previously, in Section 2.2.1, we presented an energy measurements platform with five ZXCT 1086 amplifiers and an ATmega328P microcontroller with ADC. However, other power meters can be used if we follow the schema of Figure 2.2. Another more sophisticated microcontrollers and ADCs or a solution based on a field-programmable gate array (FPGA) were planned to be used in order to recover the energy measurements from a WSN nodes. Moreover, this task can be accomplished by a test bench created by National Instruments.

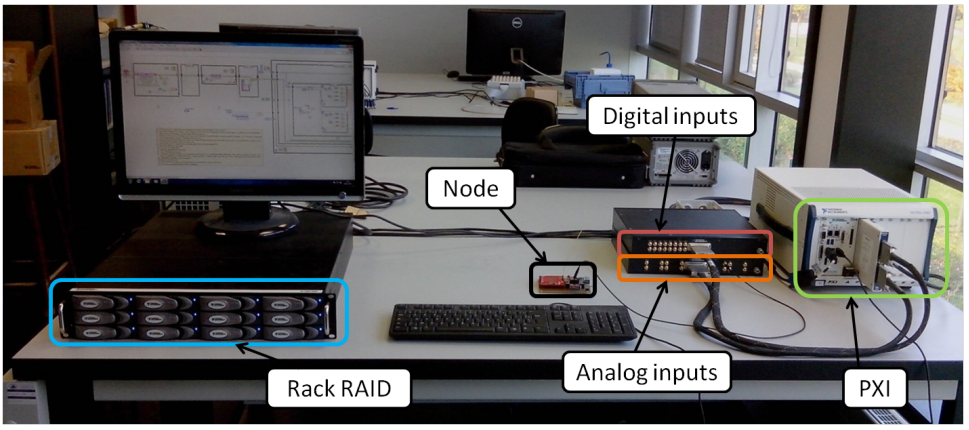


FIGURE 2.30: Energy measurements NI test bench.

This test bench is composed of a NI 5751 digitizer adapter module [154] that works with a NI FlexRIO FPGA module [155]. This NI 5751 module contains 16 analog input channels with 14 bits of resolution and a data acquisition rate of 50 MS/s. Moreover, this device has 8 digital input channels. Then, the NI 5751 module recovers the energy measurements and the digital indicators from the node under evaluation. This information is taken by the FPGA module and stored in a NI HDD-8265 RAID [156] with 6 TB of capacity. All these modules are managed by a program in LabVIEW [157] included in a computer. This computer, the NI 5751 and the NI FlexRIO FPGA modules are included in a NI PXIe-1082 chassis [158], as shown in Figure 2.30. Then, these data stored in the RAID may be analyzed in mathematical software such as MATLAB.

The first experiments using this test bench has been realized with a LoRa node (explained in Section 2.2.4). The energy measurements from the node are correctly recovered by the system and shown in LabVIEW, but some problems occur in our case when these data are exported to the mathematical software. This is why, the next steps of this study with the NI test bench are considered as future work.

2.5.2 Energy Consumption vs. Interference with NI USRP

The future work of the interference study consists to substitute the TelosB as channel sniffer by a high-performance device as a NI USRP (National Instruments Universal Software Radio Peripheral). We have already this Software Defined Radio (SDR) device, precisely the NI USRP-2922 [159] with a frequency band from 400 MHz to 4.4 GHz, to realize some channel measurements using LabVIEW [157] software.

Figure 2.31 shows an example using the USRP as channel sniffer, where the amplitude is in ordinate axis and the time in abscissa axis. The channel analyzed is the IEEE 802.15.4 channel 12 with 2 MHz of bandwidth centered in 2.410 GHz. Several signals of three types of wireless technologies that coexist in the same frequency band are differentiated. The USRP is placed close to a XBee node which sends a 100-byte payload packet every 100 ms to the coordinator node. A Wi-Fi access point is situated at a distance of 10 meters from the XBee node and, moreover, a smartphone with the Bluetooth module activated is placed to 2 meters. Then, the smartphone communicates via Bluetooth with another smartphone in the same room. The Wi-Fi access point is activated and continuously sends the control packets, but any Wi-Fi device in the same room communicates with this access point.

In Figure 2.31, we have an IEEE 802.15.4 packet with a considerable on-air time due to the slow data rate of this technology. A small Wi-Fi control packet also appears just after the IEEE 802.15.4. Then, a Bluetooth packet arrives at the same time as the IEEE 802.15.4 and, even if this Bluetooth packet is much smaller, it achieves to corrupt the IEEE 802.15.4 packet. Thus, a retransmission of the IEEE 802.15.4 packet is done and the time in active mode of the XBee node increases. This shows that an impulse noise, which cannot be detected by the IEEE 802.15.4 transceivers, may corrupt a packet, increases the energy consumption and decreases the lifetime of the node.

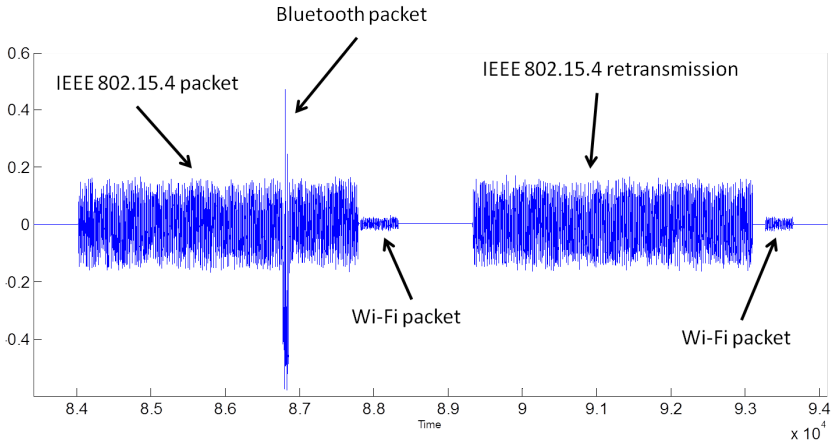


FIGURE 2.31: Coexistence of technologies in 2.4-Ghz band. Measurements by NI USRP.

Chapter 3

Energy Consumption Model

3.1 Introduction

Energy measurements have been carried out by an experimental platform. Our main objective consists in optimizing the hardware (the use of the electronic components in the circuit) and the software (the embedded code in the microcontroller of the node) to consume as less energy as possible. For that, we have to evaluate in a realistic way the energy consumed by the nodes of a WSN. We can then calculate the energy consumption and estimate the battery lifetime thanks to an energy model. Finally, we reach some conclusions to modify the software and/or the hardware in the node. Figure 3.1 depicts this process. Between the energy measurements obtained by the energy platform and the estimation of lifetime, we need to develop a mathematical algorithm which describes an energy model. This process is carried out as many times as necessary to reach the optimization.

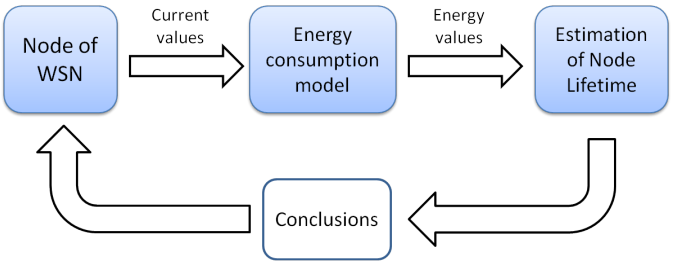


FIGURE 3.1: Process of node lifetime estimation.

In this chapter, firstly, we present a theoretical energy consumption model based on the state of the art of the energy models. Then, we introduce a new algorithm which automatically analyzes the data in order to obtain the parameters of the model, based on a Markov chain. We show how the automatic energy model predicts the lifetime of the node and how this algorithm allows to optimize the hardware and the software of the nodes in a easy way.

3.2 Theoretical Energy Consumption Model

3.2.1 Deterministic Model

In this Section, we will present a deterministic approach to model the energy consumption, we will then discuss related works and show the limitations of such an approach.

The total energy consumed in a node is equivalent to the sum of energy consumed by every electronic component, as expressed in (3.1).

$$E_{\text{Total}} = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N} P_i \cdot T_i \tag{3.1}$$

where i represents each electronic component in the circuit of the node and N is the total number of components in this circuit. E corresponds to the energy consumption, P_i to the power and T_i to the functioning duration.

The energy consumption in each electronic component is composed basically of two different modes: the *state* and the *transition*. The current in state generally corresponds to a stable level of current. This level of current will be higher or lower, depending on the action executed by the component. When this component modifies its action, the level of current can change and, then, a transition between the current levels occurs. This transition may be important in energy consumed due to a high (in current) or a long (in time) change of level. Then, the difference in the states and the transitions between states must be included in the energy model. The energy consumed by an electronic component in the circuit is presented in (3.2) whereas the expressions of energy consumed in a state and in a transition are presented in (3.3) and in (3.4), respectively. We can notice that, generally, in a theoretical model, the transition between states is represented by a linear progression but, in a real situation this progression can differ from such a linear function.

$$E_i = \sum_{\text{state}=1}^{M} E_{\text{state}} + \sum_{\text{trans}=1}^{M-1} E_{\text{trans}} \tag{3.2}$$

where M corresponds to the number of different states in an experiment whereas E_{state} and E_{trans} are represented in

$$E_{\text{state}} = P_{\text{state}} \cdot T_{\text{state}} \tag{3.3}$$

$$E_{\text{trans}} = \frac{(P_{\text{state}} - P_{\text{state+1}})T_{\text{trans}}}{2} + P_{\text{state+1}}\,T_{\text{trans}} = \frac{(P_{\text{state}} + P_{\text{state+1}})T_{\text{trans}}}{2} \tag{3.4}$$

The power consumed in the transition state P_{trans} is calculated from the equation of the middle point. It means that the midpoint between two consecutive states are identified and multiplied by the transition duration T_{trans}.

Besides, we can find two types of energy consumption states: the *fixed states* and the *data-rate-depending states*. For *fixed states*, the time (T_{state}) does not depend on other parameters that the activation time. This type corresponds to states as sleep, active or idle. However, some electronic components use some different rates to execute an action. For instance, an external flash memory uses a certain data rate to read or write data; an RF transceiver can send and receive packets with a given data rate whereas the serial communication with a microcontroller can be established with a different data rate. Then, the time spent depends on the length of the data flow and on the data rate of the device. This type of states is called *data-rate-depending state* (DR) and its energy consumption is defined in (3.5).

$$E_{\text{state}_{\text{DR}}} = P_{\text{state}_{\text{DR}}} \cdot T_{\text{state}_{\text{DR}}} = P_{\text{state}_{\text{DR}}} \sum_{j=1}^{N_p} \frac{L_j}{R_k} \tag{3.5}$$

where $P_{\text{state}_{\text{DR}}}$ is the power needed and $T_{\text{state}_{\text{DR}}}$ is the time spent by the component in this state. $T_{\text{state}_{\text{DR}}}$ can be expressed as the division between the length L_j of the j^{th} packet and the data rate R_k of the k^{th} action. N_p represents the number of packets to be treated.

The objective of the energy model is to reproduce the different states of the electronic components of the node as the finite state machine (FSM) shown in Figure 3.2. In fact, each electronic component works in different operating modes with different current levels.

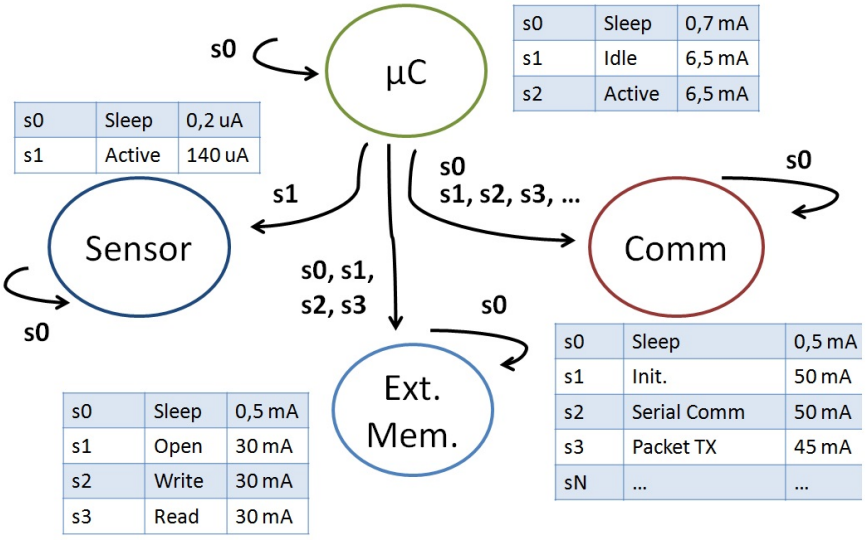


FIGURE 3.2: FSM of the operating modes in the electronic components of a node.

In general terms, the microcontroller contains three operating modes: sleep, idle and active. The sensor is composed of different states which depend on the type of sensor, but the sleep and the active operating modes constitute the basic states for this type of hardware component. In an external memory, we find always the same basic operating modes: sleep, write, read and other states as open a file and close a file. The communication or radio module is generally the most complex device because it is composed of some operating modes such as sleep, initialization, serial communication with microcontroller, packaging of the information, calculation of cyclic redundancy check (CRC), clear channel assessment and channel energy detection (CCA/ED), packet transmission and packet reception.

The timetable of Figure 3.3 represents an example of the time spent in each state for the state machine presented in Figure 3.2. Both, currents and time of each state play a key role in the energy consumption and both must be integrated in the model. The FSM and the timetable with the states of the electronic components are used in most of the energy consumption models found in research studies, as we will see in Section 3.2.3.

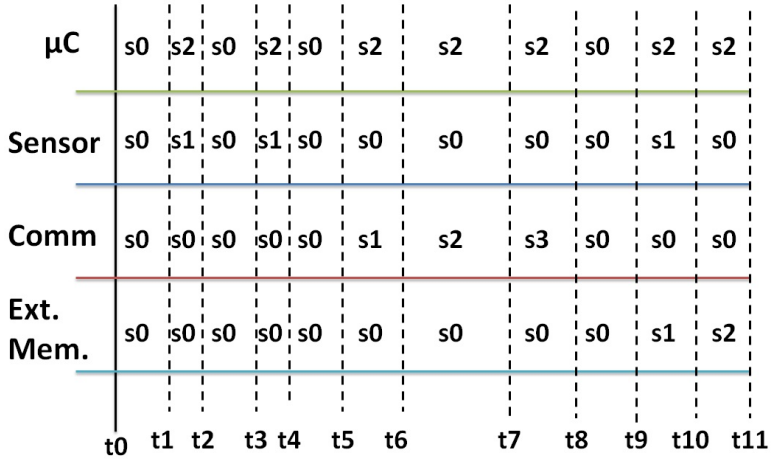


FIGURE 3.3: Timetable of states in each device.

Furthermore, each operating mode may consist in several variations, for instance the sleep mode in a microcontroller has several low-power modes (LPM) with different current levels. An example of a microcontroller with different LPMs is a MSP430F1611 microcontroller [138] from Texas Instruments, where the sleep mode is composed of LPM0 with 75 μA, LPM1 with also 75 μA, LPM2 with 17 μA, LPM3 with 2.6 μA and LPM4 with 0.2 μA. In this microcontroller, the LPM4 operating mode has a lower current level than LPM0 but, on the other hand, the microcontroller takes more time to wake up in LPM4 than in LPM0. Then, depending on the type of application, the LPM0 can be more energy efficient than the LPM4. Another example of variations in an operating mode is the transmission power of a radio module. For instance, in CC2420 transceiver [110] from Texas Instruments, the output transmission power may take different values, as shown in Table 3.1 and the current requested by this device varies with the output transmission power.

TABLE 3.1: TX mode in CC2420 and the respective currents.

TX Mode	Current
TX (+0 dBm)	17.4 mA
TX (-1 dBm)	16.5 mA
TX (-3 dBm)	15.2 mA
TX (-5 dBm)	13.9 mA
TX (-7 dBm)	12.5 mA
TX (-10 dBm)	11.2 mA
TX (-15 dBm)	9.9 mA
TX (-25 dBm)	8.5 mA

In Figure 3.4, we present the FSM of the possible operating modes in a microcontroller as, for instance, the ATmega328P microcontroller. This microcontroller is integrated in a node, such as the node evaluated in Chapter 2, with other components as radio module and sensor. This FSM has been created from the behavior of the microcontroller observed in the measurements and from the information contained in the datasheet of the component. As seen in Figure 3.4, at the beginning, the microcontroller stays in sleep mode, the lowest power mode. After some time, it wakes up, this state is considered a transition between the precedent sleep state and another state. The duration of the wake-up state depends on the chosen sleep mode. Then, the embedded software can reach different states as idle, the use of a serial communication (UART, I2C, SPI) to take a value from the sensor or to wake the radio module up. The idle state corresponds to the central state with which the embedded software can reach most of the other states. After taking a value from the sensor, generally, this value is

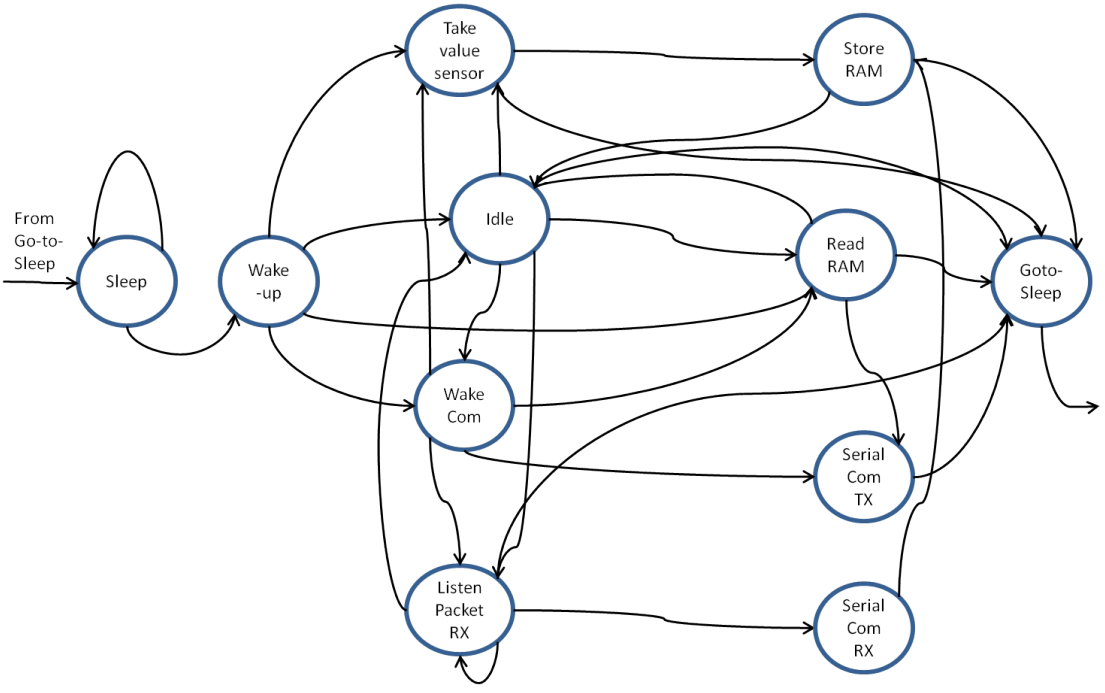


FIGURE 3.4: FSM in microcontroller.

stored in the RAM of the microcontroller. On the other hand, in the embedded software under evaluation, once the microcontroller wakes the RF module up, the microcontroller reads the values stored in RAM and offers these values to the transceiver via a serial communication. Another action that the microcontroller can carry out is the reception of a packet. For that, it wakes the transceiver up and asks this RF module to listen to the wireless channel. The packets received by the RF module are transmitted to the microcontroller via a serial communication and, then, this information is stored in RAM. Almost any state in the process can reach the sleep mode through an intermediate transition called *Goto-sleep* in the diagram of Figure 3.4.

The actions of the RF module can also be expressed as a FSM, as shown in Figure 3.5. This FSM is even more complex than the FSM in microcontroller because of the greater number of possible states. The presented example also represents the case of Chapter 2 with a XBee module Series 1 as transceiver. The states are grouped in four fields and, moreover, a fewer generic states. The first field corresponds to the sleep mode where three different sleep states can be configured. These three sleep states are called in XBee transceiver datasheet [127] as Sleep 1, Sleep 2 and Sleep 4. The Sleep 1 is the Hibernate mode where the current level is the lowest ($< 10\,\mu$A) but the time to wake up is quite long, about 13.2 ms. The Sleep 2 corresponds to the Doze mode with a higher current than in Hibernate mode ($< 50\,\mu$A) but with a lower

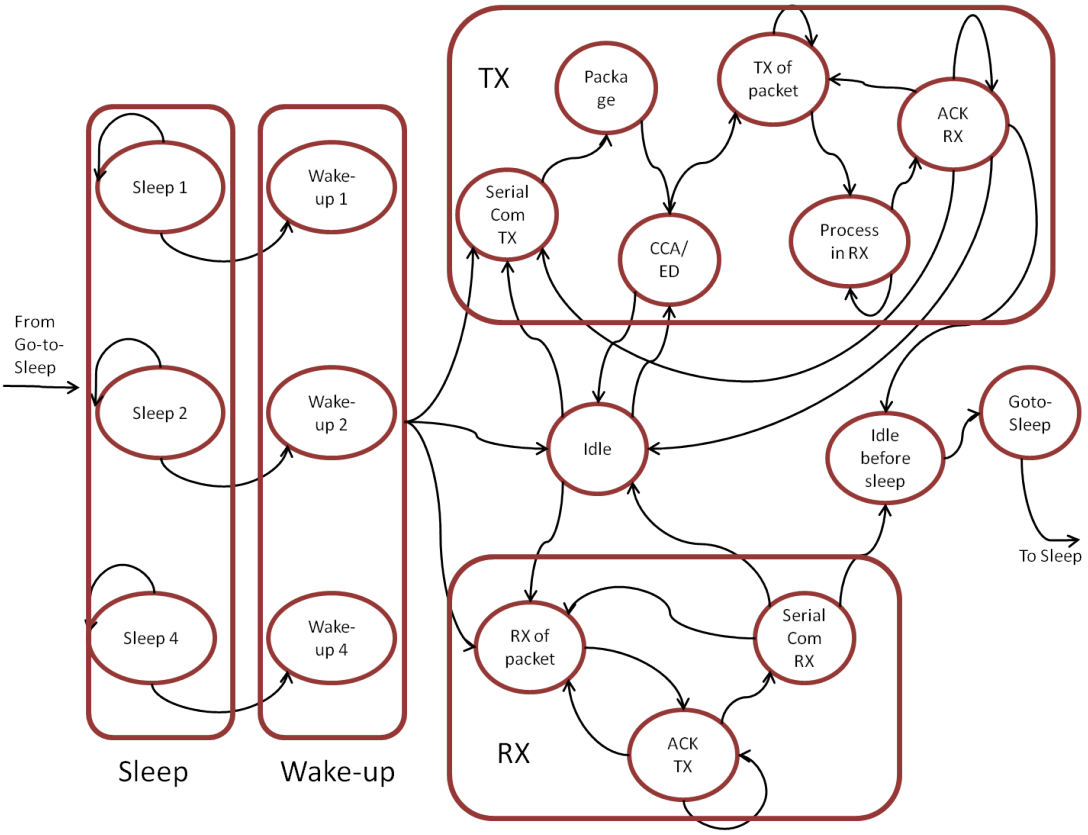


FIGURE 3.5: FSM in radio module.

wake-up time (about 2 ms). Finally, Sleep 4 corresponds to a cyclic period time determined by the RF module. A time to wake up is configured in the transceiver and this device wakes up automatically to listen to the channel while the microcontroller continues to sleep. If the transceiver identified a packet in the channel, this device wakes the microcontroller up to receive and treat the information. This is a different way to work in the node that may be interesting depending on the type of application. As explained before, each sleep mode has a different way to wake up, as presented in Wake-up states 1, 2 and 4. Once the transceiver is awake, three possible actions can happen: idle, transmission (TX) or reception (RX). Also, from idle state, the process can reach the TX or RX sets of states:

- The *TX process* begins with the transmission of the information from the microcontroller to the transceiver via a serial communication. Once the information arrives to the RF module, the packaging process is done where a header is added at the beginning of the packet and the cyclic redundancy (CRC) field is added at the end after its calculation from the bytes in the information (payload) and in the header. After this packaging process, the packet is ready for transmission. Then, an energy detection (ED) of the wireless channel and a clear channel assessment are carried out by the transceiver to evaluate if the channel is not occupied in the instant of transmitting the packet. The data packet is sent and processed by the receiver. At this moment, the transmitter listens to the channel waiting for the acknowledgment (ACK) frame which confirms the correct reception. After these states, the transmission action finishes and the transceiver returns to the sleep mode or it remains active in the idle state.
- In *RX modes*, the device listens to the wireless channel in order to receive a packet. If the received packet is correct, the receiver sends an ACK frame to the node which transmitted the packet. If the ACK frame is well received by the other node, the RX mode ends with the communication of the data to the microcontroller in the node. Then, the process in the transceiver can stay in idle state or go to sleep mode.

Besides, a node may contain different types of sensors. The main states of a sensor are represented by the FSM in Figure 3.6. Only two states, sleep and idle or active, and two transitions between the states: wake-up from sleep to active state and goto-sleep from active to sleep mode. Depending on the sort of sensor, the current level in sleep and in active states changes as well as the time spent in the transitions.

With the previous FSMs in microcontroller, transceiver and sensor, we show the complexity of a generic energy model where each electronic component can work in different states. The FSM in microcontroller contains 11 states, the FSM in transceiver is composed of 18 states whereas 4 states are in the FSM of the sensor. This means that the timetable of the global states in the node (as seen in example of Figure 3.3) can contain up to $11 \times 18 \times 4 = 792$

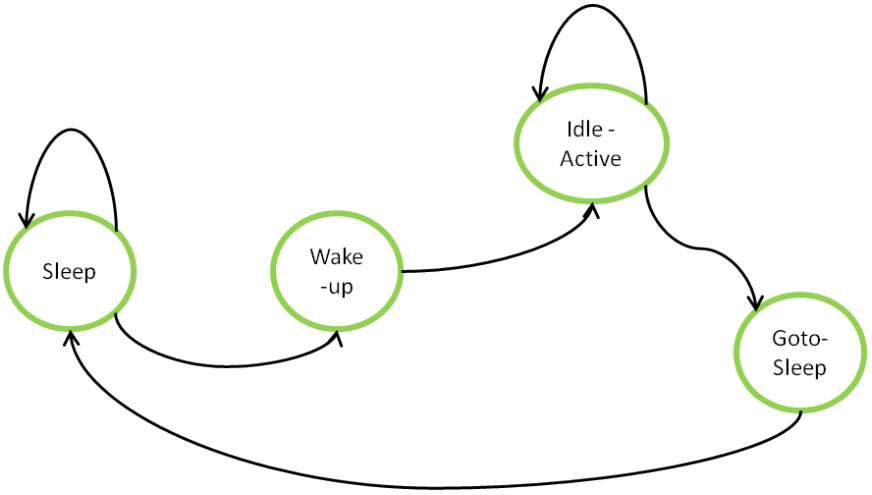


FIGURE 3.6: FSM in sensor.

different global states. Moreover, we explained that some states are composed of variations as the sleep mode in microcontroller, the sleep mode in transceiver or the output transmitting power in the transceiver. Then, the deterministic modeling of the energy consumption becomes a really hard task.

3.2.2 Lifetime Estimation

We now present an example of the estimation of the battery lifetime in a node based on such a theoretical energy consumption model. These tests have been carried out to complete the results in energy aspects of Amal Abbadi's thesis [160], in CSAM research team [3]. Amal's thesis consisted in the study of structural health monitoring in buildings, bridges or others civil infrastructures through wireless technologies as WSN. The constraint of energy consumption and the lifetime of the node is a major issue because the nodes are installed in the concrete of the solid structures when these structures are built. These nodes must be operational during the building lifespan, i.e. tens of years. Furthermore, there are two critical phases to measure the characteristics of the concrete (temperature, humidity, deformation): 1) the first six months, when the structure has just been built and the concrete is still wet; and 2) after 40 years, when the infrastructure becomes old and some problems may appear. Unfortunately, once the nodes are installed in the infrastructure, they cannot be removed. This is why, the energy efficiency in the nodes is crucial.

A simple generic energy consumption model based on the expressions previously presented (3.2)-(3.5) has been used to calculate the lifetime of the nodes. The nodes integrate different devices as a TI MSP430F2370 [161] microcontroller, a sub 1-GHz Nordic Semiconductor nRF905 [162] RF module and temperature, humidity and deformation sensors. The current values obtained from the datasheets for the different operational modes in these devices

are represented in Tables 3.2, 3.3 and 3.4, respectively. The sensor used in this case is the temperature-humidity sensor.

TABLE 3.2: Currents in microcontroller

Mode	I (mA)
Power-off	0.00085
Stand-by	0.023
Active 1	0.39
Active 2	6

TABLE 3.3: Currents in RF module

Mode	I (mA)
Power-off	0.0025
TX (-10 dBm)	9
RX	12.8

TABLE 3.4: Currents in sensor

Mode	I (mA)
Sleep	0.001
Active	2

The embedded software consists in taking a value from the sensor after t_{sleep} seconds. The time spent to take a value is 20 ms and each value is equivalent to 2 bytes. Then, the microcontroller and the sensor go back to sleep mode. This process is repeated for $N_{\text{data}} = 10$ times, after what, the RF module is activated. The serial communication between the microcontroller and the transceiver is done with a data rate $R_{\text{ser}} = 57600$ bps while the data rate of the RF communication between two nodes is $R_{\text{RF}} = 100000$ bps. We add the header to the payload with a destination address of 2 bytes and a source address of 2 bytes, in addition to a CRC field at the end of the packet composed of 2 bytes. We complete the time of active mode in RF module with a preambule of 1 ms before sending the packet and an ACK frame comprising the same header as for the main packet and the CRC field, thus, the length of the ACK frame is 6 bytes. The battery used by the evaluation of the lifetime corresponds to two AA batteries with 2500 mAh of capacity. We consider that with a capacity of less than 5%, the voltage level offered by the battery falls down under the level necessary for the correct functioning of the circuit.

Now that the hardware and the embedded software in the node for this test have been presented, we can calculate the lifetime for several situations. In the structural health monitoring applications, the duration that the node remains in sleep mode between two measurement times from the sensor can be quite long because the characteristics of the concrete do not change much (this is particularly true after the first six months of the construction). Then, the first test consists in comparing the lifetime of the node for different periods of sleep mode. We choose $t_{\text{sleep}} = [1, 2, 3, 4, 5, 10, 20, 30, 60, 120, 180, 240, 300, 600, 1200, 1800, 3600]$. The current values chosen for the sleep and active modes in microcontroller are the lowest values in Table 3.2, obtained from the datasheet, i.e. 0.00085 mA for sleep mode and 0.39 mA for active mode. In this case, the node lifetime in relation of the sleep time is represented in Figure 3.7 and the values are shown in Table 3.5. We can observe that in best cases, when the sleep period reaches values higher than 3 minutes (180 seconds), the difference between two consecutives values is no longer significant. In this zone, the function reaches a saturation state, when most of energy is consumed in sleep mode.

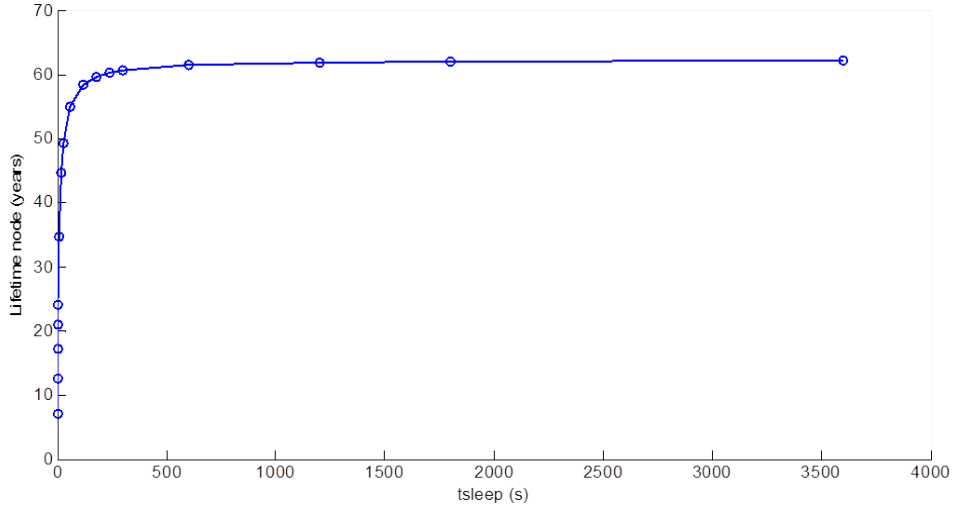


FIGURE 3.7: Lifetime of node vs. t_{sleep}

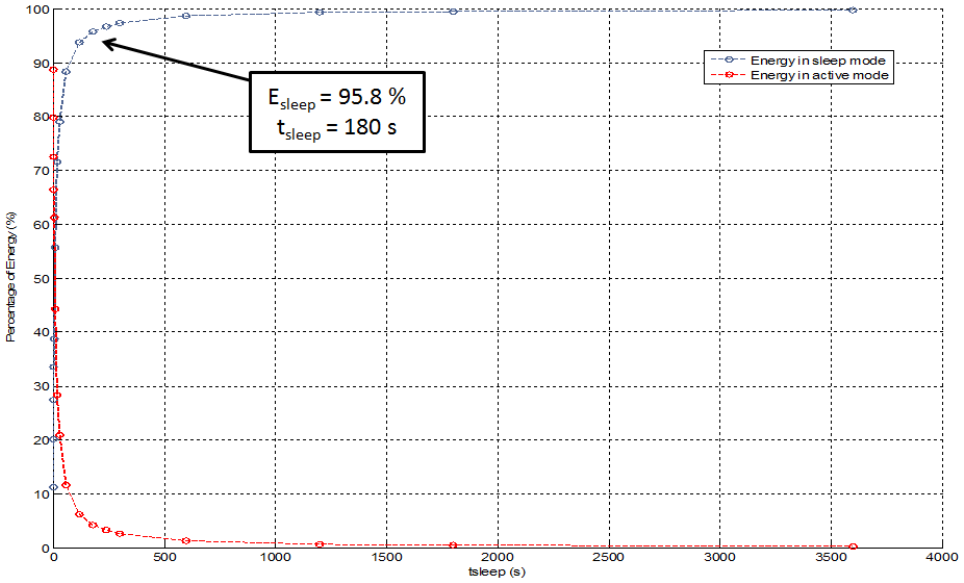
TABLE 3.5: Lifetime values vs. t_{sleep}

$t_{\text{sleep}}(s)$	1	2	3	4	5	10	20	30	60	120	180
Lifetime (years)	7.1	12.6	17.2	20.9	24.1	34.8	44.6	49.3	55.1	58.47	59.7

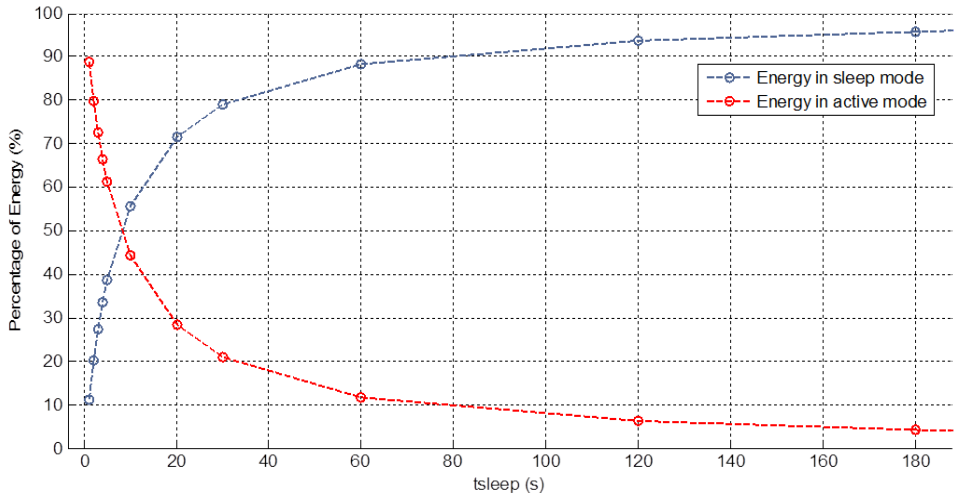
$t_{\text{sleep}}(s)$	240	300	600	1200	1800	3600
Lifetime (years)	60.3	60.7	61.5	61.9	62.1	62.2

Figure 3.8a depicts the part of energy spent in sleep mode and in active mode depending on the t_{sleep} value. We observe that when $t_{\text{sleep}} = 180$ s, the weight of the sleep mode with respect to the active mode is higher than 95%. Then, after this point, even if t_{sleep} increases in order to extend the node lifetime, the situation does not improve very much because the proportion of energy in sleep mode approaches 100%. In conclusion, extending in this example the sleep mode duration over 180 s is not interesting, because the throughput decreases whereas the lifetime remains more or less the same, around 60 years. Figure 3.8b represents a zoom of Figure 3.8a from $t_{\text{sleep}} = 1$ s to $t_{\text{sleep}} = 180$ s. We observe the progression of the percentage of energy used by the sleep mode according to the increase of the period of this sleep mode.

In a second test, we carry out an estimation of the node lifetime for the same node and the same embedded software but with different current values in sleep and in active mode, as shown in Table 3.2. Figure 3.9 presents these values of lifetime according to t_{sleep}, where *Sleep low* corresponds to the current value 0.00085 mA, *Sleep high* is 0.023 mA, *Active low* is 0.39 mA and *Active high* is 6 mA. In Figure 3.9a, we observe that the lifetime depends only on the current level of the sleep mode when the time spent in this mode is significantly longer than the time in active mode. For this reason, the graph with *sleep low* level converge to a lifetime value around 62 years while the lifetime in both graphs with *sleep high* current level is slightly above 10 years. Figure 3.9b depicts the same charts but with values $t_{\text{sleep}} \leq 60$ s. In these graphs, we notice that for values $t_{\text{sleep}} \leq 4$, the use of *Sleep high*, *Active low* current



(A) Percentages of energy.



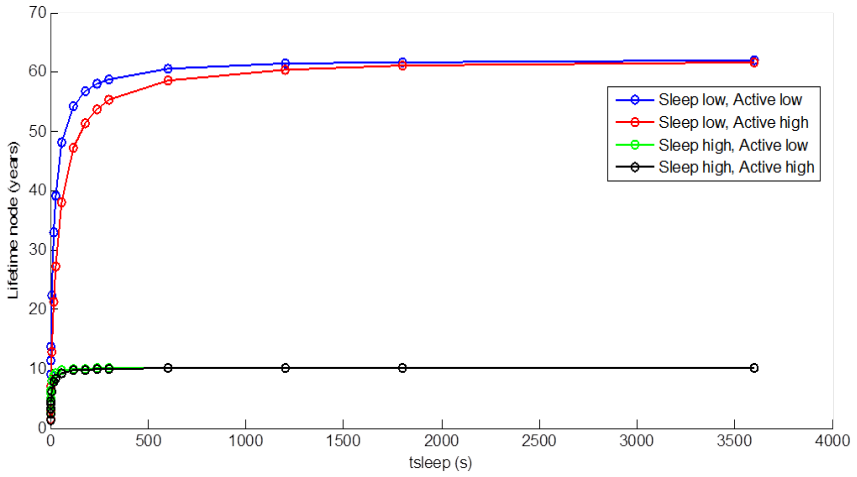
(B) Zoom in Figure 3.8a.

FIGURE 3.8: Percentage of the sleep and active modes in total energy consumption.

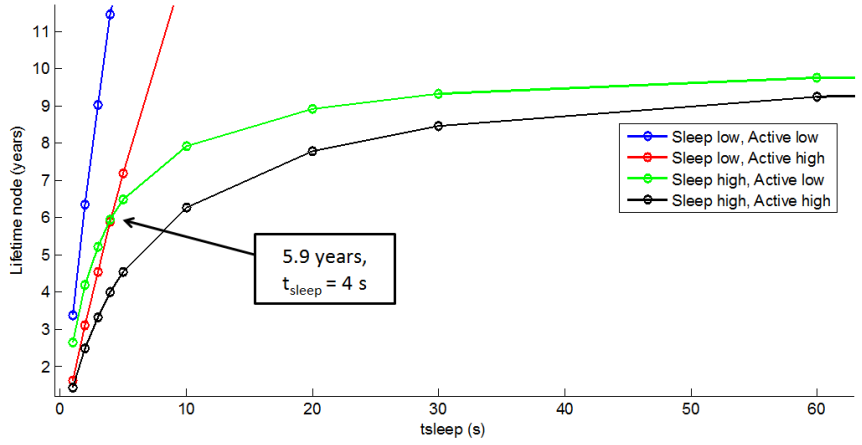
levels is more energy efficient than with *Sleep low, Active high*. This means that the active mode may be important in the energy efficiency of the node but for periods of sleep mode less or equals to 4 seconds.

The next step would be to use measurement platform of Synergie to adjust Tables 3.2, 3.3 and 3.4, but the number of states make the task quite difficult and dependent on the hardware and the software implementation. We want to alleviate this difficulty by automatically analyzing the measured energy values.

Before that, we present in Section 3.2.3 the studies found in the research publications concerning the energy consumption model.



(A) Lifetime in different cases.



(B) Zoom in Figure 3.9a.

FIGURE 3.9: Node lifetime for different current values in sleep and active mode.

3.2.3 Related Work

The energy consumption model in WSN has been of considerable interest in the last years to the research community. Proof of this is the quantity of related research publications. We can find works that use the deterministic approach that we proposed in Section 3.2.1 or other works that try to solve the problem of complexity encountered by such as approach. In this Section, we present some of those works related to models and we discuss the different points treated in these studies.

Deterministic models are often used with WSN simulators. For instance, a generic theoretical energy consumption model is presented in [163]. The authors develop an energy model for each principal hardware component in the node: processor, transceiver and sensor. These energy models follow the concept of the expressions (3.2)-(3.5). The processor energy model (PEM) is composed of three states called *sleep*, *idle* and *run*. A power value characterizes each

state and the time for the transitions between these states is also indicated. In transceiver energy model (TEM), six different states are identified: Tx, Rx, Off, Idle, Sleep and Clear Channel Assessment and Energy Detection (CCA/ED). Energy consumption in Tx and Rx states follows the (3.5) expression where these energy depends on the data rate of the communication and on the length of the packet. The transition time for all the changes of states are also considered. Sensor energy model (SEM) consists of one state (*sensor-run*) and two transitions (*on-off, off-on*). The node energy model (NEM) involves the three models PEM, TEM and SEM whom are evaluated in parallel. Using OPNET simulator [164] with 20 nodes of a WSN randomly distributed in an area, they conclude that the nodes located in the center of the area consume more energy than those which are in the edges due to the routing tasks.

In [165], the energy model proposed by the authors exclusively focuses on the RF module in the node. This model consist of a FSM with four states: transmission, reception, idle and sleep. In transmission state, the data rate of the communication as well as the length of the packet are considered. Moreover, the power required by the transceiver depends on the power used by its amplifier which depends on the minimum output power required for the receiver to listen a packet correctly. This minimum output power (P_{min}) is calculated according to the Friis transmission equation, shown in (3.6).

$$P_{\text{min}} = \left(\frac{4\pi R}{\lambda}\right)^2 \cdot \frac{1}{G_t G_r} \cdot P_r \tag{3.6}$$

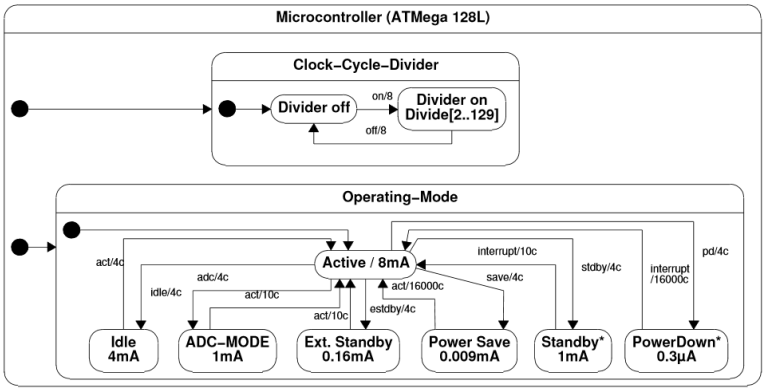
where R represents the distance between transmitter and receiver, λ is the wavelength, G_t and G_r correspond to the antenna gain in both sides, transmitter and receiver, and P_r is the power required by the receiver to receive a packet, generally called sensitivity. In receive state, the authors consider the energy consumed in the reception as well as in the decoding of the packet. The idle state corresponds to the time that the transceiver listens to the channel but, also, for the CCA activities. The transitions between the states are not considered in this model. Again, a simulation of a node of WSN is done in order to calculate the energy consumed by each state in the transceiver model.

A similar study is presented in [166], where energy consumed depends on the output transmit power of the nodes. The output transmit power is chosen according to the link connection: the received power value is measured through the RSSI in receiver node and, then, the value for transmit power is evaluated. The results in this study show the number of nodes that die in a network after some time and according to some parameters as the number of nodes in the network or the distance between them. These results have been obtained after simulations in three different simulators: OMNeT++, C++ and ns-2 simulators.

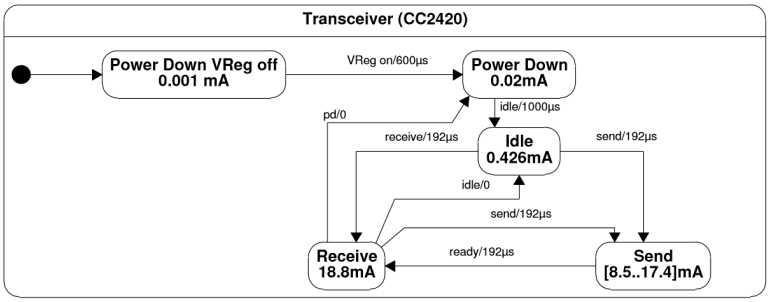
In [167], another energy consumption model which depends only on the energy consumed by the transceiver is presented. In this case, the authors go into more details. The power in all

the parts of the transceiver is concerned: the baseband digital signal processing (DSP) circuit, the front-end transmitting and receiving circuits as well as the amplifiers in transmitting and receiving parts. The characteristics of the path loss and of the antennas are also considered. For a transmission distance R, the energy consumed by the whole network in a multi-hop configuration with n hops is defined. Then, the authors show interesting results about the total energy consumption according to the distance R and the number of hops n. Three types of transceivers are evaluated: TI CC1000 [168] at 433 MHz, TI CC1000 at 868 MHz and TI CC2420 at 2.4 GHz. The power values are extracted from their datasheets. Concerning the communication between nodes, the authors assume a simplified Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [135] with Request-To-Send (RTS), Clear-To-Send (CTS) and acknowledgment (ACK) frames. The length of each frame and of each data packet is also considered. Thanks to this detailed theoretical model, the energy consumed by the transmitting unit of the nodes in the whole network is calculated and, then, an optimization of the routing in the network is possible, as with the Best Effort Routing Algorithm (BERA) proposed by the authors.

In [169], D. Schmidt *et al.* present an energy model of a MICAz [80] mote, also based on a FSM. States as well as transitions between states are considered. Figure 3.10a and 3.10b show the FSMs of the energy model in microcontroller and transceiver, respectively. We can observe



(A) FSM in microcontroller



(B) FSM in transceiver

FIGURE 3.10: Energy model in microcontroller and transceiver of MicaZ [169].

that each state is defined by a current level and the time of the transitions is also indicated.

The transitions in microcontroller are counted in clock cycles whereas in the transceiver, they are in seconds. The authors used the Avrora [82] simulator to test their energy model and to obtain the results.

In [170], W. Dargie presents an interesting study about the dynamic power management (DPM): a power management unit can dynamically vary the voltage level (dynamic voltage scaling (DVS)) or the clock frequency (dynamic frequency scaling (DFS)) in order to improve energy efficiency. The power consumption decreases when the voltage level decreases as well as when the clock frequency decreases. The author uses a deterministic approach for their model: he introduces an example of a node with three parts and several operating modes in each part: memory unit with three states, transceiver with three states and microcontroller with six states. Then, $3 \times 3 \times 6 = 54$ different global states. The DPM technique can be expressed as a state machine where the dynamic variations of voltage and frequency in circuit are made with regard to the task scheduling, the workload and the energy required, as shown in Figure 3.11.

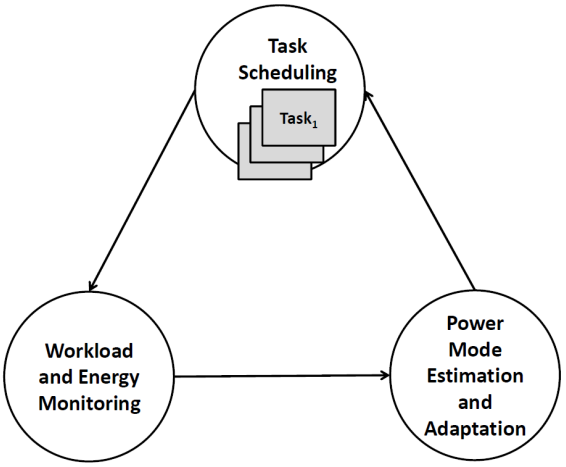


FIGURE 3.11: DPM method as a state machine [170].

Besides, the authors in [171] [172] define an overall energy consumption model in a network thanks to the Hierarchy Energy Driven Architecture (HEDA). They aim at calculating the lifetime of a network, taking into account all the elements in the network. The energy consumed contains different energy elements as the energy consumed by the electronic components in a node ($E_{individual}$), the effect of the neighbor nodes (E_{local}), the effect of all the nodes in the network (E_{global}), the possible energy harvesting recovered from the environment ($E_{battery}$) and the energy consumed by the communication between the node and the sink in the network (E_{sink}). In the individual energy component, the authors take into account the energy consumed by the microcontroller, the sensor, the transceiver and a external storage unit. Figure 3.12 shows the FSM of these types of nodes. The authors consider some operating modes in each electronic component but they forget others as the sleep mode in all components

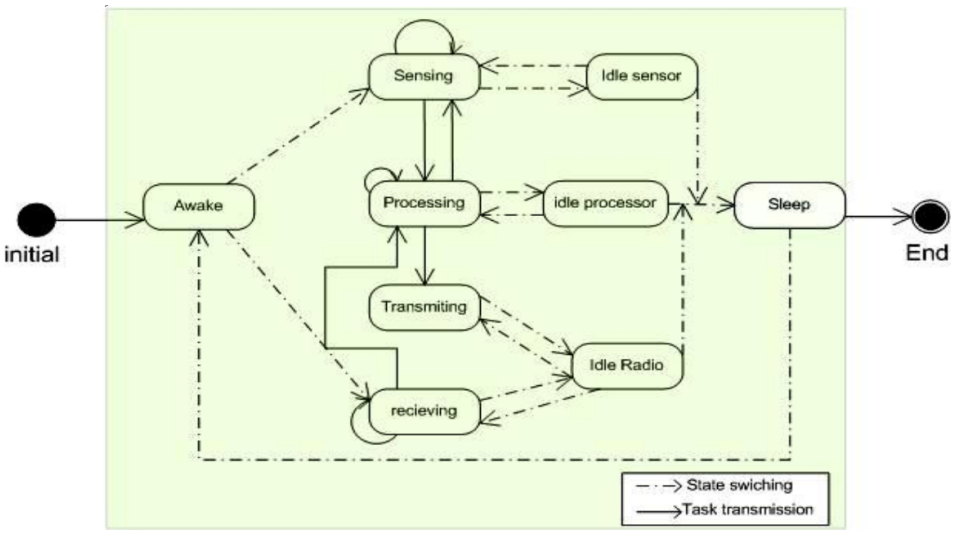


FIGURE 3.12: FSM of HEDA node [171].

or the serial communication between the microcontroller and the other components. However, in local communication between neighbor nodes, the model considers some elements as the neighbor monitoring, communication protocols, the effect the collisions as well as the overhead of the transceiver in the channel. The energy function of the global constituent takes into account the topology of the network, the packet routing, the packet loss and the protocol overheads in the whole network. This overall energy consumption model integrates many elements of the network, but in a superficial way. The lack of depth in the details of the different energy models (individual, local, global, sink) does not allow to reach a higher reliability in the model.

In [173], the authors compare a energy model based on Petri nets and another model based on Markov chains. A Markov model is a discrete-time stochastic model where the next state depends only on the current state. However, Petri nets is a simulated based approach for modeling that includes the states called *places* with intermediate states called *transitions*. The transitions decide the place to go based on some conditions. These conditions are introduced by the user as well as the firing distribution mode that indicates the delay of the process in the transition. A firing distribution can be instantaneous (without delay), exponential (according to a coefficient) or deterministic (a fixed delay). The Petri net model takes more time to calculate the energy consumed than the Markov chains because of the conditions (a time to decide the place to move) and the delays in the transitions. However, Petri net model can easily be modified and is much more flexible than Markov chains. According to this study, thanks to its dynamic way to work, the results obtained with the Petri net model are more accurate than those of the Markov model.

Many ways to model the energy consumption of a node in a WSN have been studied in the literature. We have explained in this Section some of these methods. Most of them express

an energy model as a FSM of the states in each electronic component in the node. Others add the part of the interaction between two nodes in the channel with the output transmit power, the minimum received power, the distance between nodes and the path loss in the channel. The models can also contain the energy consumed by the nodes in a multi-hop network when a packet is transmitted from a node to the sink. Other parameters as the collisions in the channel or the overhead in a receiver node are studied by some methods. In most of the cases, the current values for each state of the electronic components are extracted from datasheets. In other cases, these values are previously measured by a measuring instruments as an oscilloscope. In most of the cases, these models are created through mathematical expressions, *i.e.* they are deterministic. These models contain some limitations due to the complexity of introducing realistic energy values in the mathematical algorithms. Moreover, they cannot show some parameters of the real circuits as the effect of the current of some components in the current of the other components or the real current design of the transitions between states. This is why, it is important to create an energy model based on the statistics of a real circuit, from real measurements.

In Section 3.3, we present an energy consumption model created automatically from the real energy measurements in a node. An algorithm analyzes the energy measurements from the node and creates automatically a model which depends on the behavior of the node. This model is based on a Markov chain and allows to estimate the node lifetime. Furthermore, we can modify easily some parameters in the model to find out how to change the elements in the node (in hardware or in software) in order to optimize the energy consumption. Several experiments will be exposed to show the utility of this automatic energy consumption model.

3.3 Automatic Energy Consumption Model

We want to create an energy consumption model based on the current values obtained from the Synergie platform experiments (see Chapter 2). The main objective deals with connecting the energy measurements Synergie platform to a node in its real deployment environment and automatically analyzing its energy behavior. Another important goal is to detect the different states where the node consumes more energy and to optimize those states. This is why, we propose a different approach from the deterministic generic energy consumption model. This new model allows to find out the energetic behavior of the node and to obtain an accurate and realistic estimation of its lifetime.

This method can be called *Automatic Energy Consumption Model*. It consists in an algorithm which automatically creates an energy consumption model from the energy measurements in any existing WSN. It is based on a Markov model.

The main objectives are to identify the different levels of energy consumed by a node and to associate the period with similar consumption pattern in a single state in order to be able to optimize the different aspects (hardware, software, channel conditions, MAC protocol, routing protocol, among others) with the final purpose of minimizing the energy consumption.

A diagram of the Automatic Energy Consumption Model is shown in Figure 3.13. The energy measurements and the indicators given by the measurements platform are used to do an automatic statistical analysis. The objective of this analysis is to obtain the different states of the node, characterized by the current values (mean power consumptions) of the node's components and its duration as well as the transition probabilities between states. These results form a Markov model of the energy behavior of the node. These statistical data is composed by mean values of current, time and energy for each one of the states identified from the automatic statistical analysis. Moreover, probabilities of remaining in a state and probabilities of transition between states are also obtained from this analysis, which consists in a Markov model. Thanks to these statistical data and the Markov chain, we are able to estimate accurately the lifetime of a node if this node uses a certain battery.

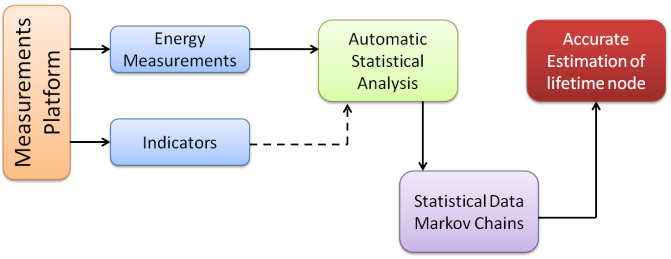


FIGURE 3.13: Diagram of the Automatic Energy Consumption Model; from the energy measurements to the estimation of the lifetime node.

The automatic statistical analysis is divided in several steps as represented in Figure 3.14. Firstly, the energy measurements are obtained and expressed in current values for each electronic component in the circuit. Then, we make an automatic segmentation according to the level of the current values. This segmentation is seeking to detect the breaking points in the current values, *i.e.* the changes between two current levels. We can then identify the periods without any changes in the current levels. We call these periods *individuals*. These individuals are characterized by different parameters as its duration, the mean current values and variance. With these parameters, we can group the individuals that are similar in

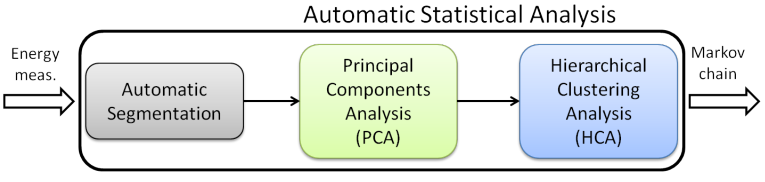
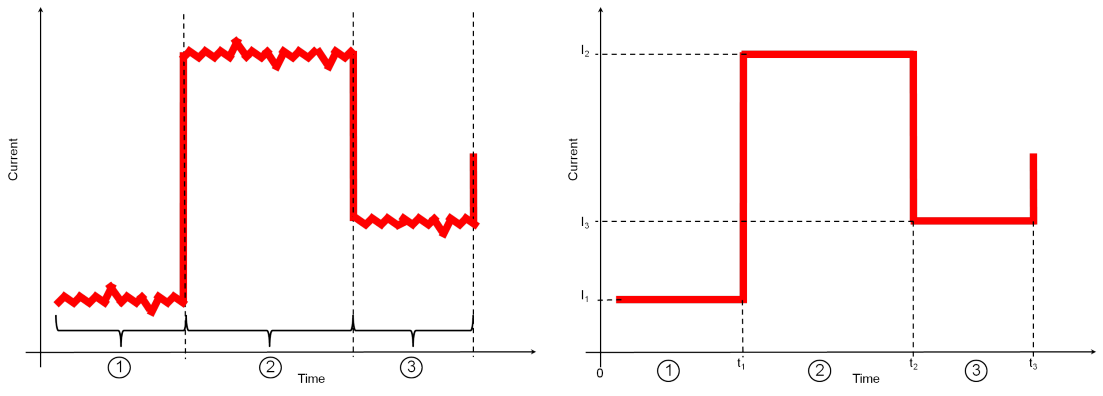


FIGURE 3.14: Diagram of the Automatic Statistical Analysis.

different clusters or classes. The algorithm which offers the information about the degree of differentiation between the individuals is called *Principal Components Analysis* (PCA). Moreover, we are able to classify the individuals according to their degree of differentiation using a statistical algorithm based on *clustering*. This algorithm is the *Hierarchical Clustering Analysis* (HCA). The results of this chain of algorithms are different individuals extracted from the real energy measurements and grouped in classes. Once the classes are established for all the individuals, we obtain the mean current value, the mean time value and, consequently, the mean energy consumption value for each node component in each class. Besides, the number of times that a class appears in the measurements is known as well as the transitions between classes, giving the transition probabilities between classes and, consequently, the Markov model. We can finally get an accurate and realistic estimation of the lifetime of the node. The development of these algorithms is detailed in the next Sections.

3.3.1 Automatic Segmentation Algorithm

The automatic segmentation aims at detecting the severe changes of current over the time of the test. These changes of current determine the breaking points and divide the current values in different periods called *individuals*. The individuals are characterized by a set of parameters: the duration, the mean and the variance of the current. Figure 3.15 shows the result of this algorithm. In Figure 3.15a, an example of a current graph is presented with some variations, as in real measurements. The algorithm should be able to detect the breaking points and to ignore the variations due to noise. Thanks to the detection of the breaking points, we can identify the individuals (*i.e.* ①, ② and ③ in the example of Figures 3.15a and 3.15b) and we can calculate the characteristic parameters.



(A) Detection of breaking points

(B) Individuals 1, 2, and 3 by time and current

FIGURE 3.15: Detection of breaking points in current to determine the individuals.

These parameters are placed in a table, as shown in Table 3.6, where the first column is occupied by the number of individuals (k), the second column contains the parameter of duration

of each individual (Δt_k), the third parameter is the mean current ($\bar{I}_k$) of the samples in the individual k, whereas the fourth parameter corresponds to the variance of the samples in the individual (σ_k^2). These two last columns are repeated for each hardware component of the node measured by Synergie platform.

TABLE 3.6: Table of the individuals with their parameters

k	Δt_k	$\bar{I}_k$	σ_k^2
1	$\Delta t_1 = t_1$ - 0	I_1	σ_1^2
2	$\Delta t_2 = t_2 - t_1$	I_2	σ_2^2
3	$\Delta t_3 = t_3 - t_2$	I_3	σ_3^2
...	...	...	...
n	$\Delta t_n = t_n - t_{n-1}$	I_n	σ_n^2

We have implemented two different methods to achieve the automatic segmentation:

- The *first method* is based on the identification of the different current levels in the whole experiment. A threshold increases gradually and the number of samples equals or greater than this threshold is counted. According to our experiments, this method is valid when the variation of the current values is small.
- The *second method* is more efficient for experiments with strong variations. It consists in detecting the breaking points through the evaluation of the standard deviation of the derivative of the current values.

We now explain in more detail these two methods.

3.3.1.1 Method 1

A) Removing the transitions: the first task deals with the elimination of transitions between two levels of current. The transitions between the states in the electronic components (idle, active, sleep, transmit mode, receive mode, among others) are generally slower than the sampling rate and include several samples. These increasing or decreasing transitions do not have to be considered as individuals. Consequently, they must be suppressed. To realize this task, we first derive the current values. Let $f(s)$ be the current of sample s. The derivative of a function f at a point x is defined by (3.7). If h has a fixed value different from zero, the expression is represented as (3.8). Then, the derivative of a function with discrete values, also

called *finite difference*, is expressed as (3.9).

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.7}$$

$$\frac{f(x+h) - f(x)}{h} = \frac{\Delta_h[f](x)}{h} \tag{3.8}$$

$$\Delta_h[f](x) = f(x+h) - f(x) \tag{3.9}$$

Once the finite differences of the current values are obtained, we identify sequences of p consecutive samples with positive or negative gradient: positive if $\Delta_h[f](s), \Delta_h[f](s+1), \ldots, \Delta_h[f](s+p) > 0$ or negative if $\Delta_h[f](s), \Delta_h[f](s+1), \ldots, \Delta_h[f](s+p) < 0$. We define this positive or negative gradient sequence as an ascending or descending transition, respectively, between two changes of current. Then, the samples belonging to these transitions $(f(s+1), f(s+2), \ldots, f(s+(p-1)))$ obtain the value corresponding to the nearest value of the delimiting points of the transition ($f(s)$ or $f(s+p)$). For $u \in [s, p]$, this is represented in (3.10). The differences between the value $f(u)$ and the end points of the transition are represented by $g_1(u) = |f(u) - f(s)|$ and $g_2(u) = |f(u) - f(s+p)|$.

$$f(u) = \begin{cases} f(s), & \text{if } g_1(u) \leq g_2(u) \\ f(s+p), & \text{if } g_1(u) > g_2(u) \end{cases} \tag{3.10}$$

The transition suppression is only used for the segmentation. The true transition values will be re-introduced in the classification steps and analysis because their impact should not be neglected.

B) Counting the samples: The next task is to count of the samples whose values are greater or equals to a determined threshold (th_i). This threshold is defined in the range $[\min f(s), \max f(s)]$. It is a variable threshold which increases at each iteration by a step of $\frac{\max f(s) - \min f(s)}{N_{\text{div}}}$, as represented in (3.11). For each iteration we get the number of values greater or equals to the threshold th_i. This amount of samples is represented in (3.12) as $N_{\text{segm}}(i)$.

$$\forall i \in \{0, 1, 2, ..., N_{\text{div}} - 1\}, \qquad \text{th}_i = \min f(s) + i \cdot \frac{\max f(s) - \min f(s)}{N_{\text{div}}} \tag{3.11}$$

$$N_{\text{segm}}(i) = \text{card}\{s \in T \quad / \quad f(s) \geq \text{th}_i\} \tag{3.12}$$

The choice of N_{div} is important. It is recommended to determine N_{div} according to the step of current values in the samples. If we choose a N_{div} too small, we confuse noise and level changes whereas if we choose a N_{div} too big, we will skip some changes.

C) Detecting the sets of samples: Once the count of samples is done in $N_{\text{segm}}(i)$, we look for instants when the number of samples after an increase in the threshold is significantly decreased. An example of the count of values greater or equal to the different thresholds th_i is represented in Figure 3.16. The first plot shows the number of samples in the experiment according to $\eta_i = \frac{i}{N_{\text{div}}}$ whereas the second plot represents the derivative of the curve in the first graph. When the derivative is close to zero, it means that the threshold increase does not cross a current level. If the derivative takes larger values, it probably means that a current level exists between the two consecutive thresholds. The objective is to identify these variations of the derivative.

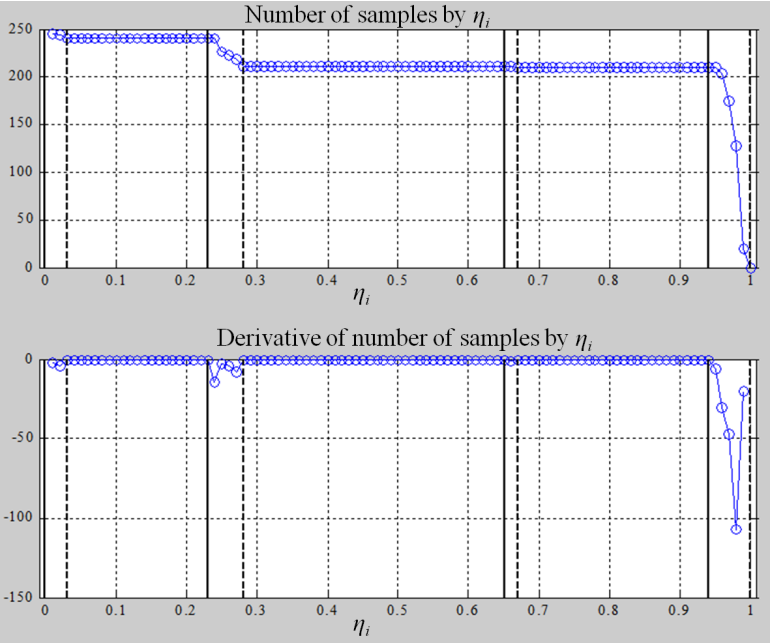


FIGURE 3.16: Detection of the variations of the number of samples according to the coefficient η_i.

In the example of Figure 3.16, the zones of significant variations in the number of samples are clearly identified. Two vertical lines are inserted by the algorithm to delimit each of these zones: solid line indicates the beginning of a zone whereas its end is shown by a dashed line. According to these vertical lines, in this example we can distinguish four zones where the variations exist: at the beginning of the tests, in the middle between $\eta_i = [0.2, 0.3]$, another small zone between $\eta_i = [0.6, 0.7]$ and the last one at the end for the values near $\eta_i = 1$. We decided that a part of the results in $N'_{\text{segm}}(i)$ is a zone of variations when we have $N'_{\text{segm}}(i) \neq 0$ for some consecutive numbers of i where $i \in [0, 1, 2, ..., N_{\text{div}}]$. Moreover, to avoid the maximum sensitivity and the effect of noise in some cases, the criteria selected says that at least three $N'_{\text{segm}}(i) = 0$ have to occur before and after the zones of variations. This criteria has two exceptions: when the first zone of variations begins in the first threshold ($\eta_i = 0$) and when the last zone of variations finishes in the last threshold ($\eta_i = 1$). For these exceptions only

three $N'_{segm}(i) = 0$ have to appear after or before these zones of variations, respectively. With this criteria, we confirm the identification of the true variation zones even if some cases of confusion may occur.

The vertical lines (solid and dashed) are exported to the original current values, as represented in Figure 3.17. The previous vertical lines that delimit the variation zones become horizontal lines and define the variation zones in the original current values. As before, a solid line represents the low threshold and a dashed line the high threshold. Figure 3.17 shows the zoom of a current measurement when the transmitter of the circuit wakes up to send a packet. Once the packet is sent and the ACK frame is received, the component goes back to sleep mode. We observe that the four different levels of current in the test have been detected and the width of the variation zones of these levels have been correctly determined by the action of the present algorithm.

Finally, each interval is identified by a different number and a sample is associated to the number of the interval that it belongs to (samples in the first interval have number 1, samples

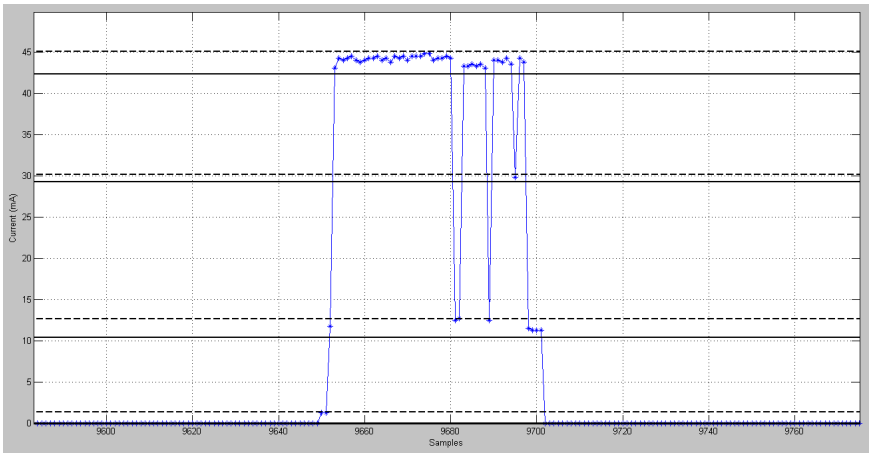


FIGURE 3.17: Variation zones shown directly in current values.

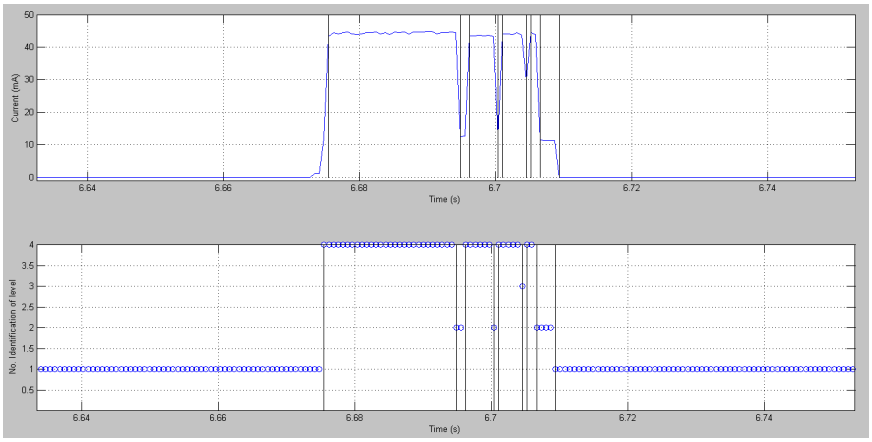


FIGURE 3.18: Breaking points detected in current values and identification of the variation zones.

in second interval have number 2, etc.). The breaking points are detected when a change of level is observed. Figure 3.18 shows in the first plot the current values and the breaking points detected by the algorithm whereas the second graph presents the identification number of each sample.

D) Combining the breaking points in several current sources: The algorithm developed can be applied to several current curves and the different results can be combined. For example, in Figure 3.19, the individuals are created not only from a current plot but from the combination of two current plots obtained from the energy measurements of two different electronic components in the circuit. The combination of the breaking points creates new individuals. The new individuals of this example are represented in Table 3.7 and characterized by five parameters: the duration (Δt_k), the mean current ($\bar{I}^1$ and $\bar{I}^2$) and the variance ($[\sigma^2]^1$ and $[\sigma^2]^2$) in both current plots. This method can be used for more than two current plots, making it a good method to characterize the individuals created from the current measurements

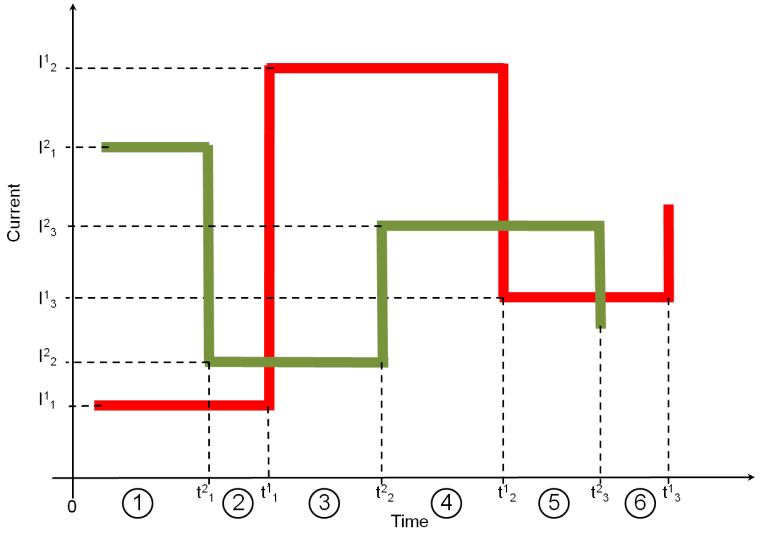


FIGURE 3.19: Determination of the individuals from two different plots of current values.

TABLE 3.7: Table of the individuals from the current values of two electronic components.

k	Δt_k	$\bar{I}^1$	$[\sigma^2]^1$	$\bar{I}^2$	$[\sigma^2]^2$
1	$\Delta t_1 = t_1^2 - 0$	I_1^1	$[\sigma^2]_1^1$	I_1^2	$[\sigma^2]_1^2$
2	$\Delta t_2 = t_1^1 - t_1^2$	I_1^1	$[\sigma^2]_1^1$	I_2^2	$[\sigma^2]_2^2$
3	$\Delta t_3 = t_2^2 - t_1^1$	I_2^1	$[\sigma^2]_2^1$	I_2^2	$[\sigma^2]_2^2$
4	$\Delta t_4 = t_2^1 - t_2^2$	I_2^1	$[\sigma^2]_2^1$	I_3^2	$[\sigma^2]_3^2$
5	$\Delta t_5 = t_3^2 - t_2^1$	I_3^1	$[\sigma^2]_3^1$	I_3^2	$[\sigma^2]_3^2$
6	$\Delta t_6 = t_3^1 - t_3^2$	I_3^1	$[\sigma^2]_3^1$	I_3^2	$[\sigma^2]_3^2$
...	...	...	...	...	...
n	$\Delta t_n = t_n - t_{n-1}$	I_n^1	$[\sigma^2]_n^1$	I_n^2	$[\sigma^2]_n^2$

of several electronic components in a circuit.

3.3.1.2 Method 2

The first method has been tried in different tests. We observed that it is efficient when the variance of the individuals is not too large. When significant variation of the current values in the individuals appears, it has difficulties to identify the limits between them. This is why, we have developed a second method which is more efficient in these situations.

The second method is based on the identification of the changes of current level higher than a threshold κ. To achieve this, the process follows several stages.

A) Removing the transitions: As in Method 1, the transitions between states must not interfere with the identification of the breaking points. Then, the first task in this method is the same as the first task in Method 1.

B) Deciding a type: At this point, we have to decide which level of differences of current corresponds to a change state. For that, we need to set a threshold κ. This threshold is determined by the standard deviation (σ) of the derivative function of the samples in the test:

$$\kappa = \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{3.13}$$

where x_i represents the derivative value, $\bar{x}$ the mean value of the derivative and N the total number of values in the experiment. The number N should be a representative number of values in the experiment.

C) Detecting the breaking points: the derivative function of the discrete values was defined in 3.9. We define the same derivative function without the transition points as $\Delta_h[f_2](x)$. Then, we consider a point $f(u)$ as a breaking point when the condition (3.14) is fulfilled.

$$|\Delta_h[f_2](u)| > \mu_2 + \kappa \tag{3.14}$$

where μ_2 corresponds to the mean value of the derivative function $\Delta_h[f_2](x)$.

D) Combining the breaking points: As in Method 1, it is possible to combine the breaking points of different current sources.

After some experiments with this second method, we have verified that it works correctly when the variance of the samples in the test is not very low. In other case, the standard deviation could be lower than noise detected in the energy measurements. In any cases,

both methods, Method 1 and Method 2, can be combined in order to identify correctly the breaking points in the measurements.

Next Section describes the next algorithm used to achieve the automatic energy consumption model.

3.3.2 Principal Component Analysis

The *Principal Component Analysis* (PCA) is an algorithm used in many fields of research, for instance genetics [174][175], demographics [176][177], image processing [178], finances [179], among others. PCA constitutes an exploratory statistical method which allows an essentially graphic description of the information included in large tables of data. In most applications, it deals with studying p variables measured on a set of n individuals. n and p are generally large, then, the goal is to synthesize the amount of information into an usable and comprehensible form. Thanks to the tools of the descriptive statistics, it is possible to study the different variables through graphical or numerical summaries as the mean, the variance or the correlation.

The study of the variables one to one is required in any statistical analysis but it is insufficient because the links between the different variables are the most important aspects here. Then, it is better to analyze the data according to their multidimensional nature. If we consider two variables simultaneously (x^1 and x^2 for example), the set of data is easy to be represented in a planar graph. The simple visual inspection of the appearance of the collection of points $\left\{(x_i^1, x_i^2), i = 1, ..., n\right\}$ allows to get an idea about the form and the intensity of the connection between these two variables as well as to notice the individuals and the groups of individuals with similar characteristics. In the case of considering three variables simultaneously (x^1, x^2 and x^3 for example), the visual examination is still possible with the geometry in the space. This time the collection of points $\left\{(x_i^1, x_i^2, x_i^3), i = 1, ..., n\right\}$ can be observed entirely through a three-dimensional plot. When we consider a number p of variables, with $p \geq 4$, the direct and total visualization becomes impossible. Here, it is necessary to find another way to display the multidimensional data.

The method used by PCA consist in projecting the set of points on one or several planes to obtain the closest representation to the exact configuration. Then, the intention of this method is to minimize the distortions between the set of points projected and the original set of points. PCA is a technique to reduce the dimension by converting a quantity of variables, often correlated among them, in a smaller quantity of non-correlated principal components.

The data are generally represented as a rectangular matrix with n rows corresponding to the *individuals* and p columns corresponding to the *variables*. The choice of the individuals and

the variables is an essential process which has a strong influence in the results of the PCA. This choice has to be done according to the objectives of the study. The selected variables must describe and show as effectively as possible the differences between the individuals by knowing that the PCA treats the variables as quantitative, *i.e.* all the variables are of equal importance, and not as qualitative.

3.3.2.1 Definition of the data matrix

The data is represented in a matrix format, as in (3.15). The dimensions of the matrix are $n \times p$ where the individuals $x_1, ..., x_i, ..., x_n$ are situated in the rows whereas their variables $x^1, ..., x^j, ..., x^p$ are in the columns. Each individual can be expressed as a point which belongs to $\mathbb{R}^p$.

$$\begin{aligned} X &= \left[x_i^j\right] \quad \text{pour } i = 1, \ldots, n \text{ et } j = 1, \ldots, p, \\ X &= \begin{bmatrix} x_1^1 & \ldots & x_1^p \\ \vdots & \vdots & \vdots \\ x_i^1 & x_i^j & x_i^p \\ \vdots & \vdots & \vdots \\ x_n^1 & x_n^j & x_n^p \end{bmatrix} \end{aligned} \tag{3.15}$$

where x_i^j represents the value of the variable j taken from the individual i.

3.3.2.2 Center of gravity or midpoint

The next step is to find the center of gravity or midpoint for the set of points. Center of gravity is called to the vector g of the arithmetic mean values of each one of the p variables. It is defined by $g = (\overline{x}^1, \ldots, \overline{x}^p)$ with $\overline{x}^j = \sum_{i=1}^n p_i x_i^j$ and where p_i corresponds to the weight associated to the i^{th} individual. Normally, the whole of the individuals are of equal importance, then, the weight $p_i = \frac{1}{n}$. However, in some cases, we can group some identical individuals in only one individual to simplify the calculation, then, the weight p_i becomes different. These weights are expressed together in a diagonal matrix of size n as in (3.16).

$$D_p = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & p_n \end{bmatrix}, \tag{3.16}$$

where the weights p_i for every individual i are $p_i \geq 0$ and the sum of the weights $\sum_{i=1}^n p_i = 1$.

Thus, the center of gravity g is expressed in (3.17) by the matrix equation g whereas the centered matrix Y associated to X is defined as (3.18). Then, in matrix Y, we put the original data placed in X around the point $(0,\ldots,0)\in\mathbb{R}^p$.

$$g = X'\mathbf{D_p}\mathbb{1} \qquad \text{where } \mathbb{1} = (\mathbf{1},\ldots,\mathbf{1})\in\mathbb{R}^{\mathbf{p}} \tag{3.17}$$

$$Y = X - \mathbb{1}g' \tag{3.18}$$

3.3.2.3 Variance and correlation matrix

We define the variance of the variable j in (3.19) as the square of the standard deviation s_j whereas the covariance of the variables k and l is represented in (3.20). These coefficients are included in the covariance matrix expressed as V in (3.21). This covariance matrix includes the variance of each variable j in the diagonal while the covariances between the variables are situated in the rest of positions. Note that the covariance matrix is a symmetric matrix where $v_{kl} = v_{lk}$. Moreover, the correlation of the variables k and l is expressed in (3.22) and the matrix form of the correlation matrix is represented in (3.23), where V corresponds to the covariance matrix and $D_{1/s}$ is the diagonal matrix containing the inverse of the standard deviation of the variables k and l.

$$s_j^2 = \sum_{i=1}^{n} p_i (x_i^j - \overline{x}^j)^2 \tag{3.19}$$

$$v_{kl} = \sum_{i=1}^{n} p_i (x_i^k - \overline{x}^k)(x_i^l - \overline{x}^l) \tag{3.20}$$

$$V = X'\mathbf{D_p X} - \mathbf{g}\cdot\mathbf{g}' = \mathbf{Y}'\mathbf{D_p Y} \tag{3.21}$$

$$r_{kl} = \frac{v_{kl}}{s_k s_l} \tag{3.22}$$

$$R = D_{1/s_k}\cdot V \cdot D_{1/s_l} \tag{3.23}$$

3.3.2.4 Centered and reduced matrix

After obtaining the centered matrix Y, we reduce the values in Y. The aim of these reduction is to give the same importance to each one of the variables of the initial data matrix, even if the difference between the set of values of these variables is considerable. Each column of the matrix Y is divided by its variance s_j^2. The centered and reduced matrix Z is represented in (3.24), where $D_{1/s}$ corresponds to the diagonal matrix with the inverse of the variances as

elements ($D_{1/s} = \text{diag}(\frac{1}{s_1}, \ldots, \frac{1}{s_p})$). Each element of Z is defined as expressed in (3.25).

$$Z = Y \cdot D_{1/s} \tag{3.24}$$

$$z_i^j = \frac{x_i^j - \overline{x}^j}{s^j} \tag{3.25}$$

3.3.2.5 Main axis

The PCA corresponds to a factorial method where the reduction of the number of variables is not done by a simple selection of several of them, but by the modification and the creation of different synthetic variables called *principal components*. The PCA is a linear method because the principal components are created by linear combinations of the initial variables. Geometrically, the n individuals are projected in a plane. The objective deal with choosing the plane of projection where the distances between the set of points become the clearest to differentiate these points in the plane as well as possible.

The criteria to decide the correct plane consists to maximize the mean of the square of the euclidean distances between the projections $f_1, \ldots, f_n$. Note that the distance between the projections is always smaller than the distance between the original points because the original data matrix has been reduced, *i.e.* $\text{d}(f_i, f_j) \leq \text{d}(x_i, x_j)$, where f_i and f_j represents the projections on the plane of x_i and x_j, respectively. To determine this plane, called *main plane*, it is necessary to find two perpendicular straight lines Δ_1 and Δ_2 which define the plane. The square of the euclidean distance of the projections in straight lines Δ_1 and Δ_2 is represented in (3.26).

$$\text{d}^2(f_i, f_j) = \text{d}^2(\alpha_i, \alpha_j) + \text{d}^2(\beta_i, \beta_j) \tag{3.26}$$

where α_i and β_i are the coordinates of the projection (f_i) of the individual i found on the straight lines Δ_1 and Δ_2, respectively. Same case for the individual j with α_j and β_j.

Thus, this method consists to search firstly the straight line Δ_1 which maximizes the mean of the euclidean distances $\text{d}^2(\alpha_i, \alpha_j)$ between all the points, then, to search the straight line Δ_2 that maximizes the mean value of $\text{d}^2(\beta_i, \beta_j)$. Besides, the method continue out of this plane to find the straight lines $\Delta_3, \ldots, \Delta_p$, perpendicular between them, which get the maximal square of the distances corresponding to the n individuals. The straight lines Δ_m, $m = 1, \ldots, p$ are called the *main axis* of the set of points. The maximum euclidean distances between the projections of the points on the planes are determined by the total inertia.

The *total inertia* of a set of points $x_i, i = 1,\ldots,n$ is defined in (3.27) as the weighted mean value of the square of the distances from the n points to the center of gravity g.

$$\text{Inertia} = I(g) = \sum_{i=1}^{n} p_i \mathrm{d}^2(x_i, g) \tag{3.27}$$

where, generally, the weight of every element is the same, then, $p_i = \frac{1}{n}$.

The total inertia measures the separation of the points according to the center of gravity, *i.e.* the global dispersion of the set of points. For instance, a total inertia of zero or near to zero means that all the individuals are identical or very similar and their values are the center of gravity g or near to g.

The total inertia can be also represented as the trace of the covariance matrix V, expressed in (3.28). It is easy to calculate according to the chosen metric M. If it is really used the covariance matrix to obtain the total inertia, $M = I_p$ and the result is the sum of the square of the standard deviation of each variable, as in (3.29). However, the correlation matrix can be also used to calculate the total inertia with a metric of $M = D_{1/s^2}$ and, then, the inertia corresponds to the number of variables, as in (3.30).

$$\text{Inertia} = I(g) = \text{tr}(\text{MV}) \tag{3.28}$$

$$\text{Inertia} = \sum_{j=1}^{p} s_j^2, \quad \text{with } M = I_p \tag{3.29}$$

$$\text{Inertia} = p, \quad \text{with } M = D_{1/s^2} \tag{3.30}$$

In conclusion, the act to search the main plane in PCA means to search the plane when the total inertia becomes maximal for the set of points projected.

3.3.2.6 Eigenvalues and eigenvectors

The goal now is to determine the straight lines $\Delta_1, \Delta_2, \ldots, \Delta_p$. This determination is done by the eigenvalues and the eigenvectors from the covariance matrix or, also, from the correlation matrix. In this case, we have used the method with the correlation matrix, thus, by using the metric $M = D_{1/s^2}$.

Firstly, the eigenvalues are calculated from the expression (3.31) where R is the correlation matrix, λ_p represents the different eigenvalues and I_p is the p-by-p identity matrix. Once the eigenvalues are calculated, the eigenvectors are obtained from (3.32), where v_p represents the different eigenvectors.

$$\det(R - \lambda_p I_p) = 0 \tag{3.31}$$

$$Rv_p = \lambda_p v_p \tag{3.32}$$

The straight line Δ_1, which represents the *main axis*, is determined by the eigenvector v_1 with the most important eigenvalue (λ_1), *i.e.* the eigenvalue with the highest value. The second highest eigenvalue λ_2 determines the second main axis from the straight line Δ_2 created by the corresponding eigenvector v_2. All the axis are established by the rest of eigenvectors $(v_3, ..., v_p)$ similarly. Note that $\lambda_1 \geq \cdots \geq \lambda_p$ and the total inertia corresponds to the sum of eigenvalues (3.33).

$$I(g) = \lambda_1 + \cdots + \lambda_p \tag{3.33}$$

The *main planes* are composed by a combination of the different main axis. Then, the main plane which contains the most information about the principal components is created by the two most important main axis (Δ_1 and Δ_2). The quality of the representation in a main axis k is represented by the quotient (3.34) whereas the quality of representation in the main planes composed by the first k main axis corresponds to (3.35). This approach allows to know the loss of information when we discard a number of axis. We have explained above that the objective of the PCA is to obtain a simplified representation of the set of points from the data matrix X. Then, the quantity of information kept by the first k main axis is given in (3.35). The decision to use an amount of main axis will depend on two elements: the simplification of the representation of the data (less main axis is better) and the quantity of the information wanted (more main axis is better).

$$\frac{\lambda_k}{I(g)} \tag{3.34}$$

$$\frac{\lambda_1 + \cdots + \lambda_k}{I(g)} \tag{3.35}$$

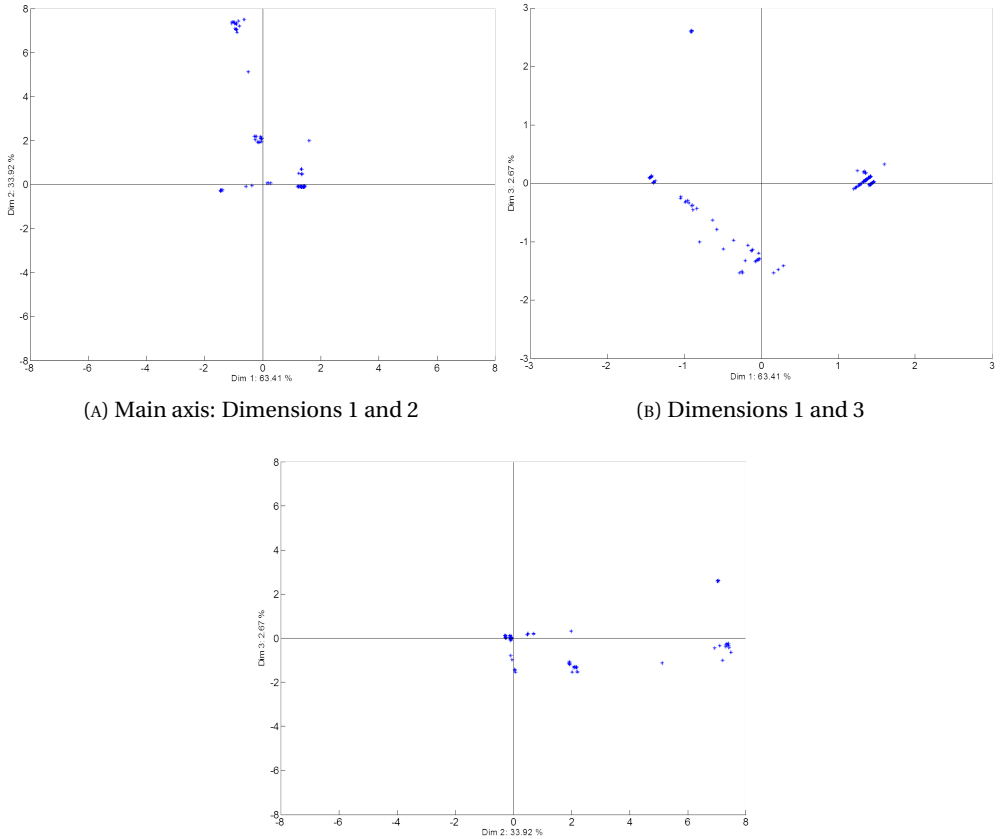
3.3.2.7 Principal components

The principal components constitute the new synthetic variables defined by the initial data and the eigenvectors. The projection of the individual i, with the initial coordinates $(x_i^1, \ldots, x_i^p)$, on the main axis creates new coordinates $(c_i^1, \ldots, c_i^p)$. These new coordinates represent the coordinates of the new synthetic variables called *principal components*. Every principal component corresponds to a linear combination of the p initial variables $x^1, \ldots, x^p$ and the components of each eigenvector $(u_1^j, u_2^j, \ldots, u_p^j)$. The linear combination is expressed in (3.36).

$$c^j = u_1^j x^1 + u_2^j x^2 + \cdots + u_p^j x^p \tag{3.36}$$

An example of the principal components is shown in Figure 3.20 seen by three different planes. We can observe some groups of points in the axis. The points which keep together

correspond to individuals with similar values in their variables. In the follow algorithm, the distance between the points of the experiment will be decisive to classify all the points in different clusters. Another interesting parameter is the percentage of the quantity of information offered by each axis. In Figure 3.20a, we have 63.41% of information represented in the main axis (x-axis) and 33.92% in the second main axis (y-axis). The total information offered by these two first main axis is the sum of the individual quantities, this is $63.41\% + 33.92\% = 97.33\%$. In this test, we work just with three variables in the initial data matrix (time, current in microcontroller and current in radio module), then, we can represent until three dimensions, using three axis. Figure 3.20b depicts the same points but in the plane formed by the dimensions 1 and 3. Note that in this second plane the maximum values in the axis is different that in the main plane (3 instead of 8). The dimension 1 and the dimension 3 contains 63.41% and 2.67% of information, respectively, which is $63.41\% + 2.67\% = 66.08\%$ information in total. Figure 3.20c shows the same principal components in dimensions 2 and 3, that being the less important plane with the $33.92\% + 2.67\% = 36.59\%$ of information. We remind that the percentage of information in each axis is calculated from the total inertia of the ensemble of points with respect to these axis. Then, if in Figure 3.20c, we can observe that some principal components are further than the principal components in the main plane of



(A) Main axis: Dimensions 1 and 2

(B) Dimensions 1 and 3

(C) Dimensions 2 and 3

FIGURE 3.20: Principal components.

Figure 3.20a, the appearances are deceptive. The set of points in coordinates around (-1,7) of Figure 3.20a is much larger than the set placed around coordinates (8,-1) of Figure 3.20c. Thus, the total inertia is higher in plane 1-2 than in plane 2-3.

3.3.3 Hierarchical Clustering Analysis

The *Hierarchical Clustering Analysis* (HCA) is a method also placed in the field of statistics called *clustering*. The main objective of this method deals with classifying a set of points in $\mathbb{R}$ according to the euclidean distances between these points [180]. The groups of the points formed by the method are called *clusters* or *classes*. The criteria to determine what points are included in each one of the classes is studied in this Section. This work has been made specially on the basis of the explanations of F. Husson *et al.* [181].

Firstly, the R package FactoMineR [182] has been studied to fully understand the HCA. For that, we have used the R software [183]. R is a free software environment and programming language dedicated to statistics and to the analysis of large volumes of data. When this environment is installed for the first time, it contains the basics functions and commands to start the mathematical analysis and calculations. However, the main interest to use this programming language is the sharing of the behaviors in the research community by creating the free add-on packages. We searched a package created to solve problems of clustering and containing the methods PCA, explained in previous Section, and HCA. This package has been created by F. Husson *et al.* [184]. The PCA function converts the input data in principal components. Then, these principal components are used as an argument for the function HCPC of the package which executes the method HCA. This package, FactoMineR, has been studied in order to acquire knowledge about the PCA and the HCA algorithms. However, the final implementation of this method has been carried out in MATLAB [124].

The following paragraphs contain an accurate explanation about the implementation of the HCA to analyze the energy consumption on a node of WSN.

3.3.3.1 Distance between points

Firstly, the distance between all the points which represent the principal components is calculated. The distance metric used corresponds to the squared euclidean distance of two points, represented in (3.37) as the sum of the squared distance of all the coordinates that form the points.

$$d^2(x_i, x_k) = \sum_{j=1}^{p} (x_i^j - x_k^j)^2 \tag{3.37}$$

where the euclidean distance between the points x_i and x_k is calculated for each coordinate j, where $j = 1,\ldots,p$.

Note that the points used here correspond to the principal components obtained previously. The number of coordinates in each point coincides with the number of variables, *i.e.* the number of columns ($j = 1,\ldots,p$), in the original data matrix X seen in (3.15). If the number of variables is high, we can depreciate the less important variables. The weight of each dimension is established by the PCA algorithm, as explained in Section 3.3.2.

The squared euclidean distances between each one of the points are inserted in a triangular matrix with the diagonal equals to zero, as shown in Table 3.8.

TABLE 3.8: Distances between the points in $\mathbb{R}$

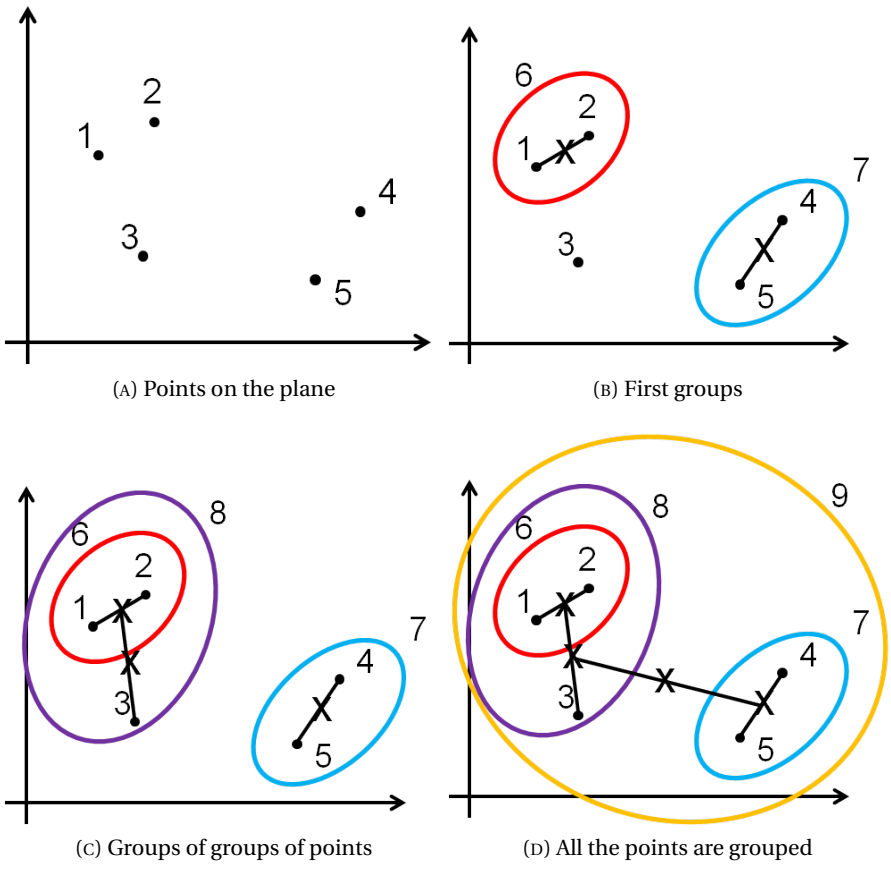
$k \setminus i$	1	2	3	...	$n-1$	n
1	0			...		
2	$d^2(x_1,x_2)$	0		...		
3	$d^2(x_1,x_3)$	$d^2(x_2,x_3)$	0	...		
...	...	...	...	...	...	...
$n-1$	$d^2(x_1,x_{n-1})$	$d^2(x_2,x_{n-1})$	$d^2(x_3,x_{n-1})$	...	0	
n	$d^2(x_1,x_n)$	$d^2(x_2,x_n)$	$d^2(x_3,x_n)$	...	$d^2(x_{n-1},x_n)$	0

3.3.3.2 Dendrogram

The following step of this analysis deals with grouping the points depending on the distances between them. The points which are closer between them will form groups of points, as shown in Figure 3.21. At the same time, a *dendrogram* or *tree* of groups of points is created according to the distance between these points. The dendrogram created from the example of Figure 3.21 is shown in Figure 3.22.

In the example given, five points are presented in Figure 3.21a. These points are placed with different distances between them. The grouping is done by taking the two nearest points for each time. The set of this two points (points 1 and 2 in the example) creates a new point (point 6) whose position is determined depending on the method used. It exists different methods to determine the position of the new points formed by the groups of points. Some of these methods are presented as follows:

- *Average*, the average distance between two points or two groups of points.
- *Single*, the shortest distance.
- *Complete*, the furthest distance.



(A) Points on the plane (B) First groups

(C) Groups of groups of points (D) All the points are grouped

FIGURE 3.21: Example of grouping the points according to the distance.

– *Ward*, based on Ward's method [185].

If we use the *average method*, the new point created by the set of two points is placed just in the middle of the distance between both points. On the other hand, if the *single method* is used, the position chosen in this case will be determined by the shortest distance between both set of points, *i.e.* when the distance between two groups of points is calculated, we take the nearest points between both groups. However, the *complete method* follows the opposite view than the single method. The evaluation of the distance between two set of points in complete method is done with the furthest points of both sets. Besides, the *Ward's method* is also studied, which works with the difference of inertia when the different set of points are formed.

In the example of Figure 3.21a, the method used is the *average method*. We can observe in Figure 3.21a that the new points (6 and 7) are determined by the average distance between the points which form these new points (1 and 2, 3 and 4, respectively). The position of these new points are represented in the example as the symbol X. Now, we consider that there are only three points in the example (points 3, 6 and 7). The process continues to execute similarly. The points nearest are grouped, as shown in Figure 3.21c, to create a new point

(point 8) and its position is calculated in the same way. Finally, all the points are grouped in Figure 3.21d and the squared euclidean distances between all the groups formed are known. Once this information is recovered, the dendrogram is made to show clearly the groups of points and their distances.

The dendrogram of this example is shown in Figure 3.22. The dendrogram is also called *tree* and it represents the links between the points in the grouping process. The different points are placed in the horizontal axis whereas the squared euclidean distances are indicated in the vertical axis of the dendrogram. The difference of distances between the groups of points is important for the classification. The black horizontal line in the figure represents the level where the algorithm cut the dendrogram to choose the number of groups in classification. In the example, the chosen level is after creating the group 8 and, then, we have two set of points: group 7 and group 8. Once the level is chosen, these groups are called *classes* or *clusters*. The decision of this level and, thus, the decision of the number of classes can be taken manually, by the user, or automatically, by the algorithm. The ideal case would be to automate this energy consumption model as much as possible. This is why a new algorithm to decide the optimal number of classes for each set of points has been made.

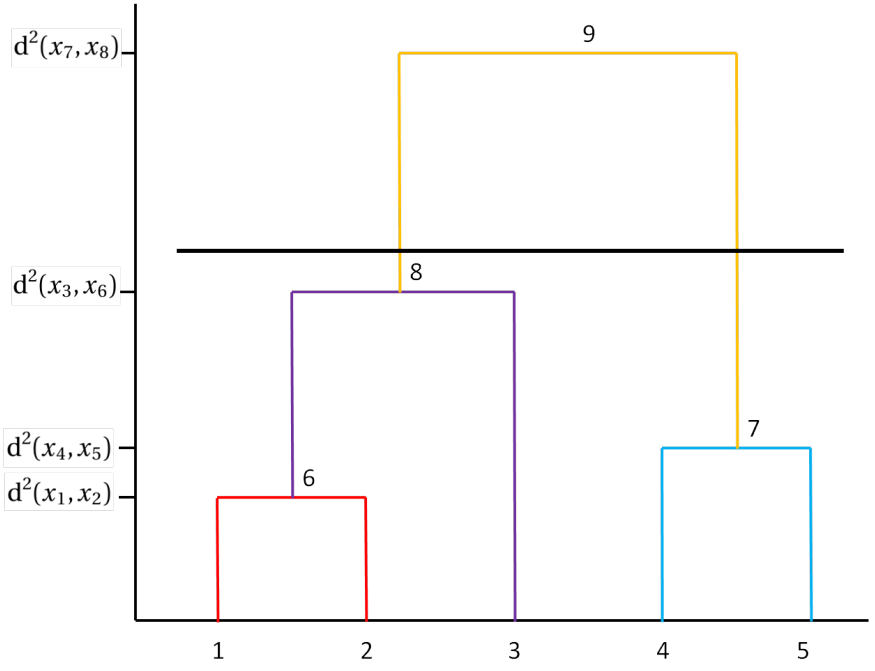


FIGURE 3.22: Dendrogram of the example of Figure 3.21

3.3.3.3 Automatic decision of number of classes

The main objective of this algorithm deals with choosing the optimal number of classes to divide the set of points of the experiment without loosing some important information. This information is lost when some points which are clearly distant are situated in a same class

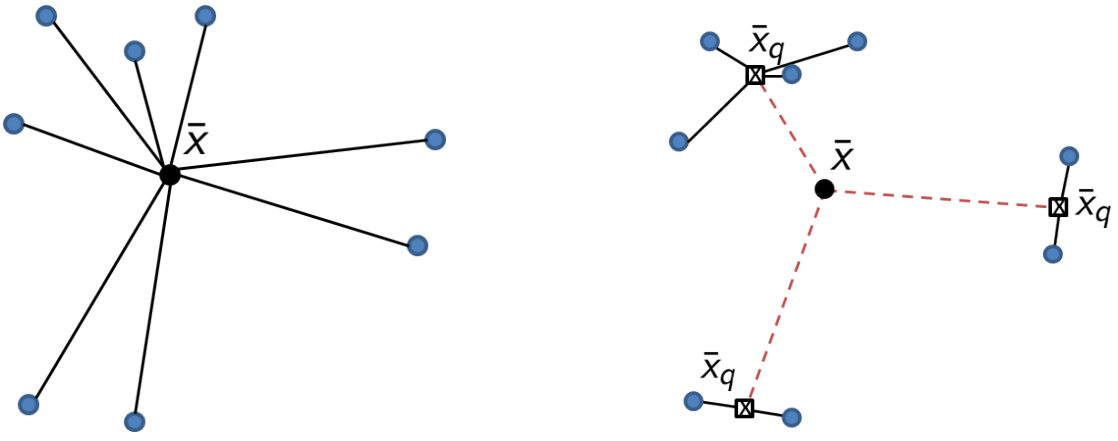
or, on the other side, some points that should be together are divided in different classes. For this reason, the Ward's criterion or Ward's method [185] is used in this algorithm.

Ward's method is called also *Ward's minimum variance method* and its purpose is to minimize the total within-cluster inertia. The total inertia is decomposed in two elements: within- and between-inertia. The *within-inertia* corresponds to the inertia calculated from the points in a class to the mean point of the same class ($\bar{x}_q$). However, the *between-inertia* is calculated from the mean values of each class to the total mean value of the set of points ($\bar{x}$), also called center of gravity. Figure 3.23a shows how the total inertia is calculated from all the points in the experiment. On the other hand, Figure 3.23b represents both: the within inertia (solid black lines) and the between-inertia (dashed red lines). The mathematical expression of the relation between the total inertia and the within- and between-inertia is expressed in (3.38) and (3.39). We observe in these expressions that the calculation of the different types of inertia is done with the squared euclidean distances between the respective points for each case.

$$\text{Total inertia} = \text{within-inertia} + \text{between-inertia} \tag{3.38}$$

$$\sum_{q=1}^{Q}\sum_{i=1}^{I}(x_{iq}-\bar{x})^2 = \sum_{q=1}^{Q}\sum_{i=1}^{I}(x_{iq}-\bar{x}_q)^2 + \sum_{q=1}^{Q}\sum_{i=1}^{I}(\bar{x}_q-\bar{x})^2 \tag{3.39}$$

where x_{iq} represents the point of the individual i which is a member of the class q, $\bar{x}$ is the mean point of all the set of points and $\bar{x}_q$ corresponds to the mean point of the class q. I and Q represents the total number of individuals and the total number of classes, respectively. Note that to calculate correctly the within- and between-inertia, these values of inertia must be normalized with the number of individuals in each class and with the number of classes.



(A) Total inertia (B) Within- and between-inertias

FIGURE 3.23: Types of inertia with the set of points.

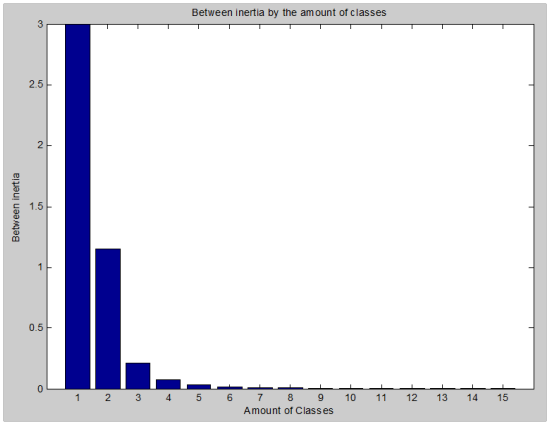
As shown in (3.38), the sum of both types of inertia, within- and between-, results the total inertia. Then, the relations between the types of inertia exist as represented in (3.40), where

the increase of the within-inertia cause the decrease of the between-inertia and vice versa. If R represents in this case the relation between the between-inertia and the total inertia, as expressed in (3.41), we find a good situation to classify a set of points when $R \longrightarrow 1$; in this situation the points of a class are close to the mean point of the same class. However, the situation becomes worse to classify, by contrast, when $R \longrightarrow 0$; in this situation the mean points of the different classes are close to the mean point of the whole set of points. This is the main condition in the algorithm to create an automatic classification and an automatic decision of the number of classes.

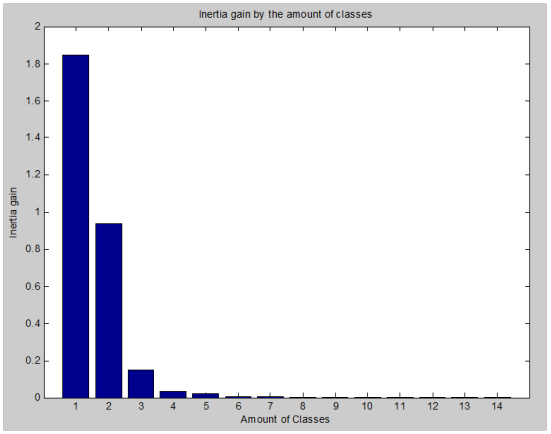
$$0 \leq \frac{\text{within-inertia}}{\text{Total inertia}} \leq 1, \qquad 0 \leq \frac{\text{between-inertia}}{\text{Total inertia}} \leq 1 \tag{3.40}$$

$$\frac{\text{between-inertia}}{\text{Total inertia}} = R \tag{3.41}$$

Now, we analyze the values of between-inertia according to the amount of classes determined to classify the set of points. An example of this type of graph is shown in Figure 3.24a. The first bar represents the inertia of the points with respect to the mean point of this class. If only a single class appears in the experiment, it means that the whole set of points are in one class and, thus, the between-inertia is equals to the total inertia, $R = \frac{\text{between-inertia}}{\text{Total inertia}} = 1$. Then, in this situation, the mean point in the experiment and the mean point in the class are the same. When the set of points are divided in two classes, the between-inertia, logically, decreases as shown in bar 2 of Figure 3.24a. We continue with the same process to calculate the between-inertia values until a number classes, in this case, we decided 15 divisions of classes as maximum value. Figure 3.24b shows the difference between the values of between-inertia for consecutive number of classes. This value is called *inertia gain*.



(A) Between-inertia depending on the number of classes



(B) Inertia gain depending on the number of classes

FIGURE 3.24: Between inertia and inertia gain vs. number of classes.

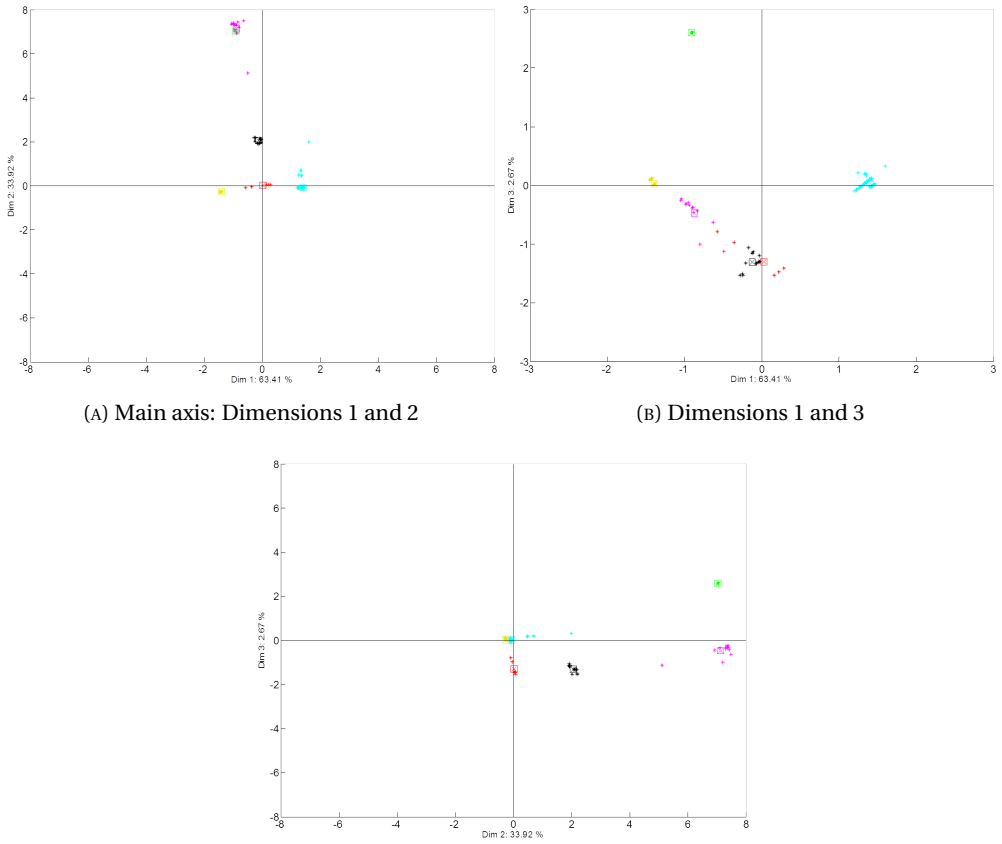
In this example, the difference of values of the between-inertia for one class and two classes is represented in bar 1 of Figure 3.24b. The process is the same for all the other number of classes. This algorithm offers a valuable information about the separation of the clusters of points or classes. We can determine the best number of classes to divide the set of points in the experiment. For that, we decide to introduce a threshold η_q in the values of the inertia gain. The expression to obtain this number of classes N_q is represented in (3.42). After some experiments, the threshold chosen has been $\eta_q = 0.01$ because at this value of inertia gain the clusters in the graphical representations describe better the possible divisions of points. The set of points are divided up to fifteen classes in order to search the first value of inertia gain less or equal to η_q. In the example of Figure 3.24b, the threshold is surpassed in the bar number 7, then, the number of classes selected for this experiment is six classes. The respective dendrogram will be cut, thus, when all the set of points are grouped in only six clusters.

$$N_q = q \tag{3.42}$$
$$\text{when } (\text{Between-inertia}_q - \text{Between-inertia}_{q+1}) < \eta_q,$$

where q corresponds to the number of classes, then, $\forall q = 1, 2, \ldots, Q$.

Figure 3.25 presents the set of points classified in six clusters. Note that this ensemble of three figures is the same as above in Figure 3.20 but with the classification. The number of clusters as well as the points in each cluster have been determined by the automatic algorithm of HCA for this example. In Figure 3.25a, we observe in the main axis that four of the six clusters are clearly distinguished (black, yellow, red, and cyan clusters) whereas the other two (magenta and green clusters) seems to be placed in the same position. However, in the other planes (Figures 3.25b and 3.25c), we can see a clear separation between these two clusters.

Once the classification of the set of points is done, we return to the original current measurements of the experiment and we classify each sample belonging to an individual with its cluster. Figure 3.26a represents the different classes in current measurements and shows the accurate results obtained for this experiment. We observe the clear classification of the individuals according to the duration of the individual, the microcontroller current (solid blue plot) and the RF module current (solid red plot). Then, we can notice in Figure 3.26a that the yellow cluster corresponds to the sleep mode in microcontroller and in RF module; the cyan cluster is done when the microcontroller wakes up to take a value from the sensor; in green class, the RF module wakes up to establish a serial communication with the microcontroller; after that, black cluster represents the transmissions of the data packet and the reception of the ACK frame as well as the go-to-sleep transition in the transceiver (different states because they are similar current and time profiles in the electronic components under evaluation); in pink class, the RF module carries out the reception of packets; and, finally,



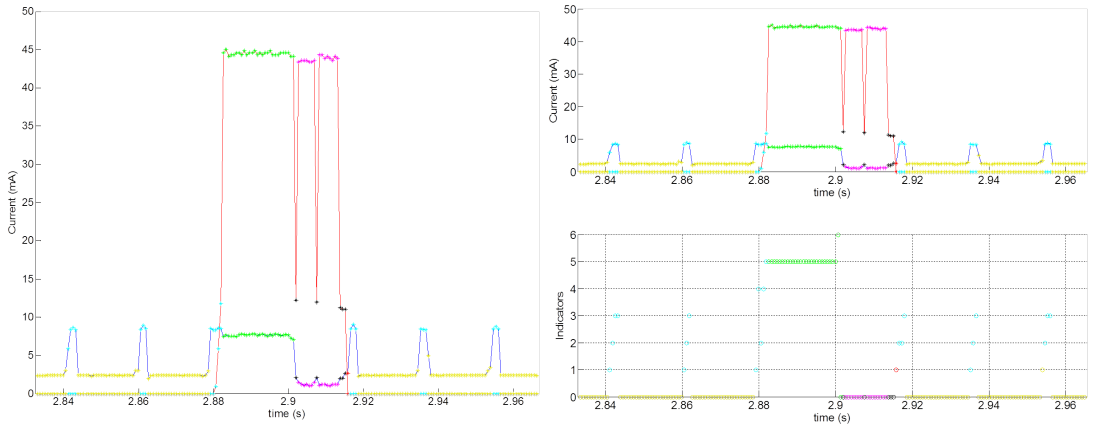
(A) Main axis: Dimensions 1 and 2

(B) Dimensions 1 and 3

(C) Dimensions 2 and 3

FIGURE 3.25: HCA. Six different clusters in this experiment.

red cluster appears as only a sample just after the black individual of the go-to-sleep transition in the transceiver. Figure 3.26b contains the same current graphs as in Figure 3.26a but with the corresponding hardware indicators. We remind that these indicators were introduced as macros in the embedded software in order to identify the actions of the node with the energy measurements. The node under evaluation and the measurements platform are linked by four digital input/output pins where the indicators are transferred from the microcontroller in the node to the microcontroller in the measurements platform. In Figure 3.26b, the indicator 0 introduced for the sleep mode coincides clearly with the yellow individuals; also, we observe that the indicator 5 appears with the green individual which describes the active mode in RF module when the serial communication between microcontroller and RF module is carried out. Indicators 1, 2 and 3 are introduced when the microcontroller wakes up, when microcontroller takes a value from the sensor and when this is value is stored in RAM, respectively. We observe that these three actions are grouped in a single cluster (cyan cluster) because the automatic model cannot distinguish the tiny differences in current between them. As future work, the energy platform will be able to identify these small changes of current and the automatic energy model could analysis these new states. Moreover, the black and the pink classes correspond to the indicator 0 in these results because the XBee



(A) Classification of the currents

(B) Classification of the currents and the hardware indicators

FIGURE 3.26: Classification of the individuals in the currents.

transceiver continues to operate even if the microcontroller is in sleep mode. Then, the RF module sends the packet which the microcontroller gave to it and, after a good reception of the ACK frame from the receiver node, the RF module returns to the sleep mode. With these results, we can verify that the current required by each electronic component as well as the states identified by the automatic energy consumption model are linked to the embedded code in microcontroller.

The main goal of the next Section is to obtain statistical data and the transitions between classes from the analysis. This information forms the Markov chain to, later, estimate the node lifetime.

3.3.4 Markov Model

Firstly, this Section describes the statistical data calculated from every class. The first values to obtain from each class are the mean current values for each electronic component used in the experiment. This current values are placed in a table as represented in Table 3.9 where each row defines an electronic component of the node and each column corresponds to each class. We represent the maximum number of electronic components as $p-1$ because in the matrix of the original data, in (3.15), the maximum number of columns is p. We do not take into account the first column because, in the first column, we include the time values. The rest of the columns is completed with the mean current values of every period for every electronic component, as explained in Table 3.7. Then, the total number of components of the node evaluated in the experiment is $p-1$ energy currents belonging to the $p-1$ electronic components evaluated in the experiment.

The second parameter to obtain is the mean time value per class. These time values are calculated from the first column of the data matrix in (3.15). Other parameters of the currents, as the variance, could be evaluated if these parameters were important to characterize the clusters of points.

Then, we calculate the mean energy consumption per class from the values under evaluation: the mean current per class and per electronic component and the mean time per class. Moreover, we have to know the voltage value on the circuit to complete the calculation of the energy consumption in the experiment.

TABLE 3.9: Mean current values per electronic component and per class.

	C_1	C_2	...	C_q	...	C_Q
1	$\bar{I}_1^1$	$\bar{I}_2^1$	...	$\bar{I}_q^1$	...	$\bar{I}_Q^1$
2	$\bar{I}_1^2$	$\bar{I}_2^2$	...	$\bar{I}_q^2$	...	$\bar{I}_Q^2$
...	...	...	...	...	...	...
j	$\bar{I}_1^j$	$\bar{I}_2^j$	...	$\bar{I}_q^j$	...	$\bar{I}_Q^j$
...	...	...	...	...	...	...
$p-1$	$\bar{I}_1^{p-1}$	$\bar{I}_2^{p-1}$	...	$\bar{I}_q^{p-1}$	...	$\bar{I}_Q^{p-1}$

Other important statistical data is the number of repetitions of every class in the experiment. This parameter is represented as v_q where the sum of all the v_q is the total number of individuals n, as expressed in (3.43). These individuals corresponds to the rows of the original data matrix in (3.15). In addition, the weight of each class w_q in the test is calculated in (3.44).

$$\sum_{q=1}^{Q} v_q = n \tag{3.43}$$

$$\frac{v_q}{n} = w_q \tag{3.44}$$

Moreover, the transitions between classes constitute a key parameter to characterize the behavior of the system. These changes of classes are represented in a matrix called *transition matrix* (3.45) where the element of the rows corresponds to the class where the process comes from whereas the element of the columns represents the class where the process goes to. The elements in this matrix represent the probability $p_{i,j}$ to change from a class i to another class j in a scenario with the same characteristics as the scenario studied in the experiment. Then, a property of this transition matrix is that the sum of the elements in a same row must result 1, as explained in (3.46).

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} & \cdots & p_{1,Q} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} & \cdots & p_{2,Q} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} & \cdots & p_{i,Q} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{Q,1} & p_{Q,2} & \cdots & p_{Q,j} & \cdots & p_{Q,Q} \end{bmatrix} \tag{3.45}$$

$$\sum_{j=1}^{Q} p_{i,j} = 1 \qquad \forall i = 1,2,\ldots,Q \tag{3.46}$$

The transition matrix is used as a matrix of probabilities to predict the classes that could appear in the future. To accomplish this task, we consider the system as a Markov model where the classes obtained in the HCA correspond to the states of the Markov chain and the transitions between classes become the probabilities to change of state. A graphical example of Markov model obtained from the same experiment treated in this Section is represented in Figure 3.27. In this example, the algorithm has identified six classes or defined as states in Markov models. The transition probabilities between classes are represented by the arrows which link the states with each other according to the probability values in the transition matrix.

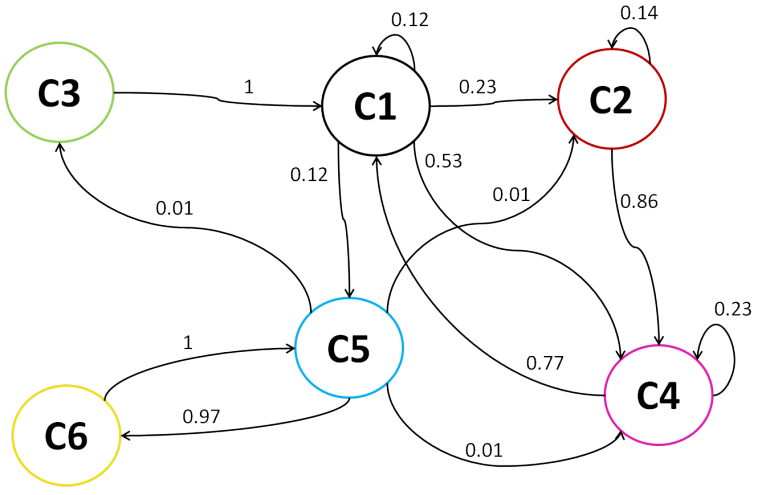


FIGURE 3.27: Markov model in this experiment.

On the basis of these results: average current per class and per electronic component, average time per class, frequency to stay in a class and transition matrix between classes; we are able to calculate the energy consumption by the node in the future if the same scenario remains for the same node of WSN. Also, the sequence of the states for a time of the experiment

can be predicted. With these parameters, an accurate estimation of the lifetime of the node is made from the results of real experiments using a real node of WSN.

3.3.5 Estimation of Node Lifetime

The estimation of node lifetime is generated from the Markov chain after knowing the average current and time values per state. Moreover, the voltage used in the circuit has been introduced in the algorithm to calculate the energy consumed per state.

Firstly, the mean energy consumed per state ($\bar{E}_q$) is calculated with the multiplication of mean current ($\bar{I}_q$), voltage (V) and mean time ($\bar{t}_q$) per state, as in (3.47). The voltage is fixed for all the states because it corresponds to the voltage used in the circuit. Moreover, the total energy consumed by the node is calculated from the mean energy value per state and the number of repetitions of each state (v_q) in the experiment, as represented in (3.48).

$$\bar{E}_q = \bar{I}_q V \bar{t}_q, \qquad \forall q = 1,2,\ldots,Q \tag{3.47}$$

$$E_{\text{TOTAL}} = \sum_{q=1}^{Q} \bar{E}_q v_q \tag{3.48}$$

To obtain the lifetime of the node, we consider that the node is supplied by the 850-mAh TCL PL-383562 polymer Li-ion battery studied in [133] where the discharge function has been given in (2.4) and shown in Figure 2.8. Three methods have been employed to estimate the node lifetime.

The first method for the node lifetime consists in starting the process in a determined class and in changing the class by considering the probability values of the transition matrix. The change of class is determined by random values which are generated in the algorithm. Then, we calculate the accumulation probability value from every row of the transition matrix defined as $a_{i,j}$ and expressed in (3.49). In the iteration k, the process finds the class i ($q_k = i$) and a random number r_k is extracted. Then, the next class q_{k+1} that will appear in iteration $k+1$ is determined by the value of $a_{i,j}$ immediately greater than r_k, as shown in Figure 3.28. Thus, if $a_{i,j-1} < r_k \leq a_{i,j}$, the next class will be $q_{k+1} = j$ and the corresponding values of energy and time in this class will be $E_j(k)$ and $t_j(k)$. In this way, a new plot of currents is created from the transitions and the mean current values. With this new plot, a prediction of

the future current values is entirely possible.

$$\begin{cases} a_{i,0} &= 0 \\ a_{i,1} &= a_{i,0} + p_{i,1} &= p_{i,1} \\ a_{i,2} &= a_{i,1} + p_{i,2} \\ \cdots \\ a_{i,Q} &= a_{i,Q-1} + p_{i,Q} &= \sum_{j=1}^{Q} p_{i,j} = 1 \end{cases} \tag{3.49}$$

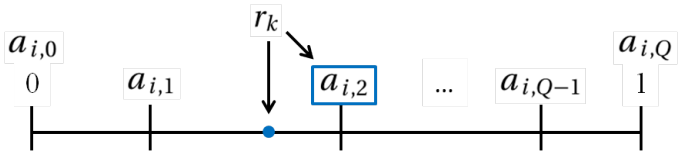


FIGURE 3.28: Method of random values to predict the future states.

The energy consumed by the new current values is calculated and compared with the initial energy stored in the battery. The State-of-Charge (SOC) of the battery for each iteration k is calculated in (3.50), where E_{init} corresponds to the initial energy stored in battery and $E_{\text{rem}}(k)$ is the energy remaining after k iterations of the process. This energy remaining corresponds to the subtraction of the initial energy and the energy consumed in the circuit $E_{\text{cons}}(k)$. This energy consumption corresponds to the energy accumulated between an initial instant $k=0$ and a number of iterations K.

$$\text{SOC}(k) = \frac{E_{\text{rem}}(k)}{E_{\text{init}}} = \frac{E_{\text{init}} - E_{\text{cons}}(k)}{E_{\text{init}}} = 1 - \frac{\sum_{k=0}^{K} E_q(k)}{E_{\text{init}}} \tag{3.50}$$

In a real situation, the voltage offered by the battery is decisive to a proper functioning of the node because the electronic components of the circuit stop working when the voltage becomes less than a determined voltage value. This minimal voltage value (V_{min}) depends on the characteristics of each component. This is why, for each iteration k, the value of SOC is calculated and, then, the new value of V_{OC} in battery is also obtained from (2.4). We compare each new value of V_{OC} with V_{min} and, then, the algorithm decides if the process continues or, otherwise, the battery is not able to supply the node because of the too low voltage in its terminals. This process is represented in the flow chart of Figure 3.29.

The estimation of lifetime battery of the node in this method is done by adding the average time of each class $t_q(k)$ for each iteration in the process. If K_{end} represents the total number of iterations until the final state, *i.e.* when $V_{\text{OC}}(k) < V_{\text{min}}$, then, the lifetime of the node will correspond to the sum of the average time for every class that has appeared in the random values process, as expressed in (3.51).

$$t_{\text{lifetime}} = \sum_{k=0}^{K_{\text{end}}} t_q(k), \qquad \forall q = 1,2,\ldots,Q \tag{3.51}$$

The second method to calculate the lifetime node from the values obtained in the automatic energy model consists to use the theory of the Markov chain.

According to the Markov chain theory, the probability to arrive to the different states after the first iteration Π_1 is determined by (3.52), where Π_0 is the vector of the probability of the initial states and P corresponds to the original transition matrix. In (3.53), the second iteration of the process is shown, where Π_2 represents the probability to arrive to the different states after two iterations in the Markov chain. In this expression, we observe that the vector of probabilities for all the iterations depends only on the initial probabilities vector and on the transition matrix. The transition matrix is multiplicated by itself the number of iterations. Then, the general expression to calculate the probability of reaching a state after k iterations is represented in (3.54), which shows that it depends only on the transition matrix and on the vector of probability of the initial states.

$$\Pi_1 = \Pi_0 P \tag{3.52}$$

$$\Pi_2 = \Pi_1 P = \Pi_0 P \cdot P = \Pi_0 P^2 \tag{3.53}$$

$$\Pi_k = \Pi_0 P^k \tag{3.54}$$

The process to estimate the lifetime of the battery begins with the calculation of the probability to reach each state, iteration by iteration. After that, these probabilities are multiplied

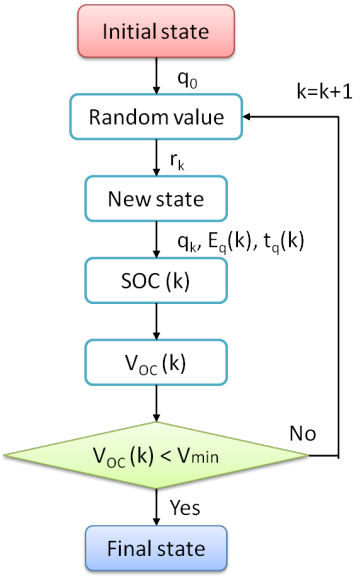


FIGURE 3.29: Flow chart of the random values process method.

to the average energy consumption values per state. The sum of these values of energy corresponds to the energy consumed by the system in the current iteration k. The expression of this energy consumption in an iteration k is represented in (3.55) where Π_k^q deals with the probability to be in the state q in the iteration k and E_q corresponds to the energy consumed by the state q. Then, the energy consumed ($E_{\text{cons}}(k)$) from the beginning of the process until the iteration K is calculated as the sum of the energies calculated in every iteration (3.56).

$$E(k) = \sum_{q=1}^{Q} \Pi_k^q E_q \tag{3.55}$$

$$E_{\text{cons}}(k) = \sum_{k=1}^{K} E(k) \tag{3.56}$$

Now, the lifetime of the battery is calculated similarly as the energy consumption. The time spent per iteration is calculated in (3.57) with Π_k^q and t_q, where t_q is the time corresponding to the state q. Then, the time spent in the iteration k is calculated as the sum of the products of the probabilities per state and the time of each state. The expression of the node lifetime is shown in (3.58), where K_{end} represents the iteration when the process reaches the final state, *i.e.* when the voltage in the battery (V_{OC}) is less than the minimum voltage allowed to make the node work (V_{min}). The flow chart of the process of this method is now shown in Figure 3.30.

$$t(k) = \sum_{q=1}^{Q} \Pi_k^q t_q \tag{3.57}$$

$$t_{\text{lifetime}}(k) = \sum_{k=1}^{K_{\text{end}}} t(k), \qquad \text{when} \quad V_{\text{OC}}(k) < V_{\text{min}} \tag{3.58}$$

The algorithms for both methods presented above, *random values method* and *probability states method*, to estimate the lifetime node from the experimental values have been created and tested. We have found the same problem in both: the energy consumed by each state is too small according to the capacity of the battery. The energy consumed in a state is around the millijoules or tens of millijoules, whereas the energy stored in the battery reaches the 850 mAh, what is higher than 10 000 joules. The difference of quantity of energy between a state and the battery is huge, then, the value of total iterations $k = K_{\text{end}}$ to arrive to the final state, when $V_{\text{OC}}(k) < V_{\text{min}}$, will be huge. This is a problem according to the large number of resources used to execute this process, as the time and the capacity of calculation invested. Then, to solve this problem, a simplification of the process is made by considering that the impact of this simplification must be minimum in the final results. This simplification deals with finding the stationary matrix ($P(\infty)$) to complete the calculations.

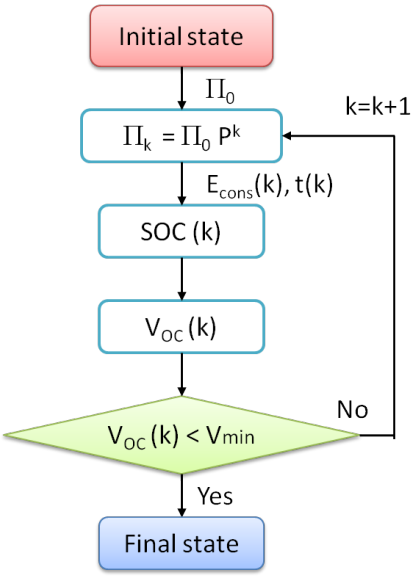


FIGURE 3.30: Flow chart of the probability states process.

In the first method, the random values determine the new states and, for each k, it appears a new transition between the new states. Then, a new transition matrix (P_1) is created from these new transitions. Both, the original transition matrix and the new transition matrix created from the random values, are compared for each iteration k. We consider that the new transition matrix has arrived to the stationary state when the maximum value of the subtract of the elements of both transition matrix is less than a determined value ΔP_1, as represented in (3.59).

$$P_1(k) \equiv P_1(\infty), \qquad \text{when } \max|P_1(k) - P(k)| < \Delta P_1 \tag{3.59}$$

The discussion to choose ΔP_1 comprises two different elements: the accuracy to get close both transition matrices, the original matrix and the matrix created by the random process method; and the number of iterations to reach the difference of ΔP_1, then, the time of the algorithm in execution. If we consider these two elements for all the tests that have been done, we decided $\Delta P_1 = 10^{-3}$ as optimal threshold. The time of execution for this value of ΔP_1 tends to vary in a range from 3 seconds to 2 minutes. These changes of time of execution are due to the random character of the values used to create the new transition matrix. By depending on these values, the process will get sooner or later to reach the stationary matrix.

In the second method, the process to have the stationary matrix is slightly different. In this case, we consider that the transition matrix has reached the stationary state ($P_2(\infty)$) when the maximum difference in elements between the transition matrix in the iteration k and the same matrix in the previous iteration $(k-1)$ is less than a threshold represented as ΔP_2. This expression is presented in (3.60).

$$P_2(k) = P^k \equiv P_2(\infty), \qquad \text{when } \max\left|P^k - P^{k-1}\right| < \Delta P_2 \tag{3.60}$$

In this case, we calculate the energy consumption for each iteration and we compare the accumulated energy consumed until this iteration with the energy stored in battery. This process is executed until the stationary matrix is found. Then, the energy consumed until the end of the battery is easily obtained from the energy per state and from the stationary matrix. This technique that uses the stationary matrix to calculate the energy consumption accelerates significantly the process of estimation of the battery lifetime. In this case, the ΔP_2 to determine the accuracy of the method has been chosen in $\Delta P_2 = 10^{-12}$. The time of the execution of this process with this threshold is usually less than 4 seconds. On this occasion, the time of execution for the same experiment is always the same because it does not depend on random values, the calculation becomes unchangeable every time.

Then, after some evaluation tests with the previous methods, a new method to estimate the lifetime of a node is presented. This third method is based on the stationary transition matrix which is calculated in this case from the theoretical form expressed in (3.61). The second method explained previously that the stationary transition matrix provides the information necessary to find out the stationary probability per state vector $\Pi(\infty)$ (3.62). In this expression, the probability vector of the initial states Π_0 is not needed when we work with the stationary matrix. Once the stationary probability per state vector is obtained, the energy consumed and the time spent in the stationary state is calculated as in (3.63) and (3.64).

$$\lim_{k\to\infty} P^k = P(\infty) \tag{3.61}$$

$$\Pi(\infty) = \Pi_0 P(\infty) \simeq \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} P(\infty) \tag{3.62}$$

$$E(\infty) = \Pi(\infty) E_q \tag{3.63}$$

$$t(\infty) = \Pi(\infty) t_q \tag{3.64}$$

The process executed by the algorithm to use this third method is shown in the flow chart of Figure 3.31. We increment each time the number of k in an order of 10 to arrive to the stationary matrix with the minimum delay. The exponent of 10 of the number k is called k_{index}. We consider that the transition matrix has arrived to its stationary state when the element with the maximum value corresponding to the difference between two consecutive transition matrices (P^{k-1} and P^k) is less than a threshold ΔP_3, as in (3.65). This case follows the same accuracy as in the second method $\Delta P_3 = 10^{-12}$ because this value is enough to the expected precision. The difference between this method and the previous one is the execution time, now the process spends several tens of milliseconds to estimate the lifetime

of the battery with the same accuracy. Other difference between both methods is found in the first iterations of the second method. The values of these first iterations till arriving to the stationary matrix are discarded because the results show that these values are negligible regarding the total values considered until the battery is empty.

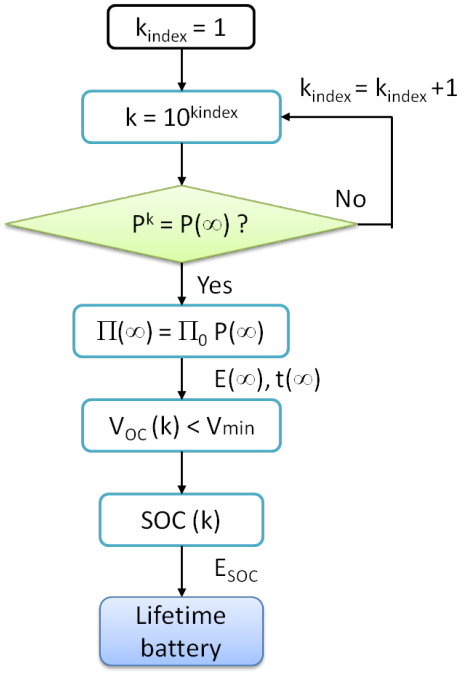


FIGURE 3.31: Flow chart of the stationary transition matrix process.

$$P_3(k) = P^k \equiv P_3(\infty), \qquad \text{when } \max\left|P^k - P^{k-1}\right| < \Delta P_3 \tag{3.65}$$

Once the energy consumption in the stationary state is obtained, we calculate the number of times (υ) that this stationary state can be used until the battery is exhausted. As explained in the first and in the second method, the node does not use all the energy in battery but the energy offered by the battery when the voltage in its terminals (V_{OC}) is higher than a minimum value (V_{min}). Then, in this case, the process is the opposite of the process in previous methods. Here, the SOC is calculated from the expression of V_{OC} when $V_{OC} < V_{min}$. The energy that the node can use is calculated from this value of SOC (E_{SOC}) and, then, we obtain υ as represented in the expression (3.66). Finally, the lifetime of the node is easily calculated in (3.67) by multiplying the time spent in the stationary state and the maximum number of times that this stationary state is used for these conditions.

$$\upsilon = \frac{E_{SOC}}{E(\infty)} \tag{3.66}$$

$$t_{lifetime} = t(\infty) \cdot \upsilon \tag{3.67}$$

We have presented three methods to estimate the lifetime of a node of WSN based on the Markov chains. With the first method, we use random values and the precision of the prediction is not very high. However, this first method is able to represent on a plot of currents a prediction of the future states that every hardware component in the node may take in the future. The second method is the most accurate of the three methods because the threshold to the precision is much smaller than in first method. Furthermore, the first iterations until obtaining the stationary transition matrix are taken into account. However, the third method discards the first states until reaching the stationary matrix but it calculates immediately the stationary transition matrix and estimates the lifetime node with the same precision as the second method and, moreover, faster. Finally, the experiments show that the results of the second and the third methods are basically the same. The development of the three methods reinforces the algorithm of the estimation of a realistic and reliable node lifetime.

In Section 3.4, we present the results of an experiment with a real node of WSN. Moreover, we also use the automatic energy consumption model to modify the hardware and the software in the node and verify the optimization of the energy consumption.

3.4 Experimental Results

Several experiments have been studied to probe the utility of the automatic energy consumption model. The automatic character of the model allows to compare easily and rapidly different tests.

Three experiments have been analyzed with our energy consumption model. In order to achieve clear results and conclusions, the changes between the three experiments are minimized. Then, for the three experiments, we have used the Arduino Nano v3.0 with ATmega328p microcontroller [125], the XBee Series 1 as transmitting unit and the ADXL345 accelerometer.

The experiments follow an embedded software where the microcontroller wakes-up every period of Watchdog Timer (WDT), t_{WDT} milliseconds, and, then, it takes a value from the accelerometer. Each value composed by 1 byte is stored in RAM of the microcontroller. After a determined amount of data N_{data} is stored in RAM, the transceiver is awaken by the microcontroller and all this information is transmitted via the transceiver. The destination node is the sink of the network which is situated at a distance of 3 meters from the single node. This process is expressed graphically in Figure 3.32.

In this case, the value of N_{data} chosen for the three experiments is fixed to 100 data. However, the time value of sleep mode between two active modes in microcontroller t_{WDT} changes for each test.

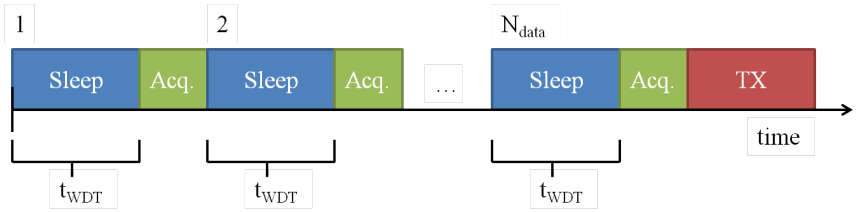


FIGURE 3.32: Process of the embedded software in the node.

For the first experiment, we choose $t_{\text{WDT}} = 16\text{ms}$, that constitutes the fastest WDT with a frequency clock of 16 MHz in ATmega328p microcontroller. The coordinator node of the network will receive N_{data} bytes of information in the TX state after $N_{\text{data}} \times (t_{\text{WDT}} + t_{\text{acq}})$ milliseconds. The time spent in acquisition state (t_{acq}) depends on the number of values taken each time from the sensor n_{sens}. Here, we take only 1 value for each acquisition state and each value is composed by 1 byte, then, $n_{\text{sens}} = 1\text{byte}$.

The first experiment, with $t_{\text{WDT}} = 16\text{ms}$ has been shown in Section 2 as example of energy measurements. This experiment also coincides with the example presented in Section 3.3 to explain the automatic energy consumption model. Then, the automatic segmentation of this experiment is seen in Figures 3.16, 3.17 and 3.18 whereas the PCA is presented in Figure 3.20a. Moreover, the HCA is shown in Figure 3.25a and the different classes are represented on the current values in Figure 3.26a. The statistical data of currents and times for every class obtained from the automatic model are shown in (3.68), where I_C^1 corresponds to mean current values in radio module whereas I_C^2 corresponds to the microcontroller. Moreover, the times n_C and the percentage of times (P_C) that each class appears in the experiment are also expressed in (3.68). as well as the transition matrix of the classes in this experimentation. The probabilities of transitions between states in the Markov model have been graphically represented in Figure 3.27.

	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1 (mA)	11.87	0	44.42	42.26	0.067	0.0002
I_C^2 (mA)	1.87	2.61	7.66	1.05	8.54	2.43
t_C (s)	0.0011	0.0026	0.0194	0.0038	0.0024	0.0166
E_C (mJ)	0.0517	0.0221	3.3282	0.5398	0.0668	0.1333
n_C	17	7	5	13	489	481
P_C (%)	1.68	0.69	0.49	1.28	48.32	47.53

(3.68)

The automatic model has identified six different classes in this experiment by taking into account the three parameters which characterize the individuals: current on the radio module,

current on the microcontroller and time for each individual.

From (3.68), we can identify the differences of each class. The first class (C_1) corresponds to the situation when the microcontroller (component I_C^2) is in sleep mode (near 2 mA) whereas the radio module transmits a packet and disconnects the receiver part, then, the current level falls from 44 mA to about 12 mA. We can observe in t_C values that the duration of these situations is very short ($t_C = 1.1$ ms for C_1), then, the energy consumed in this class is about $E_C = 51$ μJ. Moreover, C_1 appears $n_C = 17$ times in the experiment, representing the P_C =1.68% of all the classes. Regarding the energy consumption and the percentage of times that this class C_1 appears, we reach the conclusion that C_1 is not important in the energy consumption of the system.

Now, we analyze C_2. According to its current levels in both electronic components, radio module (I_C^1) and microcontroller (I_C^2), this class corresponds to the situation when both components reach their respective sleep modes. The current in radio module can decrease near 0 mA whereas the microcontroller stays around 2 mA. If we watch the mean current values in (3.68), we observe that C_6 includes similar values that C_2. Then, the mean time values (t_C) shows that the usual class corresponding to the sleep mode of the node is C_6 because its time value is 16.6 ms, very close to the WDT used for this experiment ($t_{\text{WDT}} = 16$ ms). If we consider also the amount of times that C_2 and C_6 appear, there is also a big difference: while C_6 is present 47.53% of the times, C_2 just 0.69%. We conclude by regarding this information that C_2 represents a spurious class, then, this class is negligible for our results.

C_3 and C_4 corresponds to the states when the radio module remains in active mode, more precisely, it reaches the receive mode because the current level in this mode can be up to 45 mA. In this case, the mean current values calculated are 44.42 mA and 42.26 mA, respectively. In the current values of the microcontroller, there is a clear difference. In C_3, the microcontroller is in active mode whereas, in C_4, it is in sleep mode. In time values, we observe also a big difference. This is why C_3 consumes six times more of energy (E_C) than C_4. Besides, C_3 is present only 5 times in the experiment (n_C), which means 0.49%, however, in this case, C_3 cannot be defined as a spurious class due to the important quantity of energy consumed by this class (3.3282 mJ).

Otherwise, C_5 represents the class where the microcontroller is in active mode and the radio module in sleep mode. This state of operation corresponds to the acquisition of the data from the sensor in the node. The mean time of this state calculated by the algorithm corresponds to t_{acq}, then, $t_{\text{acq}} = 0.0024\text{s} = 2.4\text{ms}$. C_5 represents the class that contains the most of repetitions in the experiment, with 489 times, it is the 48.32% of the total.

We can identify in mean current values of (3.68) that the class C_6 corresponds to the sleep mode in both microcontroller and radio module since these current levels are low in this state.

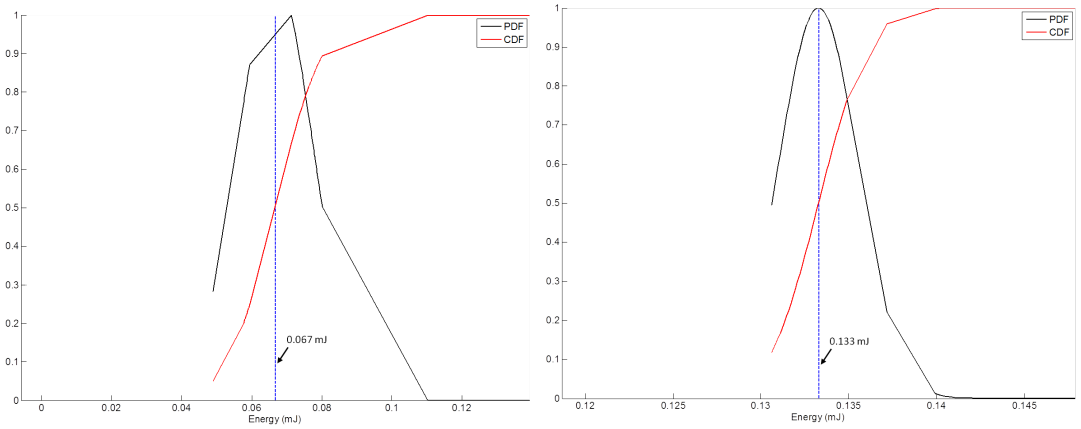
The average energy per class (E_C) is expressed in millijoules in (3.68). This average energy values are calculated from the average energy currents for each component (I_C^1 and I_C^2) and from the average time values (t_C). We use the probability distribution function (PDF) and the cumulative distribution function (CDF) to show the distribution of the individuals of each class according to their energy consumption values. We assume that theses elements follow a normal distribution or Gaussian distribution [186], whose normalized PDF is expressed as in (3.69). The CDF is a function which represents the probability that a variable x obtains a value $\leq K$, as represented in (3.70).

$$P(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.69}$$

where μ represents the mean and σ^2 is the variance of the values in the distribution.

$$D(x) = \int_{-\infty}^{K} P(x)\,\mathrm{d}x \tag{3.70}$$

We have evaluated the PDF and CDF in C_5 and in C_6 of this example. We chose these classes because they contain the largest number of points. In Figure 3.33a, the PDF (in solid black line) of C_5 as well as the CDF (in solid red line) of this class is represented. We have normalized both, the PDF and CDF. The mean energy value for each class is shown with a vertical dashed blue line. The same graphs are depicted in Figure 3.33b for C_6.



(A) PDF and CDF of class 5

(B) PDF and CDF of class 6

FIGURE 3.33: PDF (in black) and CDF (in red) for classes 5 and 6 in this example.

Figure 3.32 shows the *Sleep state* which is equivalent to C_6 created by the Automatic Energy Consumption Model. Also, the *Acquisition state* corresponds clearly to C_5. However, *TX state*

is harder to evaluate because, according to the automatic algorithm, this state is formed by a combination of different classes ($C_3 - C_1 - C_4 - C_1 - C_4 - C_1 - C_2$), as depicted in Figure 3.34.

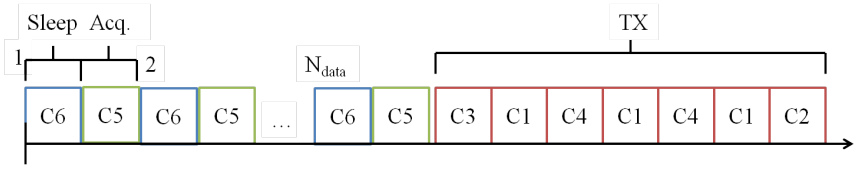


FIGURE 3.34: Process of the embedded software in the node with classes from the automatic energy model.

The Markov model is based on the transition matrix (3.71) which shows the probability to move from a class (in the rows) to another class (in the columns) in the experiment studied. From this transition matrix, we observe that some classes are totally linked. When the process reaches C_2, it will continue certainly in C_5 because the probability $p_{2,5} = 1$. We find the same case for C_3 to C_1 and for C_6 to C_5. After C_5, the process will reach C_6 in a 98.36% of the cases whereas in a 1.02% of the cases, it will pass to C_3. However, the transitions C_5 to C_2 and C_5 to C_4 deal with spurious cases due to the realistic character of the experiments: we find some errors because of the possible noise obtained from the real measurements. The values of probability in the row of C_1 are more diverse because the process leaves this class to reach other different classes as C_4 and C_2, as shown in Figure 3.34. The probability $p_{1,6} = 0$ occurs because, after the first transmission, the process reaches immediately the class C_5 instead of C_6.

$p_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6
C_1	0.1176	0.2353	0	0.5294	0.1176	0
C_2	0	0	0	0	1	0
C_3	1	0	0	0	0	0
C_4	0.7692	0	0	0.2308	0	0
C_5	0	0.0041	0.0102	0.002	0	0.9836
C_6	0	0	0	0	1	0

(3.71)

Once the statistical data in the Markov model for this experiment are presented, the battery lifetime in the node is calculated.

This lifetime is obtained by using the three methods explained in Section 3.3.5. In the first method, the random values chosen to reach the stationary matrix of the transition matrix cause some variations in the calculation of the lifetime battery. These variations are small due to the small tolerance value selected to this situation, $\Delta P_1 = 10^{-3}$. However, the random nature of the values used in this method is more notable regarding the time of execution. Both values, lifetime and time of execution, for 10 calculations are presented in Table 3.10. This table includes the mean and the variance values for each variable, in $\bar{x}$ and σ^2 columns,

respectively. The big differences on the time of calculation for each execution is clearly represented with the variance $\sigma^2 = 2.51 \cdot 10^3$.

TABLE 3.10: Node lifetime with random values method.

Number exec.	1	2	3	4	5	6
Lifetime (days)	8.553	8.5491	8.5501	8.5421	8.5412	8.5474
Time Exec. (s)	3.24	5.49	3.28	22.47	39.29	5.04

Number exec.	7	8	9	10	$\bar{x}$	σ^2
Lifetime (days)	8.5495	8.5567	8.5499	8.526	8.5465	$7.28 \cdot 10^{-5}$
Time Exec. (s)	73.68	15.36	161.74	8.8	33.84	$2.51 \cdot 10^3$

Now, we have estimated the node lifetime with the second method. This value is fixed to 8.5549 days for this experiment. In this case, with an accuracy value of $\Delta P_2 = 10^{-12}$, the time of execution has been only of approximately 3.1 seconds. Furthermore, we also used the third method which consists of calculating quickly the stationary transition matrix to later estimate the lifetime node. Here, we use the same accuracy as in the second method, $\Delta P_3 = 10^{-12}$. The node lifetime coincides with the value obtained in the second method, 8.5549 days, but, in this case, the execution time does not exceed 1 millisecond.

Aside from the calculation of the battery lifetime in the node, we are also interested to find out the causes of this energy consumption and how we could minimize the energy consumed to increase the lifetime of the node. This is why, the information of the percentage of energy consumed per class is important, as well as the percentage of time that the process stays in each class. Thanks to this information, we are able to propose some solutions to increase the battery lifetime in this example.

3.4.1 Increasing Node Lifetime

The percentage of the energy consumed (E_{perc}) the percentage of the time (t_{perc}) spent by each class in this experiment is shown in (3.72). To calculate these values of percentage, we consider the average energy and time (E_C and t_C values in (3.68)) as well as the number of repetitions (n_C) for each class, as represented in (3.73). We can observe that the most of energy in the whole of this experiment has been consumed in the class C_6 because the process stays in this class the most of the time, exactly the 85.71% of the time, as shown in (3.72). This fact is due to the long period of C_6 (16.6 ms whereas the period of the other classes is considerably less) and the number of times that C_6 appears in the experiment (47.53% of times as presented in (3.68)).

(%)	C_1	C_2	C_3	C_4	C_5	C_6
E_{perc}	0.72	0.12	13.7	5.78	26.9	52.78
t_{perc}	0.21	0.19	1.04	0.53	12.33	85.7

(3.72)

$$E^i_{\text{perc}} = \frac{E^i_C \cdot n^i_C}{\sum_{q=1}^{Q=6} E^q_C \cdot n^q_C} \times 100\,(\%), \qquad t^i_{\text{perc}} = \frac{t^i_C \cdot n^i_C}{\sum_{q=1}^{Q=6} t^q_C \cdot n^q_C} \times 100\,(\%) \tag{3.73}$$

where i represents the i^{th} class.

This information help to know which elements of the experiment should been changed to optimize in energy. At first, we test the optimization of energy consumption by varying the current and time values in the classes. The transition matrix remains unchanged.

In this case, we observe that the first class to optimize is C_6. This class represents the *Sleep state* in the node, according to the Figure 3.32. This means that both, the microcontroller and the transceiver, reach the sleep mode. According to their current levels in (3.68), the current level measured for the transceiver in C_6 is quite low ($I^1_C = 0.0002$ mA), but the current obtained for the microcontroller is too high ($I^2_C = 2.43$mA). This high mean value of the current in sleep mode of the microcontroller is due to the current needed by another component placed in the Arduino Nano v3.0 platform. One of these energy-hungry components correspond to the USB to serial UART interface FT232RL [187] used as interface in serial communication between the microcontroller and the computer. This fact responds to the current level of 2.43 mA in sleep mode for the microcontroller. Now, we can change the current value in C_6 to remove the current corresponding to the FT232RL component. Then, the new value I^2_C for C_6 is 0.1 mA.

The new table including the current values for both components is represented in (3.74). Now, we estimate the node lifetime for these new values. This calculation has been realized by using directly the third method of estimation of lifetime described above. This method has been selected in order to achieve a good accuracy with an efficient execution time. Here, the node lifetime increases to 17.32 days, a 102% more than in the precedent test. The new values of percentages in energy per class are also expressed in (3.74) whereas the comparison of these percentages of the energy in precedent test and in current test are depicted in Figure 3.35. We can notice in these bar graphs that the values of energy in each class have completely changed. The percentages of time per class seen in (3.72) keeps unchanged because the time values have not been modified in this test.

(mA)	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1	11.87	0	44.42	42.26	0.067	0.0002
I_C^2	1.87	2.61	7.66	1.05	8.54	0.1
E_{perc} (%)	1.46	0.26	27.73	11.7	54.44	4.41

(3.74)

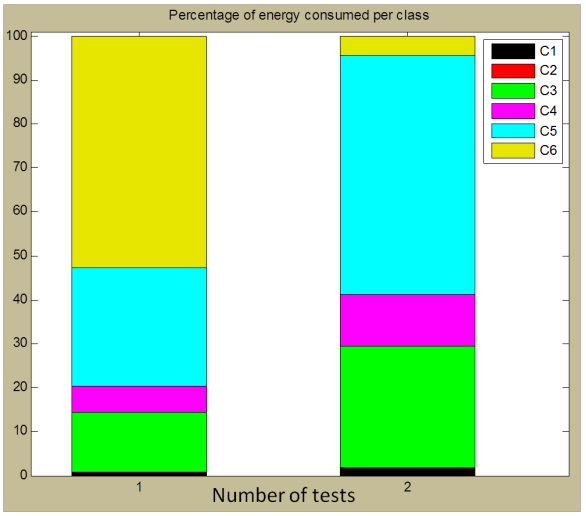


FIGURE 3.35: Stacked bars of the percentage of energy consumed per class in both tests.

In this case, the part of energy consumed by C_6 in relation to the whole energy consumed in the experiment is only 4.41%. Now, the most important class regarding the energy consumption becomes C_5 with 54.44% followed by C_3 with 27.73% of the total energy. The causes of the important weight in this case of C_5 and C_3 are different: C_5 appears many times, 48.32% of times, whereas the repetitions of C_3 are only the 0.5% of times but the energy consumed each time by C_3 is very high (3.3282 mJ).

The next step consists of removing the effect of the FT232RL interface for all the classes. This fact is done just to keep the consistency of the experiment because if we consider that the current used by the FT232RL interface is removed in one class, it should be removed in all the classes. Then, the current in sleep mode on the microcontroller is decreased to 0.1 as in C_6 whereas the current values in the classes where the microcontroller stays active are reduce equally of 2.5 mA. The current values and the new percentages of energy consumption per class after these modifications are shown in (3.75). The node lifetime in this situation reaches 21.07 days, a 21.6% more than in the precedent test.

(mA)	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1	11.87	0	44.42	42.26	0.067	0.0002
I_C^2	0.1	0.1	5.16	0.1	6.04	0.1
E_{perc} (%)	1.55	0.01	32.13	13.92	47.02	5.36

(3.75)

Not only can we modify easily the current values of the microcontroller in the node, but also the current values of the other hardware components evaluated in the experiment. In this new case, we can change the currents in the the radio chip. Note in (3.75) that the mean current values measured by Synergie platform when the radio module (I_C^1) is in active mode (C_3 and C_4) are 44.42 mA and 42.26 mA, respectively. This coincides with the current indicated in XBee Series1 datasheet [127] when the component works in active mode (around 45 mA). This radio module may be replaced by other more energy-efficient transceiver that uses the same standard to communicate (IEEE 802.15.4), for instance the TI CC2420. TI CC2420 [110] is an RF transceiver widely used in WSN and designed to low-power applications. According to the CC2420 datasheet, the current in transmission mode is 17.4 mA and, in reception mode, it reaches the 18.8 mA. CC2420 uses the same wireless technology and realizes the same tasks as XBee, even if we show in Chapter 2 that the current profile in both transceivers is not the same. Then, XBee can be replaced by the CC2420 in the same circumstances. After observing the current plots of the RF module, we conclude that C_3 corresponds to the reception mode and C_4 to the transmit mode. Then, the values of I_C^1 in these two classes are modified by the values corresponding to the CC2420 (17.4 mA in transmit mode and 18.8 mA in receive mode). The modifications in currents and the new percentages of energy consumed by each class are shown in (3.76). In this test, the node lifetime calculated by the algorithm amounts to 28.01 days, then, a 32.9% better than the immediately preceding test.

(mA)	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1	11.87	0	18.8	17.4	0.067	0.0002
I_C^2	0.1	0.1	5.16	0.1	6.04	0.1
E_{perc} (%)	2.06	0.02	20.64	7.64	62.5	7.14

(3.76)

According to (3.76), at this moment, C_5 is the dominant class because it takes the 62.5% of all the energy consumed by the node. At the moment, we have emulated the changes of the hardware in node to minimize the energy consumption, but the changes in software may be also emulated. In the precedent tests, the period to wake the microcontroller up with the purpose of taking a value from the sensor is 16 milliseconds (ms). This parameter is managed by the WDT in the microcontroller. The algorithm has measured exactly 16.6 ms for this

period, as seen in (3.68). This period is the minimum possible by the WDT for an external clock of 16 MHz in this type of microcontrollers. Other periods as 32 ms, 64 ms, 0.125 seconds (s), 0.25 s, 0.5 s, 1 s, 2 s, 4 s, up to 8 s can be configured as indicated in Atmega328p datasheet [125]. Then, we can change the mean time corresponding to C_6 because this class represents the sleep mode of the node. The embedded software remains the same: the microcontroller wakes up every $t_W DT$ seconds to take a value from the sensor, after N_{data} all the values stored in RAM are sent through the RF module. Now, the new value becomes $t_W DT = 1$ s. This new time value is presented in C_6 of (3.77) as well as the new percentages of energy per class.

We observe that the percentages of energy consumed per class are totally different that the precedents. Now, C_6 is the dominant class with the 82.2% of the total energy because, in the 99.72% of the time, the process remains in this class. This is due to two reasons: 1) the period of C_6 is 1 second whereas, in the other classes, this time does not exceed 30 milliseconds all together; and 2) C_6 appears in a large proportion of the times, 47.53% of times as seen previously in (3.68). Thanks to these characteristics, the lifetime of the node reaches the value of 277 days, then, almost 10 times more than the result in the previous test.

(s)	C_1	C_2	C_3	C_4	C_5	C_6
t_C	0.0011	0.0026	0.0194	0.0038	0.0024	1.0
E_{perc} (%)	0.39	0.003	3.96	1.47	11.98	82.2
t_{perc} (%)	0.004	0.0037	0.02	0.01	0.24	99.72

(3.77)

Now, we increase the time of WDT to its maximum possible value, $t_C = 8$ s in C_6 as represented in (3.78). There are many applications in WSN where the sleep time between two active states is equals or greater than 8 seconds. We also obtain the new values of percentages of energy and time per class in (3.78).

(s)	C_1	C_2	C_3	C_4	C_5	C_6
t_C	0.0011	0.0026	0.0194	0.0038	0.0024	8.0
E_{perc} (%)	0.06	0.0005	0.59	0.22	1.77	97.37
t_{perc} (%)	0.0005	0.0005	0.0025	0.0013	0.03	99.97

(3.78)

In this case, the lifetime of the node increases up to 328 days. Both, E_{perc} and t_{perc} are excessively bigger in C_6 compared to the rest of the classes. Then, with this difference, the possible modifications of current levels in other classes than C_6 will not affect much the values of E_{perc}, t_{perc} and, consequently, the node lifetime. We probe this theory with a new test. The

current values of the RF module (I_C^1) go back to the original values, with XBee modules. Then, the new current values and the percentages of energy per class are shown in (3.79).

	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1 (mA)	11.87	0	44.42	42.26	0.067	0.0002
I_C^2 (mA)	0.1	0.1	5.16	0.1	6.04	0.1
E_{perc} (%)	0.06	0.0004	1.2	0.52	1.76	96.46

(3.79)

We can observe that these percentages of energy have not changed significantly. The node lifetime decreased only up to 325 days. The effect to change the RF module did not almost modify the lifetime of the node due to the small duty cycle (DTC) 3.80. With a DTC in this test of 0.03% (DTC $= 100\% - t_{\text{perc}}^{C6} = 100\% - 99.97\% = 0.03\%$), the difference between the current values in sleep mode and in active mode should be also near 0.03% to make the active mode important in the energy consumption. We consider active mode all the classes that are different to the sleep mode (C_6).

$$\text{DTC}\,(\%) = \frac{T_{\text{active}}}{T_{\text{total}}} = \frac{T_{\text{active}}}{T_{\text{active}} + T_{\text{sleep}}} \times 100(\%) \tag{3.80}$$

With these values of DTC (0.03%), the only predominant class is C_6 which represents the sleep mode in both components, microcontroller and RF module. If the objective deals with extending the node lifetime, we should optimize the energy consumed in this class to note an improvement. Then, the microcontroller could work with a lower current level as its datasheet indicates. Then, we can test with a current of 0.01 mA in sleep mode for the microcontroller. The new results are shown in (3.81). The lifetime in this test differs totally from the precedent calculations. It reaches the 6.65 years because the current of the dominant class has been divided by 10.

	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1 (mA)	11.87	0	44.42	42.26	0.067	0.0002
I_C^2 (mA)	0.1	0.1	5.16	0.1	6.04	0.01
E_{perc} (%)	0.43	0.0034	8.97	3.89	13.12	73.58

(3.81)

According to the ATmega328p datasheet [125], the current value in Power-down Mode can decrease up to 0.1 μA when it works at 1MHz of frequency clock. In this test, the frequency clock is 16 MHz, then, we use a current near to 2 μA. The values of this new test are shown

in (3.82). The lifetime increases in this case up to 15.66 years, then, 9 years more than in the previous test.

	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1 (mA)	11.87	0	44.42	42.26	0.067	0.0002
I_C^2 (mA)	0.1	0.1	5.16	0.1	6.04	0.002
E_{perc} (%)	1.02	0.008	21.12	9.15	30.9	37.8

(3.82)

After these variations of average current, C_6 becomes not so dominant than before. Even if the DTC is the same (0.03%), the values of percentage of energy consumed change because of the low value of current in C_6. At this moment, C_3 and C_4 are quite important in this experiment with the 21.12% and 9.15%, respectively, of energy consumed by this two classes. We consider again that the RF module used is the CC2420 transceiver and we observe the changes. In this case, the changes of values in I_C^1 of C_3 and C_4, as shown in (3.83) and the lifetime of the node increases of near of 20% to reach the 18.7 years of lifetime.

	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1 (mA)	11.87	0	18.8	17.4	0.067	0.0002
I_C^2 (mA)	0.1	0.1	5.16	0.1	6.04	0.002
E_{perc} (%)	1.22	0.01	12.19	4.51	36.91	45.16

(3.83)

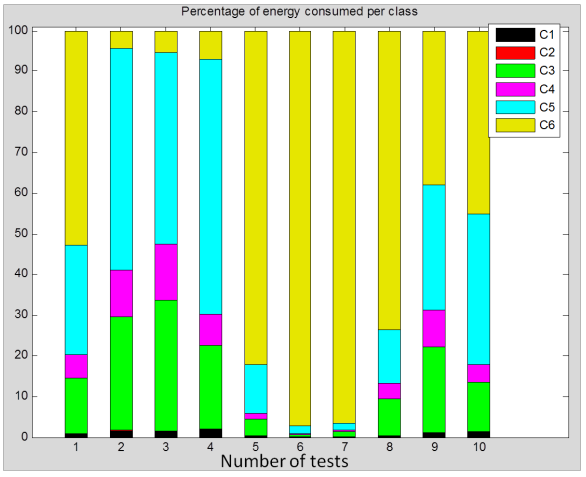


FIGURE 3.36: Stacked bars of the percentage of energy consumed per class in ten tests.

In Figure 3.36, we see the progression of the percentages of energy in each class for these ten tests. We can observe the different variations of these percentages just by changing the

current levels of the classes in the components as well as their times. Also, it is important to notice the increase of the node lifetime from 8.55 days at the beginning to until 18.7 years.

3.4.2 Model Results vs. Real Experiments

In this Section, a real experiment based on the modifications of WDT period has been realized. Here, we want to probe two things: the reliability of the automatic energy model by comparing the obtained results with real experiments and the use of this automatic model to achieve an energy optimization.

Then, the first test evaluated in Section 3.4.1 is compared with the same test but with different WDT values. First test has the original current and time values shown in (3.68). The percentages of energy and time are expressed in (3.72) whereas the lifetime of the node was near 8.55 days.

Then, we modify the time value of C_6 in this test to calculate the lifetime. This time t_C of C_6 is $t_C = 1$ s, then, the time values of all the classes are presented in (3.84) as well as their energy and time percentages for this test. The estimation of lifetime increases up to 13.73 days.

(s)	C_1	C_2	C_3	C_4	C_5	C_6
t_C	0.0011	0.0026	0.0194	0.0038	0.0024	1
E_{perc} (%)	0.004	0.0037	0.02	0.01	0.24	99.72
t_{perc} (%)	0.0224	0.004	0.4253	0.1793	0.8349	98.53

(3.84)

Now, we compare these results with the results obtained by the algorithm when we change the duration of WDT in the real circuit. To change the WDT in microcontroller, we configures the bits WDP3 - WDP0 in the WDTCSR register of ATmega328p microcontroller. For that, we just modify a line of code in the embedded software, as presented in Figure 3.37. For this example, we change the code to have $t_{\text{WDT}} = 1$ s, *i.e.* WDP3 = 0, WDP2 = 1, WDP1 = 1 and WDP0 = 0.

```
WDTCSR = 0<<WDP3 | 0<<WDP2 | 0<<WDP1 | 0<<WDP0; /* 16 milliseconds */
WDTCSR = 0<<WDP3 | 1<<WDP2 | 1<<WDP1 | 0<<WDP0; /* 1 second */
```

FIGURE 3.37: Line of C code in microcontroller to change the t_{WDT}.

The new measurements of current from the real circuit are obtained with the measurements platform of Synergie. Then, the algorithm of the automatic model of energy consumption evaluates these measurements to obtain the new Markov model. The values of average current per class (I_C^1 and I_C^2), average time per class (t_C) and average energy consumed per class

(E_C) are presented in (3.85). Also, we have the number of times (n_C) and the probability (P_C) to reach each class as well as the matrix probability of the new experiment, in (3.86). It is important to notice that the order of the class in these matrices is assigned randomly. Then, from the current values, we observe that C_6 conserves the identification C_6 in the new values of these experiments, as well as C_5. However, previous C_4 corresponded to the state where RF module listens the channel (RX mode) and microcontroller in sleep mode, now, this class is represented in C_3. The same case for C_3 which in this new experiment becomes C_4. Moreover, C_1 in first experiment corresponds to C_2 whereas the old spurious class C_2 disappears and a new spurious class appears in C_1.

(mA)	C_1	C_2	C_3	C_4	C_5	C_6
I_C^1	12.25	11.69	42.65	39.24	0.06	0.0006
I_C^2	8.09	2.17	1.02	7.19	8.66	2.40
t_C (s)	0.0004	0.0012	0.0018	0.0118	0.0022	1.04
E_C (mJ)	0.0277	0.0529	0.2603	1.8032	0.0620	8.2656
n_C	4	10	16	6	398	397
P_C (%)	0.48	1.2	1.94	0.72	47.89	47.77

(3.85)

$p_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6
C_1	0	0.25	0.75	0	0	0
C_2	0	0.1	0.4	0.1	0	0.4
C_3	0.125	0.4375	0.1875	0.25	0	0
C_4	0.3333	0.1667	0.5	0	0	0
C_5	0	0	0.0075	0.0025	0.005	0.9849
C_6	0	0	0	0	1	0

(3.86)

We observe in (3.85) that the value of the WDT, $t_{\text{WDT}} = 1$ s is correctly found by the algorithm with the $t_C = 1.04$ s in C_6. Also, the probabilities in (3.85) and in the transition matrix (3.86) are very similar to the equivalent values in the first experiment. The lifetime in this experiment is 13.89 days. We observe that this value is similar to the 13.73 days of the first experiment when we just modified the value of the time in C_6 in the algorithm.

Three repetitions of the same test of energy measurements have been carried out to probe the accuracy of the automatic model. They are three real tests with the same hardware in the node and the same embedded software. We have measured the current values and calculate the node lifetime in the tests for all the possible t_{WDT}. The Table 3.11 offers the results of

lifetime and Figure 3.38 shows the comparison of the automatic energy model results (in blue) with the three experimental tests. Also Figure 3.39a represents the error rate in days between the results in model and each real experiment whereas Figure 3.39b shows this error rate in percentage of error. We observe that Test 1 and Test 2 are very close to the estimation made by the model with 1.6% and 1.53%, respectively, as median of the values. In these two tests, we can find some peaks of error rates of more than 5%, mainly, for low t_{WDT} values, where DTC is higher. Test 3 is a little different.

TABLE 3.11: Comparison node lifetime in model and in real experiments.

WDT (s)	Node Lifetime (days)			
	Model	Test 1	Test 2	Test 3
0.016	8.55	9.02	9.03	9.33
0.032	10.30	10.82	10.83	11.13
0.064	11.76	12.17	12.17	12.45
0.125	12.69	13.00	12.99	13.25
0.25	13.26	13.50	13.49	13.73
0.5	13.57	13.76	13.75	13.98
1	13.73	13.90	13.89	14.11
2	13.81	13.97	13.99	14.18
4	13.85	14.01	14.00	14.21
8	13.87	14.03	14.01	14.23

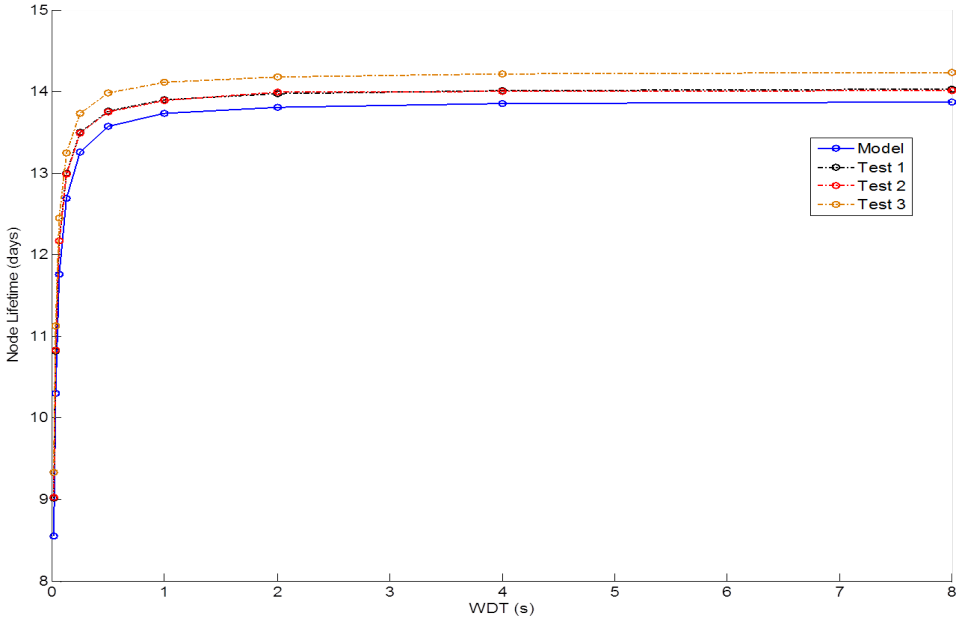
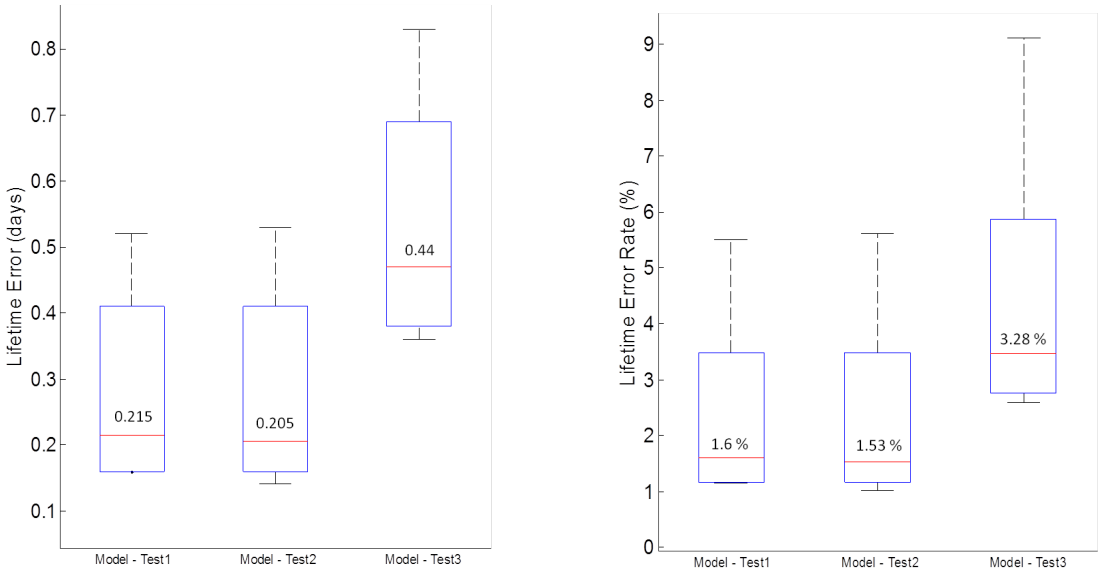


FIGURE 3.38: Comparison of node lifetime between model and real tests.



(A) Error rate in days

(B) Error rate in percentage

FIGURE 3.39: Error rate between model and the three real tests.

We observe that Test 3 has an higher error rate than the other two tests. This is because in the moment of the test, there were some perturbations in the channel as some obstacles or some people in the experimental room. In Test 3, the median of the error rate reaches 3.28% and a peak of 9.12% for the minimum value $t_{\text{WDT}} = 16$ ms.

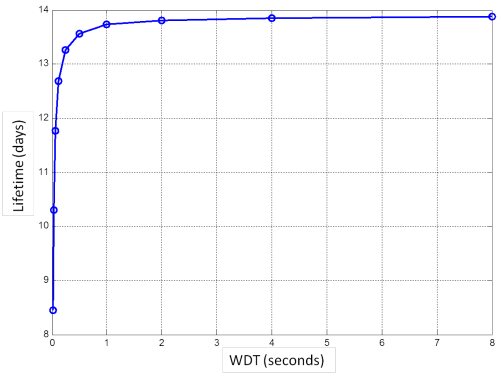
If we study the results obtained by the model, we observe that the lifetime of the node in both cases, when $t_C = 0.0166$ s in sleep state C_6 and when $t_C = 1$ s varies considerably from 8.5549 to 13.73 days, a 60% more. However, the difference of lifetime between both cases, with $t_C = 1$ s in C_6 and with $t_C = 8$ s is not so big, from 13.73 to 13.87 days, only a 1% more. Moreover, if we could increase more and more the time of WDT, the lifetime of the node would converge near 13.9 days. This is due to the relation between the DC and the difference between the current levels in active and in sleep mode. In this experiment, we consider the sleep mode to the class C_6 and the active mode to the other classes, even if each class $C_1 - C_5$ has different current values in both electronic components (microcontroller and RF module).

Figure 3.40a shows the lifetime of the node according to the t_{WDT}. The DTC is totally dependent of the value of t_{WDT}. In this experiment, DTC is expressed as in (3.87), where we take into account the time in active and in sleep mode for all the period of evaluation. The curve in Figure 3.40a has been represented with the t_{WDT} starting in the lowest value (16 ms) and increasing according to the possible values for t_{WDT} (32 ms, 64 ms, 0.125 s, 0.25 s, 0.5 s, 1 s, 2 s, 4 s, 8 s). This curve takes the form of an asymptotic function which converges near 13.9 days of lifetime. In Figure 3.40b, we represents a curve with the relation between DTC and WDT in this test which is inversely proportional as shown in (3.87). We observe that, in this case, the asymptotic curve is inverted and it reaches the zone of saturation in $t_{\text{WDT}} = 1$ s, as

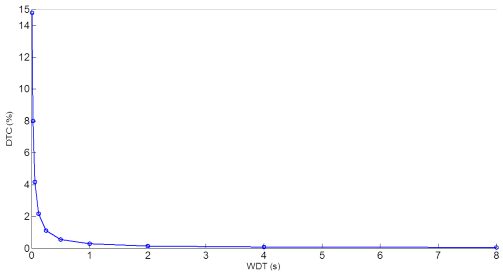
before in Figure 3.40a. Besides, Figure 3.40c depicts the node lifetime according to the DTC where we observe that there are several points very close in the zone near 14 days. These points corresponds to the lowest DTC periods, i.e. the points of highest t_{WDT} (from 1 s until 8s).

$$\text{DTC}\,(\%) = \frac{T_{\text{active}}}{T_{\text{active}} + T_{\text{sleep}}} = \frac{\sum_{q=1}^{Q=5} t_{C_q} \cdot P_{C_q}}{\sum_{q=1}^{Q=5} t_{C_q} \cdot P_{C_q} + t_{C_6} \cdot P_{C_6}} = \frac{\sum_{q=1}^{Q=5} t_{C_q} \cdot P_{C_q}}{\sum_{q=1}^{Q=5} t_{C_q} \cdot P_{C_q} + t_{\text{WDT}} \cdot P_{C_6}} \times 100\% \tag{3.87}$$

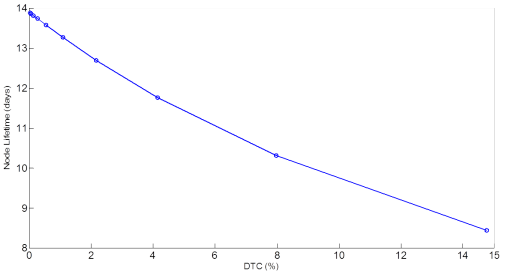
where t_{C_q} corresponds to the average time for each class q, seen in (3.68), and P_{C_q} is the relative frequency of each class.



(A) Lifetime of the node according to the WDT



(B) Duty cycle according to t_{WDT}



(C) Lifetime of the node according to the DTC

FIGURE 3.40: Relation between node lifetime, t_{WDT} and DTC.

In the first part of the curve in Figure 3.40a, the lifetime raises rapidly whereas after $t_{\text{WDT}} = 1$ s, the increase becomes very low. Table 3.12 provides the information about the lifetime node per t_{WDT} value but, also, the difference of two consecutive lifetime values in Figure 3.40a and the percentage of increment between consecutive lifetime values. Table 3.12 shows clearly the evolution of the lifetime, mainly, in the third row, percentage of increment of lifetime. If we change t_{WDT} from 16 ms to 32 ms, the increase of the lifetime is of 22.02% but, after $t_{\text{WDT}} = 1$ s the increment does not exceed the 1% between consecutive t_{WDT} values. Figure 3.41 depicts this decrease of percentage from values in third row of Table 3.12. Then, after $t_{\text{WDT}} = 1$ s, it is not interesting to increment the t_{WDT} because the lifetime does not increase a lot, moreover, the performance of the system becomes worst if the node stays in

TABLE 3.12: Lifetime, difference of lifetime and percentage of increase of lifetime per WDT value.

WDT (s)	0.016	0.032	0.064	0.125	0.25
Lifetime (days)	8.45	10.31	11.76	12.69	13.26
Difference (days)	-	1.86	1.46	0.93	0.57
Percentage (%)	-	22.02	14.13	7.91	4.47
WDT (s)	0.5	1	2	4	8
Lifetime (days)	13.57	13.73	13.81	13.85	13.87
Difference (days)	0.31	0.16	0.08	0.04	0.02
Percentage (%)	2.32	1.19	0.60	0.30	0.15

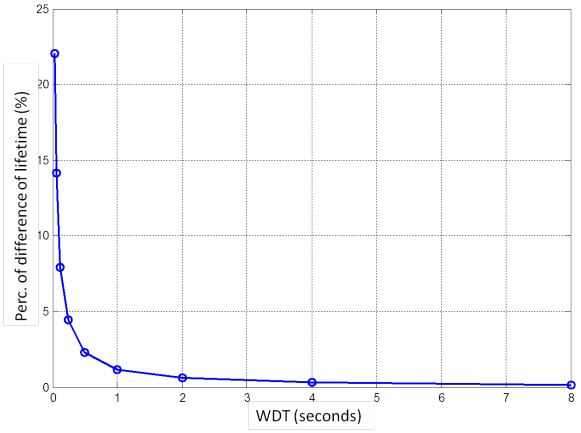


FIGURE 3.41: Percentage of difference of lifetime on the node according to the WDT

sleep mode more time. This means that the more time the node stays in sleep mode, the less amount of data the node can measure and the frequency to send these data is less. Then, the performance of the system decreases.

These results change totally if the current values in classes are modified. Now, we carry out the same study but with $I_C = 0.1$ mA for C_6. Figure 3.42 shows the node lifetime as function of t_{WDT} in this case whereas Table 3.13 presents all the information of lifetime, difference of lifetime and percentage of this difference. We observe that now the scenario is different. The relation between consecutive values at the beginning, with $t_{WDT} = 0.016$, 0.032, 0.064, 0.125 and 0.25 seconds has a difference greater than 50%. Also, in Figure 3.42, the gradient of the asymptotic function is less significant than in previous test. In the last value of WDT, $t_{WDT} = 8$ s, the percentage of difference according to $t_{WDT} = 4$ s is quite high, 4.27%. Then, we cannot consider that the node lifetime is close to the maximum possible value in this case when $t_{WDT} = 8$ s.

Then, to ensure the optimal point between lifetime and performance, we consider that the WDT values can reach higher values. This is possible if the clock frequency in microcontroller

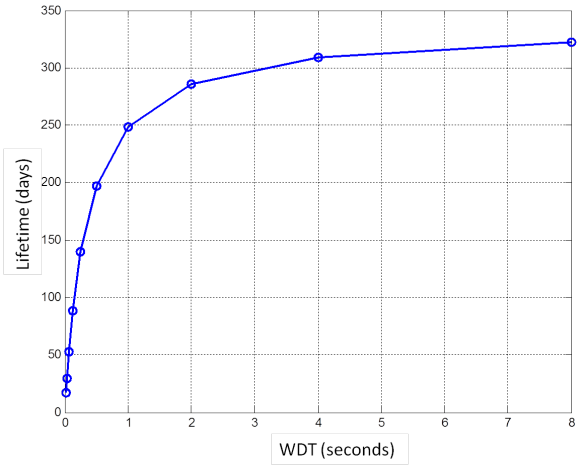


FIGURE 3.42: Lifetime of the node according to the WDT. Second test.

TABLE 3.13: Lifetime, difference of lifetime and percentage of increase of lifetime per WDT value.

WDT (s)	0.016	0.032	0.064	0.125	0.25
Lifetime (days)	16.79	29.83	52.98	88.64	139.44
Difference (days)	-	13.05	23.14	35.66	50.81
Percentage (%)	-	77.7	77.56	67.31	57.32

WDT (s)	0.5	1	2	4	8
Lifetime (days)	196.77	248.21	285.72	309.12	322.33
Difference (days)	57.33	51.44	37.50	23.40	13.21
Percentage (%)	41.11	26.14	15.11	8.19	4.27

decreases, but, in a real situation, the values of time and current in the other classes ($C_1 - C_5$) could also change. Table 3.14 shows the same parameters as in Table 3.13 but for WDT values equals and greater than 8 seconds. We observe that the lifetime of the node in this case converges near 337 days. Over $t_{\text{WDT}} = 32$ ms, the difference of lifetime becomes below 1%, then, for $t_{\text{WDT}} = 32$ ms the relation lifetime-performance reaches the optimal point. More than 30 seconds to take a value from a sensor may seem too much but some applications in WSN, as health building monitoring or smart parking, can bear this delay to recover the data without decreasing the performance of the system. Moreover, we prove in this sudy that if we choose a WDT value greater than 32 seconds, the lifetime would not really improve whereas the performance would decrease because the frequency to recover the data from the sensor would decrease.

TABLE 3.14: Lifetime, difference of lifetime and percentage of increase of lifetime per WDT value from 8 to 125 seconds.

WDT (s)	8	16	32	64	125
Lifetime (days)	322.33	329.37	333.01	334.86	335.77
Difference (days)	-	7.04	3.64	1.85	0.91
Percentage (%)	-	2.19	1.10	0.56	0.27

3.4.3 Changes in Transition Matrix

Previous Sections explain the cases when the average current values or the average time values are modified for each class. However, in these cases, the transition matrix of the classes remains unchanged. This transition matrix can also be modified to obtain different values of node lifetime. These changes of transitions represent the variations of the software, the embedded code, unlike the changes in current and time which can represent the modifications in software or in hardware. Then, these changes of the values in the transition matrix are also interesting to be studied.

We have previously explained that N_{data} corresponds to a parameter introduced in Figure 3.32 which represents the amount of measures obtained from the sensor and stored in the RAM of the microcontroller before the transmission. In previous examples, we fixed $N_{\text{data}} = 100$ data while each data is equivalent to 1 byte ($n_{\text{sens}} = 1$ byte). We can modify this parameter, N_{data}, what it means to change the frequency of transmitting and the length of the packets. The frequency of transmitting is modified with the transitions matrix and the length of the transmitted packets is equivalent to the time of serial communication between the microcontroller and the transceiver. Then, if we continue with the same embedded software in the node as well as the same description of classes as in Figure 3.34, the frequency of transmitting depends on the times of repetitions of C_5 and C_6 whereas the different packet length is equivalent to the time duration of C_3. According to the Figure 3.34, in an ideal system, the transitions between classes would reach the probability values shown in Table 3.15.

Then, for instance, in this experiment where $N_{\text{data}} = 100$, $p_{5,3}$ should be $p_{5,3} = \frac{1}{N_{\text{data}}} = 0.01$ and $p_{5,6} = 1 - p_{5,3} = 0.99$. We can observe in the corresponding transition matrix in (3.71) that the ensemble of the measurements platform and the automatic energy model have obtained $p_{5,6} = 0.9836$ and $p_{5,3} = 0.0102$. These values are really close to the theoretical values. Other unexpected transitions appear in the row of C_5, as $p_{5,2} = 0.0041$ and $p_{5,4} = 0.002$, this is mainly because of the noise obtained in the energy measurements but these small values do not cause some important changes in final results of lifetime.

We study now an experiment of analysis of the energy consumption by using the same embedded software but for different N_{data}. All the tests in this experiment have been done the

TABLE 3.15: Percentage of transitions in the test of Figure 3.34.

Type of transition	Probability of transition
C_6 - C_5	$p_{6,5} = 1$
C_5 - C_3	$p_{5,3} = \frac{1}{N_{\text{data}}}$
C_5 - C_6	$p_{5,6} = \frac{N_{\text{data}}-1}{N_{\text{data}}} = 1 - \frac{1}{N_{\text{data}}} = 1 - p_{5,3}$
C_3 - C_1	$p_{3,1} = 1$
C_1 - C_4	$p_{1,4} = \frac{2}{3}$
C_1 - C_2	$p_{1,2} = \frac{1}{3}$
C_4 - C_1	$p_{4,1} = 1$
C_2 - C_5	$p_{2,6} = 1$

same day in the same channel conditions. The embedded software carries out the same tasks explained in Figure 3.32. We study the cases when $N_{\text{data}} = [100,\ 80,\ 50,\ 20,\ 10]$ bytes. If N_{data} changes, the frequency of appearance of the $p_{5,6}$ and $p_{5,3}$ transitions also changes, as seen in Table 3.15. Furthermore, as the number of transmitted bytes varies, the time of serial communication between the microcontroller and the RF module varies in the same way. This serial communication is established at the beginning of the *TX State*, represented in Figure 3.34, *i.e.* this is C_3. The data rate of the serial communication is introduced in the embedded software as DR = 57600 bps. The data is transmitted via this serial communication byte per byte with a start bit and a stop bit before and after each byte, respectively. Then, the real amount of bits transmitted is $N_{\text{data}} \times (8+2)$ bits and the duration of the serial communication follows the expression (3.88).

$$t_{\text{serial}} = \frac{N_{\text{data}} \times (8+2)\ \text{bits}}{\text{DR}} \tag{3.88}$$

Then, according to (3.88), the t_{serial} for $N_{\text{data}} = 100$ bytes is calculated as $t_{\text{serial}} = \frac{100\times 10}{57600} = 17.36$ ms. In real experiment, we obtain an average time for C_3 of $t_{C3} = 18.3$ ms. This difference of $18.3 - 17.36 \simeq 0.94$ ms is due to a fixed time value of some tasks in RF module as the device initiation and the CCA/ED. This fixed value $t_{\text{init}} = 0.94$ ms should be evaluated for all the cases. Then, the total time for C_3 is $t_{C3} = t_{\text{serial}} + t_{\text{init}}$. With this information, we can calculate theoretically the time of C_3 for the different cases of N_{data} that we introduce in our model in order to estimate the lifetime in the node. Then, the values of $p_{5,6}$ and t_{C3} are calculated for different values of N_{data}. These values are expressed in Table 3.16.

The resulting values of $p_{5,6}$ in the experiments are also presented in Table 3.16. We observe that the difference between both $p_{5,6}$ values, in theory and in experiment, is quite small. The problem is that, because of the expression $p_{5,6} = 1 - \frac{1}{N_{\text{data}}}$, the interval between the $p_{5,6}$ values from $N_{\text{data}} = 100$ bytes to $N_{\text{data}} = 10$ bytes is not very significant, only from $p_{5,6} = 0.99$

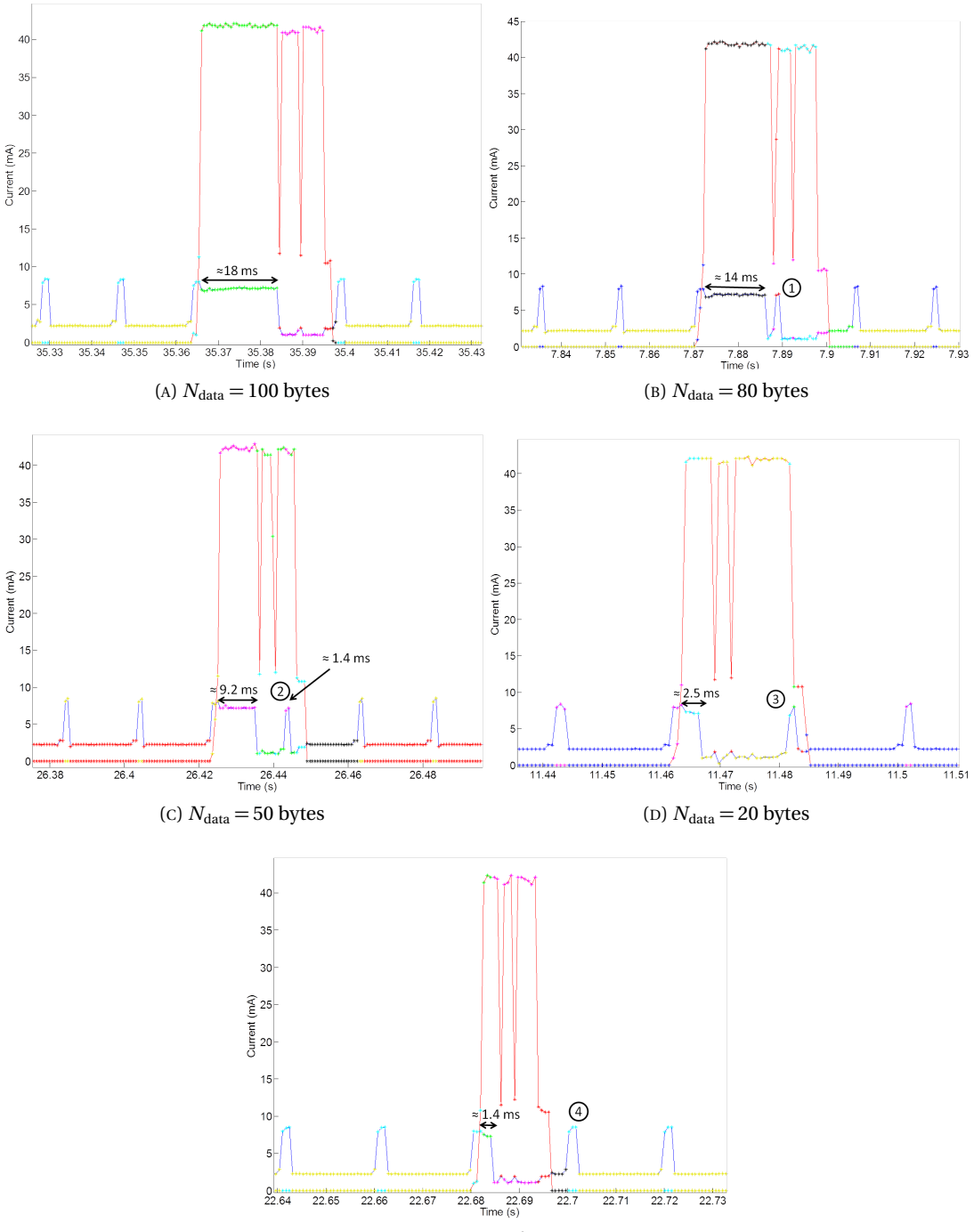
to $p_{5,6} = 0.90$. In this part, we add the node lifetime estimated from individual experiments carried out in the same conditions. The main goal here is to compare the experimental results of lifetime with the results obtained after the analysis with the automatic energy model. To carry out the estimation in the model, we take as reference the current values, the time values and the transition between states of the first test with $N_{\text{data}} = 100$. Then, we replace the $p_{5,6}$ and t_{C3} values in this model by the theoretical values shown in the *Theory* part of Table 3.16. The node lifetime values from this model are exposed in the *Model* part of Table 3.16. We can also observe the difference between both, model and experiment, and the error rate for the different N_{data}. From these results, we notice that in some cases, the model offers an important reliability with an error rate near 1% or 3.3%, but in other cases, the results are really differentiated, as in 15.19% of error for $N_{\text{data}} = 20$. In any case, the estimation of lifetime when the transition probabilities change, becomes more complicated. We have to know deeply the functions of our system. Even so, it can be a difficult task.

In this example, we distinguish the classes in Figure 3.43a when $N_{\text{data}} = 100$ bytes. In this case, $t_{C3} \approx 18$ ms. However, in Figure 3.43b, for $N_{\text{data}} = 80$ bytes, $t_{C3} \approx 14$ ms and it appears a new class in ① that corresponds to the first recovered data from the sensor after the serial communication (C_3). In Figure 3.43c, $N_{\text{data}} = 50$ bytes and measured $t_{C3} \approx 9.2$ ms. In this case, we have $t_{C3} = 5.2$ ms in experiment part of Table 3.16. This is due to the state marked in ② which corresponds to the same that ①, but, in this case, the algorithm has decided that this state is the same as C_3. This is why the mean time value of C_3 decreases until $t_{C3} = 5.2$ ms in experiment part of Table 3.16. In $N_{\text{data}} = 20$ bytes (see Figure 3.43d), the same problem as in $N_{\text{data}} = 50$ bytes appears. Now, the state of ③ is between two classes and the error between experiment and model reaches the 15.19 %. In Figure 3.43e, the state marked as ④ appears after the transceiver goes back to sleep mode. Then, this state is identified such as a normal state of recovered data from the sensor. This is why the error rate, in this case, decreases up to only 1.09 %.

In this Section, we show an example of the estimation of the node lifetime when we change

TABLE 3.16: Comparison node lifetime in model and in real experiments.

N_{data} (bytes)	Theory		Experiment			Model		
	$p_{5,6}$	t_{C3} (ms)	$p_{5,6}$	t_{C3} (ms)	Lifetime (days)	Lifetime	Difference (days)	Error (%)
100	0.99	18.3	0.9872	18.3	9.68	10.21	0.53	5.47
80	0.9875	14.84	0.9815	14.1	9.44	9.97	0.53	5.6
50	0.98	9.62	0.9622	5.2	9.13	9.43	0.30	3.28
20	0.95	4.41	0.9366	1.8	6.91	7.96	1.05	15.19
10	0.90	2.68	0.8898	1.7	6.43	6.36	-0.07	1.09



(A) $N_{\text{data}} = 100$ bytes

(B) $N_{\text{data}} = 80$ bytes

(C) $N_{\text{data}} = 50$ bytes

(D) $N_{\text{data}} = 20$ bytes

(E) $N_{\text{data}} = 10$ bytes

FIGURE 3.43: Currents in microcontroller and transceiver for N_{data}.

the probabilities between the transitions of the states in the Markov model. The complexity increases considerably because the change of a probability in the transition matrix can imply the change of other parameters such as the duration or the current level of the states. Then, it is necessary to know in detail the functions executed by the nodes. As perspectives, we will introduce in the automatic energy model the analysis of the DIO indicators from the node under evaluation. These indicators will link the actions of the embedded software with the

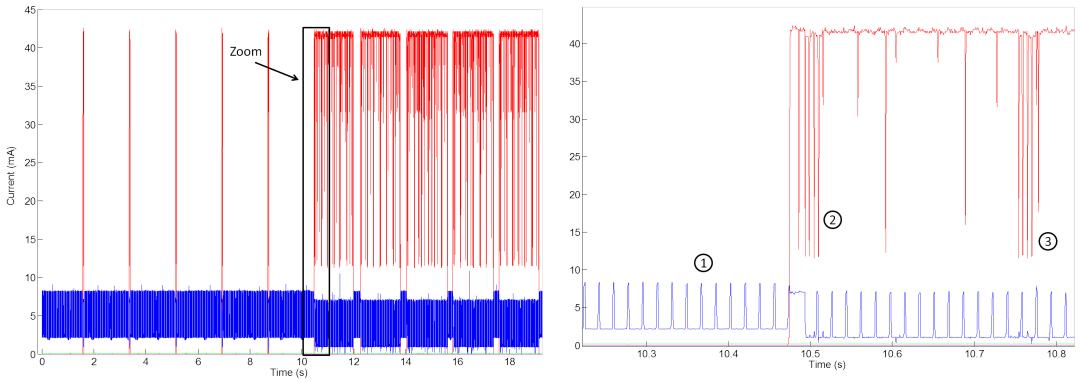
classes identified by the automatic energy model algorithm.

3.4.4 Energy Model with Interference

The automatic energy model can be used in different scenarios. The impact of the interference in energy consumption on a node has been studied in Section 2.4. We can analyze these energy values caused by the channel interference and create an energy consumption model for this type of scenario. The node to evaluate is the node composed by the XBee radio module and the ATmega328P microcontroller. The IEEE 802.15.4 standard establishes until 3 transmissions of the same frame (1 transmission and 2 retransmissions). We configure the XBee module to repeat this process of 3 transmissions up to 7 times (the first time corresponds to the normal case and the other 6 times are added by the XBee module) in order to give priority to the correct reception of the data. Then, as we explained above, if the RF module tries to send a packet during more time, the time that this device is switched on is longer and, thus, much more energy is consumed.

In this experiment, the node is programmed as in previous Sections. The embedded software consists in a periodic data collection application where the data transmission task is carried out after recovering N_{data} values. In this case, we take again the value of $N_{\text{data}} = 100$ bytes. In the first part of the experiment, the system works similarly: once the packet is ready with the N_{data} values, it is transmitted and the ACK frame is received in transmitter node. However, in the second part of the experiment, a source of interference is installed in the same wireless channel. This source of interference consists in a device such as a smartphone which contains an IEEE 802.11 (Wi-Fi) module. This module is configured to work in the channel 1 of IEEE 802.11 centered at the frequency of 2.412 GHz while the XBee module in the node is configured in the channel 12 of IEEE 802.15.4 at 2.410 GHz (see Figure 2.17b). As the bandwidth of a channel in IEEE 802.11 standard is 22 MHz and the bandwidth of a channel in IEEE 802.15.4 is 2 MHz with a difference of 5 MHz between the center frequency of two consecutive IEEE 802.15.4, an IEEE 802.11 channel can perturb an entire IEEE 802.15.4 channel. This fact causes serious problems in the reception of packets, in both, in the RF receiver with the reception of the data packets and in the RF transmitter with the reception of the ACK frames. The loss of the packets means retransmissions of the same packet, an increase of the time in active mode (transmission or reception modes) and, thus, an increase of the energy consumption in RF module.

Then, two different parts, one without interference and a second part with interference, are clearly contrasted in current measurements, as represented in Figure 3.44a. Moreover, in Figure 3.44b, we show a zoom area in the beginning of the first corrupted packet. In this figure, the marker 1 indicates the zone where the microcontroller wakes up to take a value



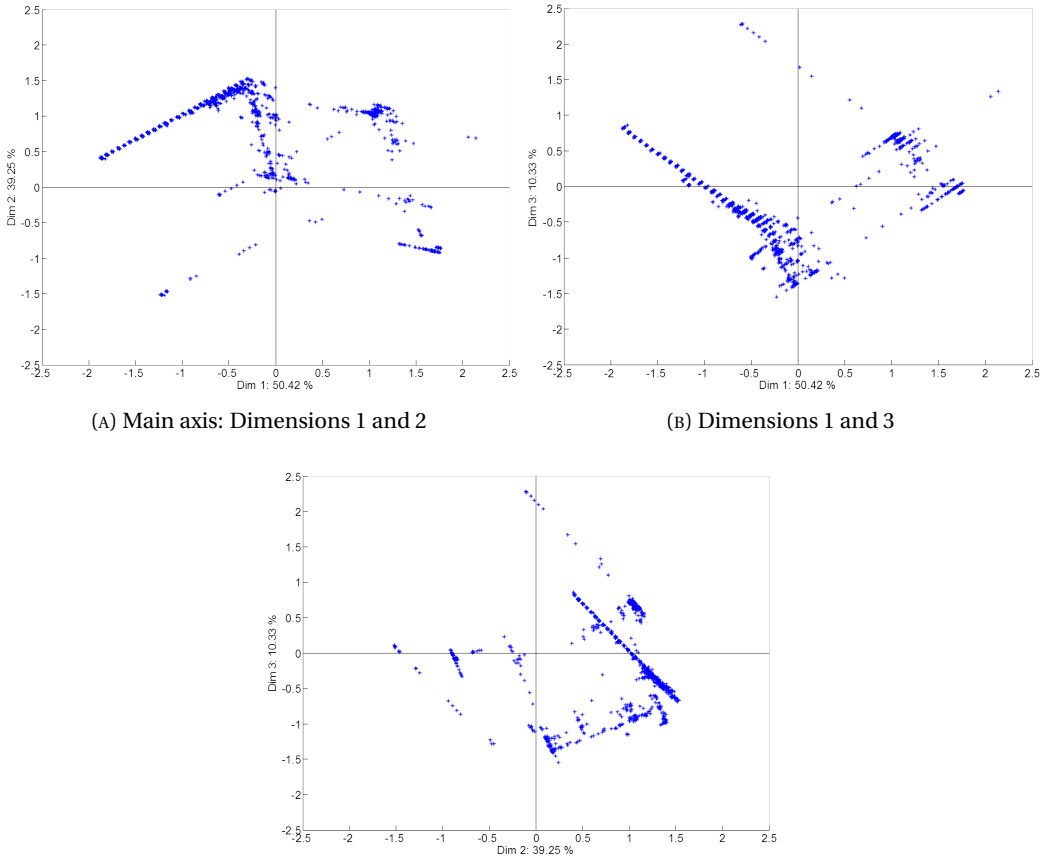
(A) Currents (B) Zoom in currents

FIGURE 3.44: Currents in experiment with interference.

from the sensor every t_{WDT} seconds (in this case, $t_{WDT} = 16$ ms). In ①, the microcontroller recovers information from the sensor. In ②, the microcontroller sends the information to the transceiver and goes back to sleep mode. After t_{WDT} seconds, the microcontroller wakes up again to continue with the same process of recovering values from the sensor. At the same time, the radio module tries to transmit correctly the packet to the sink node. Because of the interference in the channel, this packet is corrupted and the transceiver must resend it. In the zone of ②, the transceiver sends this packet up to 3 times. Then, if the packet is not well transmitted after the third attempt, the RF module remains activated for some time. After this time, the transceiver tries to send the same packet again, as seen in ③. This process will be repeated until 7 times in this experiment according to the configuration of the XBee module.

These current values have been analyzed by the automatic energy consumption algorithm. Firstly, the PCA algorithm offers the principal components from the current values in microcontroller and in RF module as well as from the periods of time for the individuals in the experiment. The three planes with the principal components are shown in Figures 3.45. We observe that the form of the set of points is totally different to the set of points seen previously in Section 3.3.2 in the experiment without interference. In this case, the groups of points are harder to distinguish. In this graph, we can observe the linear nature of this algorithm. Different lines of points are disposed in the space. This means that the variables (current and period values) of the individuals represented in the points of these lines are similar. Later, the HCA algorithm will take the decision of dividing these points in different groups. Besides, the dimensions 1 and 2 that comprise the main plane contains $50.42\% + 39.25\% = 89.67\%$ of information. This fact means that if we discard the dimension 3, we would not lose much information and we could execute the same analysis correctly.

The identification of the correct number of classes in the experiment is carried out by the HCA algorithm. For that, firstly, the between-inertia is calculated for a single class (all the



(A) Main axis: Dimensions 1 and 2

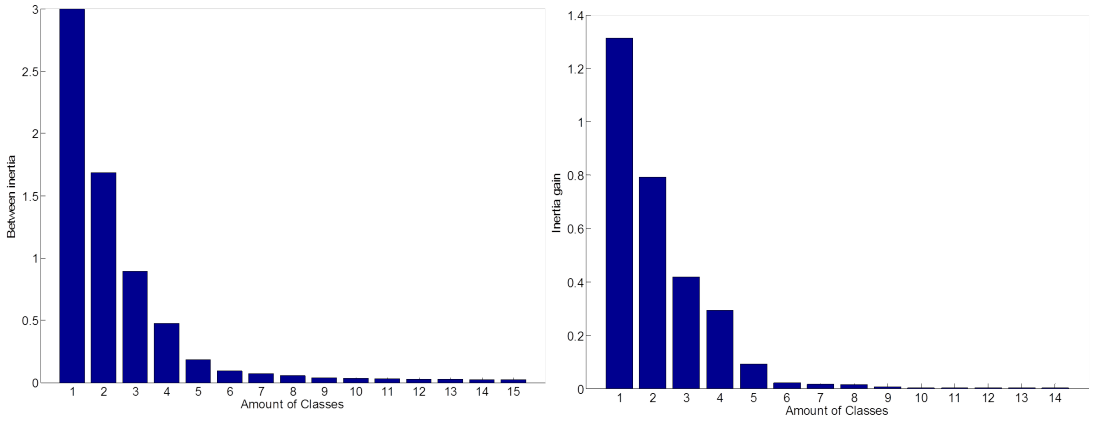
(B) Dimensions 1 and 3

(C) Dimensions 2 and 3

FIGURE 3.45: Principal components in the interference experiment.

points), then, we increase the number of classes to calculate this type of inertia until an amount of 15 classes. This between-inertia for each number of classes is represented in Figure 3.46a whereas the difference between the consecutive between-inertia values, *i.e.* the inertia gain is shown in Figure 3.46b. In this experiment, we keep the same threshold as in the experiment without interference, $\eta_q = 0.01$, in order to determine the number of classes from the inertia gain. In this case, the value of inertia gain is lower than η_q between 9 and 10 classes, then, the number of classes selected is 9 classes.

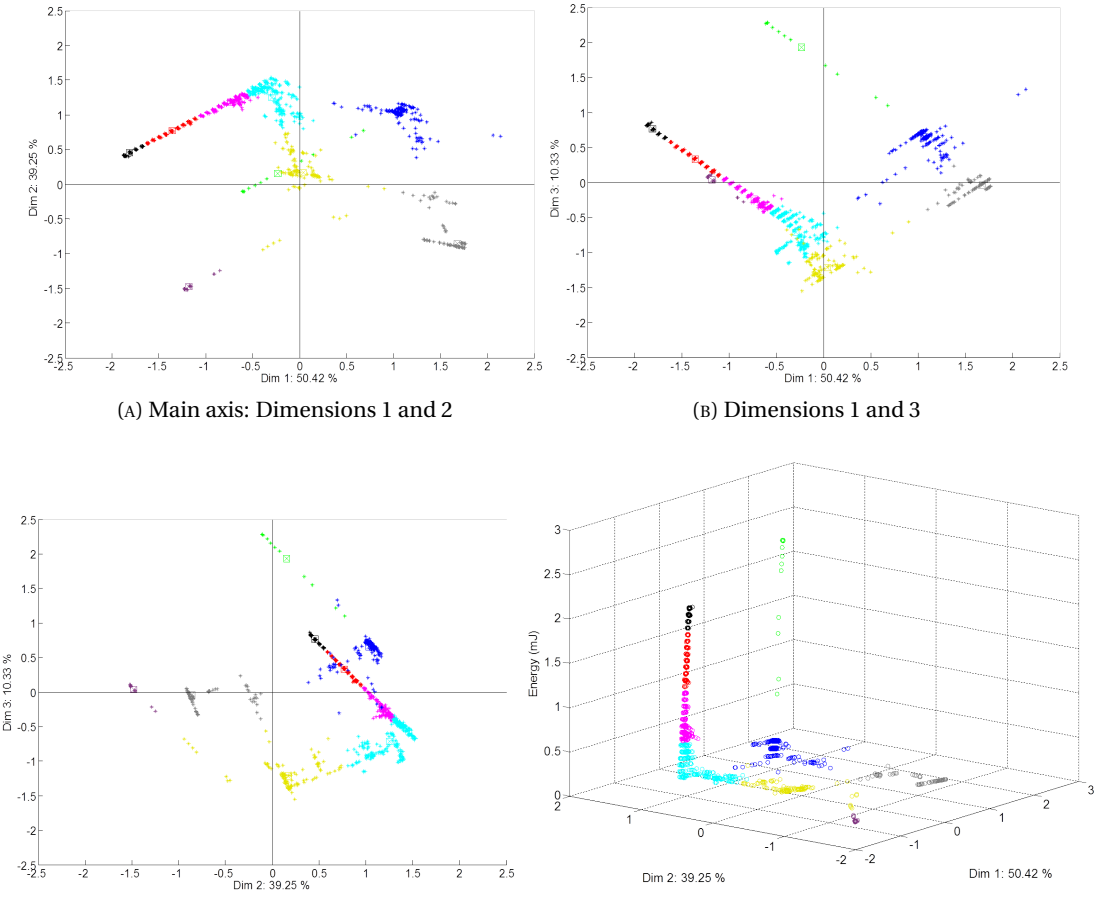
In Figures 3.47, the same set of points is presented for the three planes with the classification. We observe that division of groups of points is not easy in this case when perturbations appear. However, the algorithm carries out correctly this division. For instance, the black, red, pink and cyan classes follow the same line of points and they have been divided in a determined point, but this cut could occur in a different way. We can also observe that the average value for each class (represented with an 'x' symbol in a square) are rather separated between them. This is also an indicator of a proper classification. The set of all points in the space has been divided in 9 cluster or classes. In Figure 3.47d, we can observe in axis X and Y the main axis with the points in dimension 1 and 2, respectively, whereas the Z axis presents



(A) Between inertia (B) Inertia gain

FIGURE 3.46: Between inertia and gain inertia in the interference experiment.

the energy consumed by each individual represented as each point in this plane. This figure shows clearly the difference of energy consumption between the clusters as well as the correct classification carried out by the algorithm.



(A) Main axis: Dimensions 1 and 2 (B) Dimensions 1 and 3

(C) Dimensions 2 and 3 (D) Dimensions 1, 2 and energy per point

FIGURE 3.47: Classification of the set of points in the interference experiment.

In Figure 3.48, the current values are classified according to the results of the automatic energy model algorithm. We observe that the 9 classes are clearly different if we take into account the currents of both electronic components (microcontroller and RF module) and the time between two breaking points. In the first part of Figure 3.48, the transceiver is in sleep mode while the microcontroller wakes up each 16 ms to take a value from the sensor. The state where the microcontroller sleeps is represented with the class or state C_9, in purple color, whereas the period when the microcontroller wakes up corresponds to the C_8, in brown color. We observe in (3.89) and in (3.90) that the average values of currents and periods, respectively, have been correctly identified for C_9 and C_8. In the second part, when the radio module wakes up, we can observe the same behavior in microcontroller once the information is sent from the microcontroller to the radio module via the serial communication (represented with the green state or C_3). Even if the transceiver cannot transmit correctly the packet (this is why the transceiver remains in active mode), the microcontroller continues to takes a value from the sensor and to goes back to sleep mode. This behavior appears because, in this node, the transceiver can work independently of the microcontroller: when the transceiver has received the information from the microcontroller, the microcontroller can go to sleep mode and the transceiver will go to sleep when the information is transmitted to the sink in the network. Then, we notice the same behavior in the microcontroller before and after the serial communication (green state), but the automatic model algorithm distinguish correctly both situations, when the radio module is in sleep mode and when it is in active mode. In the second part, the periods when the microcontroller takes a value from the sensor are identified as the class C_7 (blue class). However, in the sleep periods, several states appear, generally, the black, red, magenta and cyan classes; C_1, C_2, C_4 and C_5, respectively. We observed in Figures 3.47 that these states are really close. The only difference

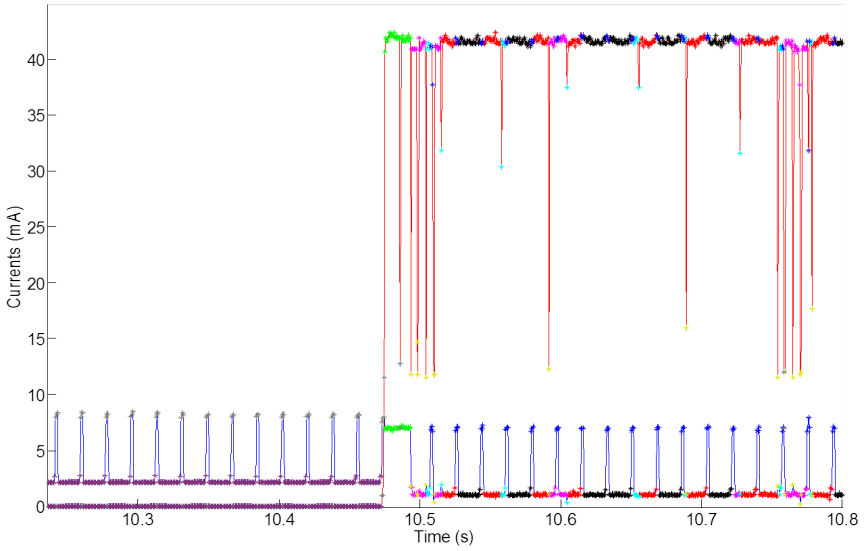


FIGURE 3.48: Currents and classes in the interference experiment.

between these classes is the time because the currents in transceiver and in microcontroller are similar, as we can identify in current measurements of Figure (3.48) as well as in average current values of 3.89. The difference between the average time values of these classes are shown in (3.90), with 16 ms, 11.4 ms, 5.2 ms and 1.3 ms for C_1, C_2, C_4 and C_5, respectively. In Figure 3.48, we notice that the principal causes of obtaining different classes for the same function (microcontroller in sleep and radio module in active) is the breaking points due to the sudden decreases of the current in the radio module. These peaks divide a period in two different periods and, thus, new states occur.

In (3.91), the average energy consumption are presented for each class, whereas, in (3.92), we show the number of times that a class appears in this experiment (n_C) as well as the percentage of these appearances (P_C). Besides, the transition matrix of this experiment is represented in (3.93). Then, the node lifetime in this experiment is estimated according to the third method presented in Section 3.3.5. It is 1.7558 days, *i.e.* almost 80% less than the lifetime for the same node with the same embedded software but in a wireless channel without interference. With these results, we prove once again the dramatic impact of the interference in the energy of a WSN node.

(mA)	C_1 (black)	C_2 (red)	C_3 (green)	C_4 (magenta)	C_5 (cyan)	C_6 (yellow)	C_7 (blue)	C_8 (brown)	C_9 (purple)
I_C^1	41.72	41.71	41.98	41.42	36.86	12.70	41.09	0.42	0
I_C^2	1.06	1.04	7.04	1.07	1.09	1.51	6.91	8.09	2.22

(3.89)

(s)	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
t_C	0.0160	0.0114	0.0148	0.0052	0.0013	0.0010	0.0015	0.0017	0.0161

(3.90)

(mJ)	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
E_C	2.2546	1.6051	2.3869	0.7306	0.1568	0.0477	0.2442	0.0475	0.1178

(3.91)

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
n_C	213	143	12	255	358	189	481	652	627
P_C (%)	7.27	4.88	0.41	8.70	12.22	6.45	16.42	22.25	21.40

(3.92)

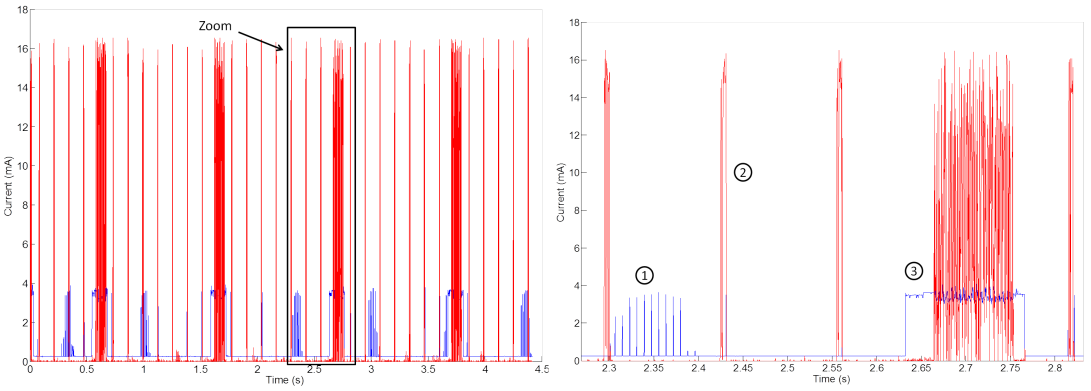
$p_{i,j}$	C_1 (black)	C_2 (red)	C_3 (green)	C_4 (magenta)	C_5 (cyan)	C_6 (yellow)	C_7 (blue)	C_8 (brown)	C_9 (purple)
C_1	0	0	0	0	0.0376	0.0047	0.9577	0	0
C_2	0	0	0	0	0.3636	0.2028	0.4336	0	0
C_3	0	0	0	0	0.1667	0.4167	0.2500	0.1667	0
C_4	0	0	0	0.0118	0.3098	0.3608	0.2980	0.0196	0
C_5	0.0279	0.1453	0	0.2291	0.2626	0.0866	0.2486	0	0
C_6	0.0319	0.1011	0	0.5266	0.2074	0.0372	0.0372	0.0426	0.0160
C_7	0.4054	0.1497	0.0021	0.1455	0.1726	0.0395	0.0644	0.0208	0
C_8	0.0031	0	0.0169	0.0015	0.0015	0.0061	0.0138	0	0.9571
C_9	0	0	0	0	0	0	0	1	0

(3.93)

We have shown in the transition matrix of (3.93) the complexity of this type of experiments when the number of classes increases. Some unexpected behaviors in the components of the node can occur. This is why, the study of the energy consumption just from general current values extracted from the datasheets of the electronic components is not realistic. We need different tools and methods as the Synergie platform to reach more realistic conclusions in order to optimize the system.

3.4.5 Energy Model in Other Type of Nodes

The automatic energy model has been tested with other type of nodes, not only with the node composed of the XBee transceiver and the ATmega328P microcontroller. In this case, the energy consumed in a WSN430 node [136] made by Inria [137] have been measured by the energy measurements platform of Synergie. We have used the same platform shown in Figure 2.14 with the same hardware and the same embedded software. Then, we measure the energy consumption in two electronic components: the MSP430 [138] microcontroller and CC2420 [110] radio module; even if we are able to measure until five components with the Synergie platform. We chose these two components because we consider that they are the most relevant components in this node. The embedded software in the microcontroller is the same as in Section 2.4.2. The node under evaluation is the transmitter node and it communicates with a receiver placed at 3.5 meters of distance. The test is carried out in a laboratory environment with a Wi-Fi access point installed in the same room. The CC2420 transceiver follows the IEEE 802.15.4 standard and the microcontroller is programmed to use the X-MAC [59] protocol where the transmitter wakes up from the sleep mode each 125 ms to listen the wireless channel. These periods of idle listening appears 7 consecutive times and,

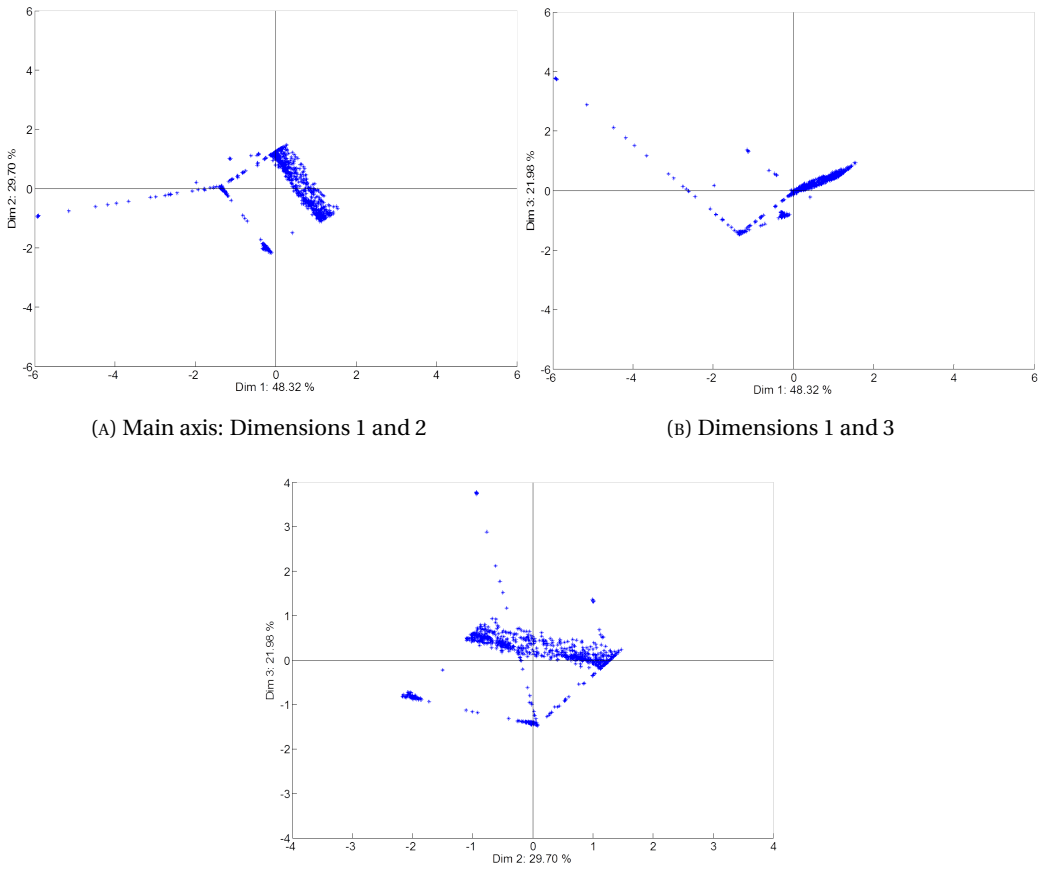


(A) Currents (B) Zoom in currents

FIGURE 3.49: Currents in experiment with WSN430 node.

then, in the eighth time that the node wakes up, the transmitter sends a packet of information to the receiver node. This software is configured through Contiki OS [139]. Figure 3.49a shows the current representations of the RF module (in red) and the microcontroller (in blue). Figure 3.49b depicts an area of zoom of these currents where we can distinguish some different behaviors of the components. In ①, the microcontroller wakes up several times between two periods of idle listening. This behavior was not established by our embedded software but we suppose that it is a behavior caused by an internal function of Contiki OS. When an OS runs in a WSN node, we risk to execute some unexpected actions that are not easy to manage. We can observe one of these actions in the current measurements of ①. Besides, the current of the transceiver in ② represents a period of idle listening. The transceiver wakes up and listens the channel during several milliseconds whether a packet includes the destination address of this node. In ③, the microcontroller wakes up before the transceiver and, then, wakes the transceiver up to transmit a packet. After this time, in this case, the microcontroller order RF module to goes back to sleep mode and, next, the microcontroller reaches again the sleep mode. We observe in Y axis of Figures 3.49 that the maximum current of the radio module reaches a value slightly greater than 16 mA instead of the 45 mA approximately consumed by the XBee transceiver included in the node previously studied.

The automatic energy model algorithm has analyzed the current measurements in this experiment. Firstly, the automatic segmentation identify the individuals according to the breaking points of the currents for each electronic component. Then, we take the three variables of each individual (period, current in microcontroller and current in transceiver), as in previous experiments. The PCA algorithm transforms these individuals in principal components in order to put in the same scale of importance all the dimension and reduce these dimensions of the values. In this case, we show in Figures 3.50 all the three 2-D planes formed by the three dimensions of the principal components. We observe clearly a large set of points in the central part of Figure 3.50a. These set of points corresponds to similar individuals that



(A) Main axis: Dimensions 1 and 2

(B) Dimensions 1 and 3

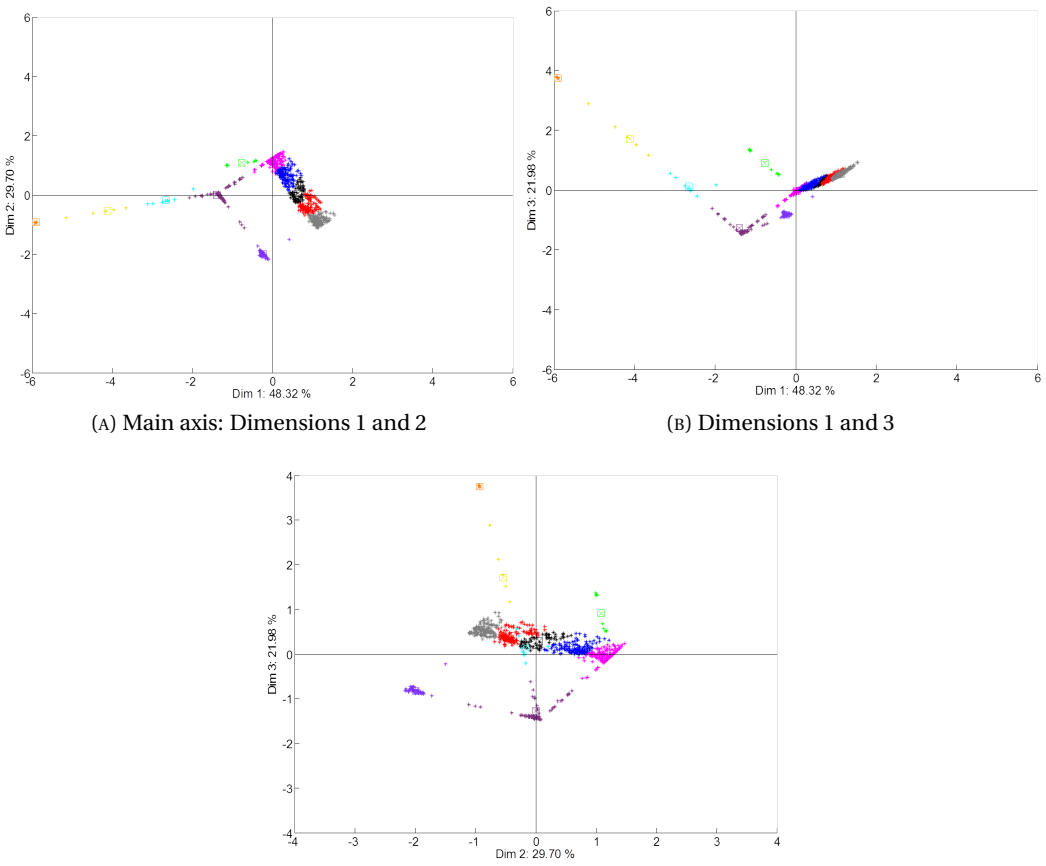
(C) Dimensions 2 and 3

FIGURE 3.50: Principal components with the WSN430 node.

will be difficult to classified later in HCA algorithm. in this case, we observe in the axis some different percentages of information for the dimensions. These are 48.32 %, 29.7 % and 21.98 % for dimesion 1, 2 and 3, respectively. In this case, the percentage of information for the least important dimension is 21.98 %. This value is not negligible and, then, the dimension 3 must be taken into account in any case for the rest of the analysis.

The HCA has automatically identified 11 different classes in this experiment. These classes are shown in the principal components in Figures 3.51. We observe that the large set of points in the center of the figures has been divided in five clusters (magenta, blue, black, red and grey classes). Other four classes are quite close to each other, these are the orange, the yellow, the cyan and the brown classes. In different positions, two other clusters are placed. These clusters are the green and the purple classes.

Figure 3.52 represents the current values shown above in Figure 3.49b but with the classification of the individuals. We distinguish clearly the 11 classes. The group of the five clusters (magenta, blue, black, red and grey) corresponds to the state of transmission of the packet in the transceiver. In this state, the current values changes rapidly from the highest to the lowest



(A) Main axis: Dimensions 1 and 2

(B) Dimensions 1 and 3

(C) Dimensions 2 and 3

FIGURE 3.51: HCA in the experiment with the WSN430 node.

values. Then, these magenta, blue, black, red and grey classes have been classified by order of increasing current, as seen in Figure 3.52. The second group of four clusters (orange, cyan, yellow and brown) are differentiated by the time values. We observe in current graphs that these four classes correspond to the instants when both components, microcontroller and RF module, are in sleep mode. Then, the current values are similar, as represented in (3.94), where I_C^1 represents the current in RF module and I_C^2 is in microcontroller. We observe that I_C^2 in C_8 is considerably higher than in the other three classes. This is because this class contains some variations in the whole experiment. However, this value of 0.6 mA is very low in comparison with other different classes which reach some current values of 8 mA, even near 16 mA. In (3.94), the average time calculated from the time values of every point in each class is also presented. We notice that the algorithm has correctly identified the difference between these average values of the periods where the period in C_8 is shortest with 0.5 ms. In Figure 3.52, we observe that this brown class is divided many times by the current peaks in microcontroller, where these peaks have been considered as magenta classes (C_4). After that, we can detect visually that the cyan class (C_5) has the second longest period of the sleep mode classes. In (3.94), t_C for C_5 is calculated much higher than the same value for C_8, 36.8 ms against 0.5 ms. Near 2 times longer than t_C of C_5, it is the period of C_6. In Figure 3.52,

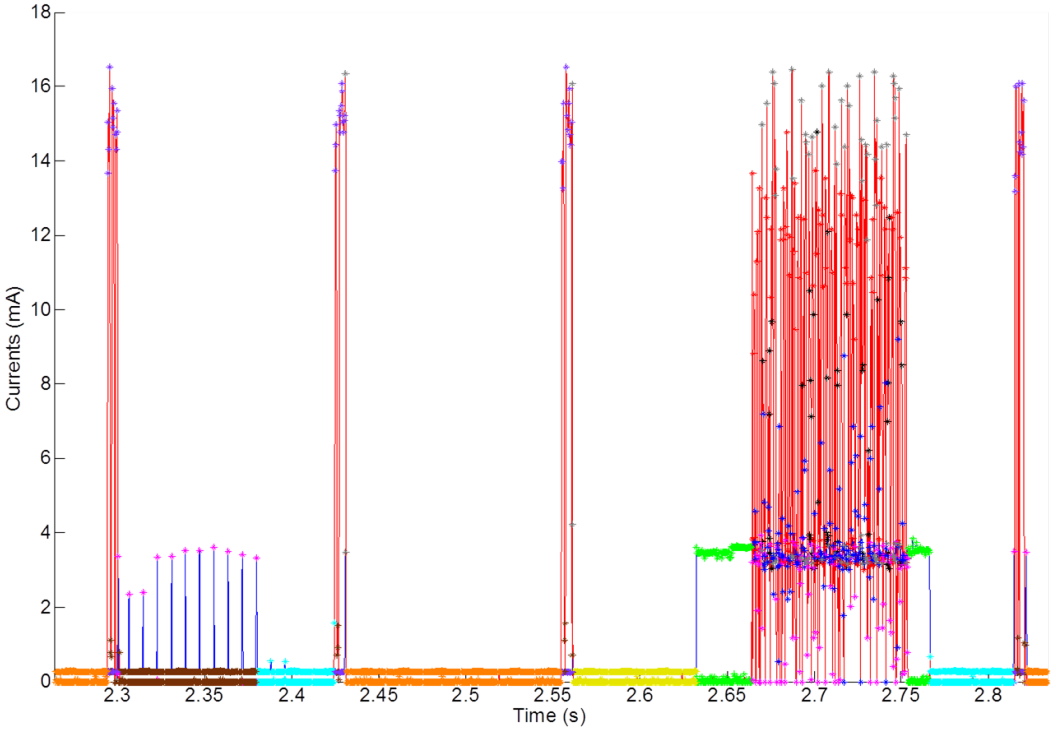


FIGURE 3.52: Currents and classes in the WSN430 node experiment.

we note clearly this difference between both classes. Finally, we have C_{11}, the orange class in this experiment, which represents the sleep period introduced in the embedded software without unexpected interruptions. These are the periods between two idle listening periods. The average value calculated by the automatic model algorithm is 123.6 ms. This value is near the time value configured in the embedded software that was 125 ms. We consider 125 ms the time between the beginning of an idle listening period and the beginning of the next idle listening, then, the time spent by the active mode of this idle listening is not included in this t_C of C_{11} because this active mode is represented with other different classes. The node lifetime has been estimated taking into account the results of average currents, average time values and the 11x11-size transitions matrix of all the classes. This lifetime results 18.38 days.

	C_5 (cyan)	C_6 (yellow)	C_8 (brown)	C_{11} (orange)
I_C^1 (mA)	0.005	0.006	0.6	0.007
I_C^2 (mA)	0.37	0.26	0.33	0.26
t_C (s)	0.0368	0.0754	0.0005	0.1236

(3.94)

The study of the node lifetime according to the length of the period in sleep mode (t_{sleep}) has also been realized for this experiment, as shown in Figure 3.53. These values are represented

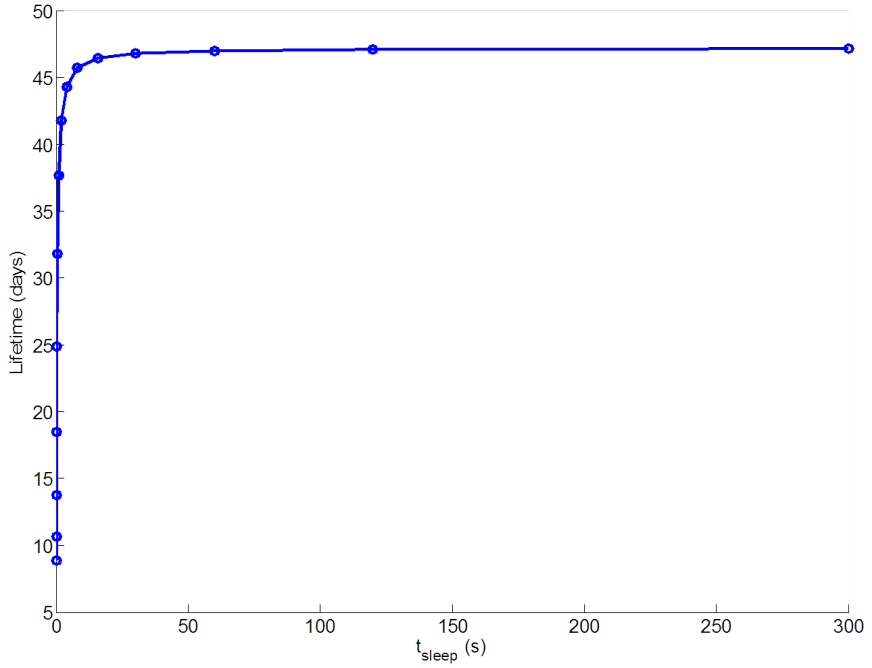


FIGURE 3.53: Lifetime in node for different periods in sleep mode.

in Table 3.17, as well as the difference in days and in percentage of two consecutive values. In the classification, we consider t_{sleep} as the average time in C_{11} because this class repre-

TABLE 3.17: Lifetime, difference of lifetime and percentage of increase of lifetime vs. t_{sleep}

t_{sleep} (s)	0.016	0.032	0.064	0.125	0.25	0.5	1	2
Lifetime (days)	8.86	10.66	13.78	18.47	24.89	31.77	37.67	41.80
Difference (days)	-	1.79	3.13	4.69	6.41	6.89	5.89	4.13
Percentage (%)	-	20.23	29.33	34.00	34.73	27.69	18.55	10.97

t_{sleep} (s)	4	8	16	30	60	120	300
Lifetime (days)	44.32	45.71	46.45	46.81	47.01	47.12	47.18
Difference (days)	2.51	1.39	0.74	0.35	0.21	0.10	0.06
Percentage (%)	6.01	3.15	1.62	0.76	0.44	0.22	0.13

sents the total time between two idle listening periods without interruptions. Nevertheless, the other three classes which have an influence in the sleep mode (C_5, C_6 and C_8) have been taken into account in a proportional way. For that, we calculate which proportion represents the t_{sleep} value under evaluation respect to the obtained $t_C = 123.6$ ms in C_{11} and we apply this proportion to the t_C values of C_5, C_6 and C_8. Then, as shown in Table 3.17, we chose as minimum t_{sleep} value 16 ms as in the precedent experiments for t_{WDT} while the maximum value is 300 s, *i.e.* 5 minutes between two idle listening functions. We observe that the lifetime increases with the t_{sleep} but when this sleep time reaches a value near 8 seconds, the lifetime values saturates. According to the percentage of the difference in lifetime seen in

Table 3.17, this percentage is below 1% with t_{sleep} values higher than 16 seconds. With this value, the node lifetime reaches 46.45 days. Then, if we are interested in achieve a better lifetime-performance relation, we would choose $t_{\text{sleep}} = 16$ s whenever we accept such a long idle listening in the transmitter.

The automatic energy consumption model has been developed in order to analyze the real energy measurements from a WSN node, estimate the lifetime of the node through a Markov chain and optimize the hardware and the software in the node in a clear and easy way by modifying the values of the Markov model. In this Section, we demonstrate that this automatic energy model can analyze the energy consumed by different nodes. The same model could be used to analyze the energy measurements resulting from other type of energy measurements platform. As perspectives, we will create a new version of the energy measurements platform faster and more accurate than the present version. With this ensemble, measurements platform and automatic energy consumption model, we expect to identify new states in the components of the node and to achieve a better optimization of the energy consumption.

Chapter 4

Conclusions and Perspectives

The energy efficiency in the nodes of WSN is a primary interest for different reasons: the harsh environments where the nodes are placed; the impossibility of changing the batteries in a huge amount of nodes due to the quick development of the WSNs in the new era of IoT; for ecological purposes, since the production of new batteries to replace the empty batteries in a large number of nodes will entail ecological issues. This is why it has been important to complete this work.

Synergie platform is presented as an ensemble of hardware, software and mathematical tools created exclusively in this work. In the first instance, we present the hardware platform entrusted with the real-time energy measurements component per component and instruction per instruction in a real node. We show some results obtained with this platform as the relation between the energy consumed per bit transmitted according to the packet size or the same relation when we store higher quantities of information in an external memory. These tests show that the energy per bit decreases if the amount of data stored in memory increases because the radio module, *i.e.* the most energy-hungry component in the node, is less used. However, after that, we use the energy measurements platform in an application where the impact of the interference in energy consumption is shown accurately. We prove that a longer packet of data has a higher probability to be corrupted by the interference in the physical channel than shorter packets (comparison between an application layer frame of 40 bytes and an ACK frame of 19 bytes) due to its higher on-air time. Another constraint when the transmitted packets are long is the decrease of the performance of the system, since the collection of the data takes more time and, thus, the sink node in the network receives the information later when the packet is longer. These parameters have to be taken into consideration when a WSN is installed.

The study of the impact of interference in energy consumption has confirmed the research of other authors where the interference in the physical channel increase dramatically the

amount of lost packets and, consequently, the energy consumption in the node. Furthermore, we distinguish this impact of interference according to the number of retransmissions of a single packet as well as of the times that an ACK frame is received. After several tests in an anechoic chamber and in a laboratory environment, we obtain a distribution of the number of retransmissions per interference level. This allows to calculate the average energy consumption as well as the node lifetime for a given battery per interference level. We also conclude with these experiments that the third transmission of a single packet fixed by the IEEE 802.15.4 standard entails more problems than advantages since its success rate is very low whereas the energy consumed when this type of packet appears is quite high.

Once the real energy measurements from the node are recovered, these results are analyzed by the Automatic Energy Consumption Model algorithm. This algorithm is based on clustering algorithms in statistics and creates a markov model from the real measurements. This fact avoids to use a theoretical energy model as well as to consult the datasheet of the electronic components in order to estimate the node lifetime. We prove that the energetic behavior of the components differs from the datasheet when these components act in an ensemble, in a circuit. This behavior is taken into considerations by the automatic energy consumption model. With this algorithm, we create an experimental energy model and we estimate the lifetime of the node in a realistic manner. Furthermore, we show that the average current values, the average period values and the transitions between the states identified by the algorithm can be modified in order to study an optimization of the energy in hardware and in software. These modifications are carried out manually in the first instance. The dynamic optimization of the software in the microcontroller of the node is a task to complete by our research team in a near future.

The development of a system which measures in real time the energy consumed in a node of WSN component per component and instruction per instruction is not usual in the literature. We present the general architecture of the measurement platform whose components (microcontroller, ADC, operational amplifiers) can be changed according to accuracy and data rate. Moreover, the automatic analysis of the measurements is an original element which can be used with any measurement platform. This sort of analysis that creates automatically an energy model based on a Markov chain is not found in the literature. Furthermore, the estimation of the node lifetime and the easy modification of the parameters in the Markov model allows to optimize quickly the hardware and the software in the node.

This study lets several perspectives to accomplish as future work. One of these perspectives is the employ of a more sophisticated energy measurement platform as the one presented in Section 2.5.1. The first tests with LabVIEW software has been realized but the analysis of the information has not been successfully completed because of the different data format. This is a problem to solve in a short period of time. Another issue to treat after obtaining the data

by the NI platform is the analysis of a huge quantity of information in the automatic energy consumption model algorithm. This algorithm should installed in a powerful computer or in a computer cluster.

Another perspective is the evolution of the study about the impact of interference in energy consumption. The next steps deal with the utilization of a USRP module to detect the power in the wireless channel using interference sources as Wi-Fi devices or microwave ovens. Other type of nodes will be created, including different communication technologies as LoRa. The behavior of the energy consumption according to the interference in the different frequency bands used by these technologies will be studied. The experimental setup and the algorithms of analysis in these tests will be the same as from the tests with the IEEE 802.15.4 nodes, then, we need just to create the nodes with the new communication technologies.

In order to further develop the automatic energy consumption model presented in Section 3.3, we plan to include in our algorithms the model of the real battery presented in Section 2.2.1 and used in the whole work. In this model, the effects of the series and parallel resistors and capacitors caused by the physical characteristics of the battery should appear as well as the resistor author of its self-discharge. The theoretical model of a supercapacitor will be also integrated in order to form a hybrid energy storage system (HESS) which supplies the node. A power management unit will control the use of battery or supercapacitor depending on the current required by the supercapacitor and the remaining energy in both storage components. Our research team works in real microbatteries and supercapacitors, this is why the interest of modeling both devices and integrating these models in the general automatic energy consumption model presented in this work.

The last work to realize in the future deals with the integration of these models in a system that will be able to change dynamically the code in the microcontroller of the node according to the forecasts carried out by the algorithms of the models. Only at this moment, the Synergie platform will measure the energy consumed by the node under evaluation, analyze these measurements in real time and inform to the node about how to change the code in order to reach the node of WSN autonomous in energy.

Publications

- V. Toldov , R. Igual-Pérez, R. Vyas, A. Boé, L. Clavier, N. Mitton. Experimental Evaluation of Interference Impact on the Energy Consumption in Wireless Sensor Networks. In *IEEE 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Coimbra, Portugal, 21-24 june, 2016.

- V. Toldov , J.P. Meijers, R. Igual-Pérez, R. Wolhuter, N. Mitton, L. Clavier. Performance Evaluation of LoRa Radio Solution for PREDNET Wildlife Animal Tracking Project. In *LPWAN 2016 Conference*, Paris, France, 18-20 may, 2016.

- R. Igual-Pérez, A. Boé, T. Vantroys, L. Clavier. Mesures et Modèle Énergétique d'un Noeud de Réseaux de Capteurs sans Fils. In *Journées Scientifiques 2016 URSI-France*, Rennes, France, 15-16 march, 2016.

- R. Igual-Pérez, A. Boé, L. Clavier, N. Rolland, T. Vantroys, G. Grimaud. Impact des Interférences sur la Consommation Énergétique d'un Noeud de Réseaux de Capteurs sans Fils. In *XIXèmes Journées Nationales Microondes (JNM 2015)*, Bordeaux, France, 2-5 june, 2015.

- R. Igual-Pérez, A. Boé, N. Rolland, L. Clavier. Experimentation and Modeling of Node Energy Consumption in Wireless Sensor Networks. In *17èmes Journées Nationales du Réseau Doctoral en Micro-Nanoélectronique (JNRDM 2014)*, Villeneuve d'Ascq, France, 26-28 may, 2014.

- R. Igual-Pérez, A. Boé, N. Rolland, L. Clavier, T. Vantroys, G. Grimaud. Experimental Results for Energy Modeling of a Wireless Sensor Network Node. In *4èmes Journées Nationales sur la Récupération et le Stockage d'Énergie pour l'Alimentation des Microsystèmes Autonomes (JNRSE 2014)*, Annency, France, 7-8 april, 2014.

Bibliography

[1] Rob van der Meulen. Gartner says 6.4 billion connected things will be in use in 2016, up 30 percent from 2015, November 2015. URL http://www.gartner.com/newsroom/id/3165317.

[2] John Edwards. The Internet of Things generates trillions in revenue by connecting billions of devices, March 2015.

[3] CSAM:Circuits, Systèmes et Applications des Micro-ondes. URL http://csam.iemn.univ-lille1.fr/.

[4] Ezgi Dogmus, Malek Zegaoui, Ludovic Largeau, Maria Tchernycheva, Vladimir Neplokh, Saskia Weiszer, Fabian Schuster, Martin Stutzmann, Martin Foldyna, and Farid Medjdoub. High structural quality InGaN/GaN multiple quantum well solar cells. *physica status solidi (c)*, 12(12):1412–1415, 2015. ISSN 1610-1642. doi: 10.1002/pssc.201510137. URL http://dx.doi.org/10.1002/pssc.201510137.

[5] P. Huang, C. Lethien, S. Pinaud, K. Brousse, R. Laloo, V. Turq, M. Respaud, A. Demortière, B. Daffos, P. L. Taberna, B. Chaudret, Y. Gogotsi, and P. Simon. On-chip and freestanding elastic carbon films for micro-supercapacitors. *Science*, 351(6274):691–695, 2016. ISSN 0036-8075. doi: 10.1126/science.aad3345. URL http://science.sciencemag.org/content/351/6274/691.

[6] A Pastre, O Cristini-Robbe, A Boé, K Raulin, D Branzea, H El Hamzaoui, C Kinowski, N Rolland, and R Bernard. Combination of porous silica monolith and gold thin films for electrode material of supercapacitor. *Materials Research Express*, 2(12):125001, 2015. URL http://stacks.iop.org/2053-1591/2/i=12/a=125001.

[7] R. Igual-Pérez, A. Boé, N. Rolland, and L. Clavier. Experimentation and modeling of node energy consumption in wireless sensor networks. In *17èmes Journées Nationales du Réseau Doctoral en Micro-Nanoélectronique (JNRDM 2014)*, May 2014.

[8] N. Cherifi, G. Grimaud, T. Vantroys, and A. Boé. Energy consumption of networked embedded systems. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 639–644, Aug 2015. doi: 10.1109/FiCloud.2015.90.

[9] Dave Evans. *The Internet of Things. How the Next Evolution of the Internet Is Changing Everything*. Cisco Internet Business Solutions Group (IBSG), April 2011.

[10] Ericsson. More than 50 billion connected devices. *White Paper*, February 2011.

[11] Casaleggio Associati. The evolution of internet of things. *White Paper*, February 2011.

[12] Jim Chase. The evolution of the internet of things. white paper. Technical report, Texas Instruments, September 2013.

[13] Libelium Smart World. URL http://www.libelium.com/top_50_iot_sensor_applications_ranking/#show_infographic.

[14] Amal Abbadi. Wireless Sensors Embedded in Concrete. In Vincent Le Cam, Laurent Mevel, and Franck Schoefs, editors, *EWSHM - 7th European Workshop on Structural Health Monitoring*, Nantes, France, July 2014. IFFSTTAR, Inria, Université de Nantes. Sensors IV.

[15] Sizing up the internet of things, August 2015. URL https://www.comptia.org/resources/sizing-up-the-internet-of-things.

[16] Jonathan Camhi. Bi intelligence projects 34 billion devices will be connected by 2020, November 2015. URL http://www.businessinsider.com/bi-intelligence-34-billion-connected-devices-2020-2015-11?IR=T.

[17] M. Ceraolo, A. di Donato, and G. Franceschi. A general approach to energy optimization of hybrid electric vehicles. *Vehicular Technology, IEEE Transactions on*, 57(3):1433–1441, May 2008. ISSN 0018-9545. doi: 10.1109/TVT.2007.909268.

[18] A. Barbato, A. Capone, G. Carello, M. Delfanti, M. Merlo, and A. Zaminga. Cooperative and non-cooperative house energy optimization in a smart grid perspective. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6, June 2011. doi: 10.1109/WoWMoM.2011.5986478.

[19] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 315–330, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-985-5. doi: 10.1145/1814433.1814464. URL http://doi.acm.org/10.1145/1814433.1814464.

[20] Mark Daugherty and Kevin C. Leonard. *Ultracapacitors for Renewable Energy Storage*. SolRayo LLC.

[21] H. Douglas and P. Pillay. Sizing ultracapacitors for hybrid electric vehicles. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6 pp.–, Nov 2005. doi: 10.1109/IECON.2005.1569143.

[22] F. Chraim and S. Karaki. Fuel cell applications in wireless sensor networks. In *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, pages 1320–1325, May 2010. doi: 10.1109/IMTC.2010.5488214.

[23] N. Rizoug, G. Feld, B. Barbedette, and R. Sadoun. Association of batteries and supercapacitors to supply a micro-hybrid vehicle. In *Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE*, pages 1–6, Sept 2011. doi: 10.1109/VPPC.2011.6043179.

[24] N. Rizoug, G. Feld, O. Bouhali, and T. Mesbahi. Micro-hybrid vehicle supplied by a multi-source storage system (battery and supercapacitors): optimal power management. *IET Conference Proceedings*, pages 3.2.05–3.2.05(1), January 2014. URL http://digital-library.theiet.org/content/conferences/10.1049/cp.2014.0399.

[25] Tao Ma, Hongxing Yang, and Lin Lu. Development of hybrid battery-supercapacitor energy storage for remote area renewable energy systems. *Applied Energy*, 153: 56 – 62, 2015. ISSN 0306-2619. doi: http://dx.doi.org/10.1016/j.apenergy.2014.12.008. URL http://www.sciencedirect.com/science/article/pii/S0306261914012574. Supercapacitors.

[26] Gateway to a new thinking in energy management - ultracapacitors. Technical report, Maxwell Technologies, January 2005.

[27] M.E. Glavin, P.K.W. Chan, S. Armstrong, and W.G. Hurley. A stand-alone photovoltaic supercapacitor battery hybrid energy storage system. In *Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th*, pages 1688–1695, Sept 2008. doi: 10.1109/EPEPEMC.2008.4635510.

[28] S. Saggini, F. Ongaro, C. Galperti, and P. Mattavelli. Supercapacitor-based hybrid storage systems for energy harvesting in wireless sensor networks. In *Applied Power Electronics Conference and Exposition (APEC), 2010 Twenty-Fifth Annual IEEE*, pages 2281–2287, Feb 2010. doi: 10.1109/APEC.2010.5433554.

[29] F. Ongaro, S. Saggini, and P. Mattavelli. Li-ion battery-supercapacitor hybrid storage system for a long lifetime, photovoltaic-based wireless sensor network. *Power Electronics, IEEE Transactions on*, 27(9):3944–3952, Sept 2012. ISSN 0885-8993. doi: 10.1109/TPEL.2012.2189022.

[30] D. Porcarelli, D. Brunelli, M. Magno, and L. Benini. A multi-harvester architecture with hybrid storage devices and smart capabilities for low power systems. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pages 946–951, June 2012. doi: 10.1109/SPEEDAM.2012.6264533.

[31] Y. K. Tan. *Energy harvesting autonomous sensor systems: design, analysis, and practical implementation.* CRC Press, 2013.

[32] W.S. Wang, T. O'Donnell, L. Ribetto, B. O'Flynn, M. Hayes, and C. O'Mathuna. Energy harvesting embedded wireless sensor system for building environment applications. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 36–41, May 2009. doi: 10.1109/WIRELESSVITAE.2009.5172418.

[33] Ibrahim W. Sutrisno and Ali G. Wahied. Power enhancement for piezoelectric energy harvester. In *World Congress on Engineering (WCE)*, volume II, July 2012.

[34] D.C. Hoang, Y.K. Tan, H.B. Chng, and S.K. Panda. Thermal energy harvesting from human warmth for wireless body area network in medical healthcare system. In *Power Electronics and Drive Systems, 2009. PEDS 2009. International Conference on*, pages 1277–1282, Nov 2009. doi: 10.1109/PEDS.2009.5385814.

[35] Benjamin L Cannon, James F Hoburg, Daniel D Stancil, and Seth Copen Goldstein. Magnetic resonant coupling as a potential means for wireless power transfer to multiple small receivers. *Power Electronics, IEEE Transactions on*, 24(7):1819–1825, 2009.

[36] Xun Zhou, Rui Zhang, and Chin Keong Ho. Wireless information and power transfer: Architecture design and rate-energy tradeoff. *Communications, IEEE Transactions on*, 61(11):4754–4767, November 2013. ISSN 0090-6778. doi: 10.1109/TCOMM.2013.13.120855.

[37] Geoffrey Werner Challen, Jason Waterman, and Matt Welsh. IDEA: integrated distributed energy awareness for wireless sensor networks. In *MobiSys*, pages 35–48, 2010.

[38] ISO/IEC standard 7498-1:1994. URL http://standards.iso.org/ittf/licence.html.

[39] Xee. my car, just better. Technical report, Eliocity, 2016.

[40] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM. ISBN 1-58113-589-0. doi: 10.1145/570738.570751. URL http://doi.acm.org/10.1145/570738.570751.

[41] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5):96–107, October 2002. ISSN 0163-5964. doi: 10.1145/635506.605408. URL http://doi.acm.org/10.1145/635506.605408.

[42] M.J. Mudumbe and A.M. Abu-Mahfouz. Smart water meter system for user-centric consumption measurement. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pages 993–998, July 2015. doi: 10.1109/INDIN.2015.7281870.

[43] S. Longhi, D. Marzioni, E. Alidori, G. Di Buo, M. Prist, M. Grisostomi, and M. Pirro. Solid waste management architecture using wireless sensor network technology. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5, May 2012. doi: 10.1109/NTMS.2012.6208764.

[44] J. Carle and D. Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *Computer*, 37(2):40–46, Feb 2004. ISSN 0018-9162. doi: 10.1109/MC.2004.1266294.

[45] Kechar Bouabdellah, Houache Noureddine, and Sekhri Larbi. Using wireless sensor networks for reliable forest fires detection. *Procedia Computer Science*, 19:794 – 801, 2013. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2013.06.104. URL http://www.sciencedirect.com/science/article/pii/S1877050913007126. The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).

[46] P.J. Pramod, S.V. Srikanth, N. Vivek, M.U. Patil, and C. Sarat. Intelligent intrusion detection system (in2ds) using wireless sensor networks. In *Networking, Sensing and Control, 2009. ICNSC '09. International Conference on*, pages 587–591, March 2009. doi: 10.1109/ICNSC.2009.4919343.

[47] Tayseer Alkhdour, Elhadi Shakshuki, Shokri Selim, and Uthman Baroudi. An optimal energy efficient and minimum delay scheduling for periodic {WSN} applications. *Procedia Computer Science*, 21:40 – 49, 2013. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2013.09.008. URL http://www.sciencedirect.com/science/article/pii/S1877050913008016. The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013) and the 3rd International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH).

[48] P. Sommer and R. Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on*, pages 37–48, April 2009.

[49] A. Kadri, E. Yaacoub, M. Mushtaha, and A. Abu-Dayya. Wireless sensor network for real-time air pollution monitoring. In *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, pages 1–5, Feb 2013. doi: 10.1109/ICCSPA.2013.6487323.

[50] Mohamed Amine Kafi, Yacine Challal, Djamel Djenouri, Messaoud Doudou, Abdelmadjid Bouabdallah, and Nadjib Badache. A study of wireless sensor networks for urban traffic monitoring: Applications and architectures. *Procedia Computer Science*, 19:617 – 626, 2013. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2013.06.082. URL http://www.sciencedirect.com/science/article/pii/S187705091300690X. The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).

[51] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, Dec 2004. ISSN 1536-1284. doi: 10.1109/MWC.2004.1368893.

[52] N.A. Pantazis, S.A. Nikolidakis, and D.D. Vergados. Energy-efficient routing protocols in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 15(2): 551–591, Second 2013. ISSN 1553-877X. doi: 10.1109/SURV.2012.062612.00084.

[53] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan 2000. doi: 10.1109/HICSS.2000.926982.

[54] S. Lindsey and C.S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1125–3–1130 vol.3, 2002. doi: 10.1109/AERO.2002.1035242.

[55] Arati Manjeshwar and D.P. Agrawal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 2009–2015, April 2001. doi: 10.1109/IPDPS.2001.925197.

[56] J.J. Lotf, M.N. Bonab, and S. Khorsandi. A novel cluster-based routing protocol with extending lifetime for wireless sensor networks. In *Wireless and Optical Communications Networks, 2008. WOCN '08. 5th IFIP International Conference on*, pages 1–5, May 2008. doi: 10.1109/WOCN.2008.4542499.

[57] Yan Wu, S. Fahmy, and N.B. Shroff. Energy efficient sleep/wake scheduling for multi-hop sensor networks: Non-convexity and approximation algorithm. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1568–1576, May 2007. doi: 10.1109/INFCOM.2007.184.

[58] Pangun Park. *Modeling, Analysis, and Design of Wireless Sensor Network Protocols*. PhD thesis, KTH School of Electrical Engineering, January 2011.

[59] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM. ISBN 1-59593-343-3. doi: 10.1145/1182807.1182838.

[60] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031508. URL http://doi.acm.org/10.1145/1031495.1031508.

[61] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 171–180, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958512. URL http://doi.acm.org/10.1145/958491.958512.

[62] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the Fourth ACM SenSys Conference*, pages 321–333, Boulder, Colorado, USA, November 2006. ACM. URL http://www.isi.edu/~johnh/PAPERS/Ye06a.html.

[63] Wireless coexistence in the 2.4 GHz band. application note. Technical Report AN5185, Freescale Semiconductor, Inc., September 2015.

[64] Ian F. Akyildiz and Mehmet Can Vuran. *Wireless Sensor Networks*. John Wiley & Sons Ltd., 2010.

[65] SIGFOX. URL www.sigfox.com.

[66] Minh-Tien Do, Claire Goursaud, and Jean-Marie Gorce. Interference Modelling and Analysis of Random FDMA scheme in Ultra Narrowband Networks. In *AICT 2014*, Paris, France, July 2014. URL https://hal.archives-ouvertes.fr/hal-01096493.

[67] Semtech wireless products. Technical report, Semtech Corporation, 2014.

[68] IEEE Std 802.15.4-2006. Technical report, 2006.

[69] Geoff V. Merrett, N.M. White, N.R. Harris, and B.M. Al-Hashimi. Energy-aware simulation for wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–8, June 2009. doi: 10.1109/SAHCN.2009.5168932.

[70] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. Springer, 2009.

[71] University of Southern California ISI. The Network Simulator - ns2. URL www.isi.edu/nsnam/ns.

[72] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim: A simulation framework for sensor networks. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '00, pages 104–111, New York, NY, USA, 2000. ACM. ISBN 1-58113-304-9. doi: 10.1145/346855.346870. URL http://doi.acm.org/10.1145/346855.346870.

[73] Ian T. Downard. Simulating sensor networks in NS-2, May 2004.

[74] András Varga. The OMNET++ discrete event simulation system. In *European Simulation Multiconference, Prague, Czech Republic*, 2001.

[75] V. Kunchakarra S. S. Iyengar R. Kannan C. Mallanda, A. Suri and A. Durresi. Simulating wireless sensor networks with OMNeT++. January 2005.

[76] National ICT Australia Ltd. Castalia - wireless sensor networks simulator. URL https://castalia.forge.nicta.com.au.

[77] TOSSIM. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM.

[78] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958506. URL http://doi.acm.org/10.1145/958491.958506.

[79] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. In Werner Weber, JanM. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-23867-6. doi: 10.1007/3-540-27139-2_7. URL http://dx.doi.org/10.1007/3-540-27139-2_7.

[80] MicaZ - Datasheet. Technical report, Crossbow Technology, Inc., . URL http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

[81] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications.

In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031518. URL http://doi.acm.org/10.1145/1031495.1031518.

[82] B.L. Titzer, D.K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 477–482, April 2005. doi: 10.1109/IPSN.2005.1440978.

[83] D. Blazakis, J. McGee, D. Rusk, and J.S. Baras. ATEMU: a fine-grained sensor network simulator. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 145–152, Oct 2004. doi: 10.1109/SAHCN.2004.1381912.

[84] O. Landsiedel, K. Wehrle, and S. Gotz. Accurate prediction of power consumption in sensor networks. In *Proceedings of the 2Nd IEEE Workshop on Embedded Networked Sensors*, EmNets '05, pages 37–44, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9246-9. URL http://dl.acm.org/citation.cfm?id=1251990.1253399.

[85] M. Imran, A.M. Said, and H. Hasbullah. A survey of simulators, emulators and testbeds for wireless sensor networks. In *Information Technology (ITSim), 2010 International Symposium in*, volume 2, pages 897–902, June 2010. doi: 10.1109/ITSIM.2010.5561571.

[86] Geoff Werner-Allen, Patrick Swieskowski, and Matt Welsh. MoteLab: A wireless sensor network testbed. In *Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, April 2005. URL http://www.eecs.harvard.edu/~mdw/papers/motelab-spots05.pdf.

[87] Mica2 - Datasheet. Technical report, Crossbow Technology, Inc., . URL http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf.

[88] Tmote Sky - Datasheet. Technical report, Moteiv Corporation, 2006. URL http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf.

[89] Clément Burin Des Rosiers, Guillaume Chelius, Tony Ducrocq, Eric Fleury, Antoine Fraboulet, Antoine Gallais, Nathalie Mitton, Thomas Noël, and Julien Vandaële. Using SensLAB as a First Class Scienti c Tool for Large Scale Wireless Sensor Network

Experiments. In *Networking 2011*, pages 241–253, Valencia, Spain, May 2011. URL https://hal.inria.fr/inria-00599102.

[90] A.-S. Tonneau, N. Mitton, and J. Vandaele. A survey on (mobile) wireless sensor network experimentation testbeds. In *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, pages 263–268, May 2014. doi: 10.1109/DCOSS.2014.41.

[91] E. Welsh, W. Fish, and J.P. Frantz. GNOMES: a testbed for low power heterogeneous wireless sensor networks. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 4, pages IV–836–IV–839 vol.4, May 2003. doi: 10.1109/ISCAS.2003.1206350.

[92] K. Shinghal, A. Noor, N. Srivastava, and R. Singh. Power measurements of wireless sensor networks node. *International Journal of Computer Engineering & Science*, May 2011.

[93] M. Healy, T. Newe, and E. Lewis. Power management in operating systems for wireless sensor nodes. In *Sensors Applications Symposium, 2007. SAS '07. IEEE*, pages 1–6, Feb 2007. doi: 10.1109/SAS.2007.374366.

[94] Hector Abrach, Shah Bhatti, James Carlson, Hui Dai, Jeff Rose, Anmol Sheth, Brian Shucker, Jing Deng, and Richard Han. MANTIS: System support for multimodal networks of in-situ sensors. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 50–59. ACM, 2003.

[95] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys '05, pages 163–176, New York, NY, USA, 2005. ACM. ISBN 1-931971-31-5. doi: 10.1145/1067170.1067188. URL http://doi.acm.org/10.1145/1067170.1067188.

[96] Philip Levis, David Gay, Vlado Handziski, Jan-Hinrich Hauer, Ben Greenstein, Martin Turon, Jonathan Hui, Kevin Klues, Cory Sharp, Robert Szewczyk, et al. T2: A second generation OS for embedded sensor networks. *Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep. TKN-05-007*, 2005.

[97] Antoine Courtay, Alain Pegatoquet, Michel Auguin, and C Chabaane. Wireless sensor network node global energy consumption modeling. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 54–61. IEEE, 2010.

[98] Imote2 - Datasheet. Technical report, Crossbow Technology, Inc. URL http://wsn.cse.wustl.edu/images/e/e3/Imote2_Datasheet.pdf.

[99] D. Hughes, E. Cañete, W. Daniels, R. Gowri Sankar, J. Meneghello, N. Matthys, J. Maerien, S. Michiels, C. Huygens, W. Joosen, M. Wijnants, W. Lamotte, E. Hulsmans, B. Lannoo, and I. Moerman. Energy aware software evolution for wireless sensor networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–9, June 2013. doi: 10.1109/WoWMoM.2013.6583386.

[100] Danny Hughes, Klaas Thoelen, Wouter Horré, Nelson Matthys, Javier Del Cid, Sam Michiels, Christophe Huygens, and Wouter Joosen. LooCI: A loosely-coupled component infrastructure for networked embedded systems. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '09, pages 195–203, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-659-5. doi: 10.1145/1821748.1821787. URL http://doi.acm.org/10.1145/1821748.1821787.

[101] C. Antonopoulos, A. Prayati, T. Stoyanova, C. Koulamas, and G. Papadopoulos. Experimental evaluation of a wsn platform power consumption. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8, May 2009. doi: 10.1109/IPDPS.2009.5161185.

[102] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press. ISBN 0-7803-9202-7.

[103] DT9816 - ECONseries. Low Cost USB Data Acquisition Modules - Datasheet. Technical report, DATA TRANSLATION, 2013. URL https://datatranslation.box.com/shared/static/aabdc787ad5cbd7d1f91.pdf.

[104] D. Macii, A. Ageev, and A. Somov. Power consumption reduction in wireless sensor networks through optimal synchronization. In *Instrumentation and Measurement Technology Conference, 2009. I2MTC '09. IEEE*, pages 1346–1351, May 2009. doi: 10.1109/IMTC.2009.5168665.

[105] S. Kellner, M. Pink, D. Meier, and E.-O. Blass. Towards a realistic energy model for wireless sensor networks. In *Wireless on Demand Network Systems and Services, 2008. WONS 2008. Fifth Annual Conference on*, pages 97–100, Jan 2008. doi: 10.1109/WONS.2008.4459362.

[106] L. Barboni and M. Valle. Experimental analysis of wireless sensor nodes current consumption. In *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pages 401–406, Aug 2008. doi: 10.1109/SENSORCOMM.2008.14.

[107] INA139 High-Side Measurement Current Shunt Monitor - Datasheet. Technical report, Texas Instruments, 2015. URL http://www.ti.com/lit/ds/symlink/ina139.pdf.

[108] LTC4150 Coulomb Counter - Datasheet. Technical report, Linear Technology, 2010. URL http://cds.linear.com/docs/en/datasheet/4150fc.pdf.

[109] Leonardo Barboni and Maurizio Valle. Experimental assessment of the battery lifetime in wsn based on the duty-cycle current average method. *Wireless Sensor Network*, 6 (10):212, 2014.

[110] *CC2420 Datasheet - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. Texas Instruments, February 2013.

[111] ATmega128L - Datasheet. Technical report, Atmel Corporation, 2011. URL http://www.atmel.com/images/doc2467.pdf.

[112] O. Hahm and S. Adler. Profiling energy consumption of wireless sensor nodes with almost zero effort. In *Communications (ICC), 2012 IEEE International Conference on*, pages 6273–6277, June 2012. doi: 10.1109/ICC.2012.6364831.

[113] Michael Baar, Heiko Will, Bastian Blywis, Thomas Hillebrandt, Achim Liers, Georg Wittenburg, and Jochen Schiller. The ScatterWeb MSB-A2 Platform for Wireless Sensor Networks. Technical report, Department of Mathematics and Computer Science at Freie Universität Berlin, September 2008.

[114] LPC2387 - Datasheet. Technical report, NXP, 2013. URL http://www.nxp.com/documents/data_sheet/LPC2387.pdf.

[115] E.P. Capo-Chichi, H. Guyennet, J.-M. Friedt, I. Johnson, and C. Duffy. Design and implementation of a generic hybrid wireless sensor network platform. In *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, pages 836–840, Oct 2008. doi: 10.1109/LCN.2008.4664289.

[116] FOX Board. Technical report, ACME Systems. URL http://www.acmesystems.it/FOXG20.

[117] Xiaofan Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 186–195, April 2007. doi: 10.1109/IPSN.2007.4379678.

[118] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. In *Information Processing in Sensor*

Networks, 2008. IPSN '08. International Conference on, pages 283–294, April 2008. doi: 10.1109/IPSN.2008.58.

[119] MAX1724. 1.5ţA IQ, Step-Up DC-DC Converters. Technical report, Maxim Integrated, 2015. URL https://datasheets.maximintegrated.com/en/ds/MAX1722-MAX1724.pdf.

[120] Rodrigo Fonseca, Prabal Dutta, Philip Levis, and Ion Stoica. Quanto: Tracking energy in networked embedded systems. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 323–338, Berkeley, CA, USA, 2008. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855741.1855764.

[121] Prabal Dutta, Jay Taneja, Jaein Jeong, Xiaofan Jiang, and David Culler. A building block approach to sensornet systems. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 267–280, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-990-6. doi: 10.1145/1460412.1460439. URL http://doi.acm.org/10.1145/1460412.1460439.

[122] Ruogu Zhou and Guoliang Xing. Nemo: A high-fidelity noninvasive power meter system for wireless sensor networks. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 141–152, April 2013. doi: 10.1109/IPSN.2013.6917581.

[123] Yi-Min Wang, W. Russell, A. Arora, Jun Xu, and R.K. Jagannatthan. Towards dependable home networking: an experience report. In *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*, pages 43–48, 2000. doi: 10.1109/ICDSN.2000.857512.

[124] MATLAB - Mathworks. URL http://fr.mathworks.com/products/matlab/.

[125] *ATmega328p Datasheet*. Atmel, Septembre 2010. URL http://www.atmel.com/Images/doc8161.pdf.

[126] *ZXCT1082/83/84/85/86/87 Datasheet*. Diode Incorporated, June 2011. URL http://www.diodes.com/datasheets/ZXCT1082_87.pdf.

[127] *XBee/XBee-PRO RF Modules*. Digi International Inc., September 2009. URL https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf.

[128] *ADXL345 Datasheet*. Analog Devices. URL http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf.

[129] *TR3000 Datasheet*. Murata Electronics, April 2015. URL http://wireless.murata.com/datasheet?/RFM/data/tr3000.pdf?ref=rfm.com.

[130] *IEEE Std 802.15.4-2006*. IEEE Standards Association, 2006.

[131] AD8216. Technical report, Analog Devices. URL http://www.analog.com/media/en/technical-documentation/data-sheets/AD8216.pdf.

[132] SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. Technical report, Semtech. URL http://www.semtech.com/images/datasheet/sx1276_77_78_79.pdf.

[133] Min Chen and Gabriel A. Rincon-Mora. An accurate electrical battery model capable of predicting runtime and i-v performance. *IEEE Transactions on Energy Conversion*, 21(2), June 2006.

[134] N6705A DC Power Analyzer. URL http://literature.cdn.keysight.com/litweb/pdf/N6700-90001.pdf?id=2076851.

[135] Ben Lauwens, Bart Scheers, and Antoine Van de Capelle. Performance analysis of unslotted csma/ca in wireless networks. *Telecommunication Systems*, 44(1-2):109–123, 2010.

[136] WSN430 node. URL https://github.com/iot-lab/iot-lab/wiki/Hardware_Wsn430-node.

[137] Inria. URL http://www.inria.fr/.

[138] MSP430F1611 - Datasheet. Technical report, Texas Instruments, March 2011. URL http://www.ti.com/lit/ds/symlink/msp430f1611.pdf.

[139] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, Nov 2004. doi: 10.1109/LCN.2004.38.

[140] Anwar Hithnawi, Hossein Shafagh, and Simon Duquennoy. Understanding the impact of cross technology interference on IEEE 802.15.4. In *Proceedings of the 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, WiNTECH '14, pages 49–56, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3072-5. doi: 10.1145/2643230.2643235.

[141] A. Sikora and V. F. Groza. Coexistence of IEEE802.15.4 with other systems in the 2.4 GHz-ISM-band. 3:1786–1791, May 2005.

[142] Wenqi Guo, W.M. Healy, and MengChu Zhou. Impacts of 2.4-GHz ISM band interference on IEEE 802.15.4 wireless sensor network reliability in buildings. *Instrumentation and Measurement, IEEE Transactions on*, 61(9):2533–2544, Sept 2012. ISSN 0018-9456. doi: 10.1109/TIM.2012.2188349.

[143] M. Petrova, Lili Wu, P. Mahonen, and J. Riihijarvi. Interference measurements on performance degradation between colocated IEEE 802.11g/n and IEEE 802.15.4 networks. In *Networking, 2007. ICN '07. Sixth International Conference on*, pages 93–93, April 2007. doi: 10.1109/ICN.2007.53.

[144] Bo Zeng, Yabo Dong, and Dongming Lu. A point-to-point interference measurement approach for large-scale wireless sensor networks. *IJDSN*, 2012, 2012.

[145] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, Oct 2002. ISSN 1536-1276. doi: 10.1109/TWC.2002.804190.

[146] J. Rohde and T.S. Toftegaard. Mitigating the impact of high interference levels on energy consumption in wireless sensor networks. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pages 1–5, Feb 2011. doi: 10.1109/WIRELESSVITAE.2011.5940899.

[147] inSSIDer User Guide. Technical report, MetaGeek, 2012. URL http://files.metageek.net/marketing/MetaGeek_inSSIDerUserGuide_WiFi-Scanner_2012.pdf.

[148] Five Factors to Consider When Implementing a Wireless Sensor Network (WSN), June 2012. URL http://www.ni.com/white-paper/10789/en/.

[149] Adam Dunkels. Rime — a lightweight layered communication stack for sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, January 2007.

[150] Sinem Coleri Ergen. ZigBee/IEEE 802.15.4 Summary. Technical report, University of Berkeley, September 2004.

[151] Sebastien Blavier. ExtraPuTTy. URL http://www.extraputty.com/.

[152] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56 (15):3531–3547, 2012.

[153] Pietro Nicoletti. IEEE 802.11 frame format, June 2005.

[154] NI 5751R User Guide and Specifications. Technical report, National Instruments Corporation. URL http://www.ni.com/pdf/manuals/375602b.pdf.

[155] NI FlexRIO FPGA Modules for PXI.

[156] NI HDD-8265 External RAID Enclosure. Technical report, National Instruments Corporation. URL http://www.ni.com/datasheet/pdf/en/ds-266.

[157] LabVIEW System Design Software. URL http://www.ni.com/labview/.

[158] 8-Slot PXI Express Chassis for PXI and PXI Express Modules. NI PXIe-1082. Technical report, National Instruments Corporation. URL http://www.ni.com/datasheet/pdf/en/ds-336.

[159] NI USRP-292x/293x Datasheet. Universal Software Radio Peripherals. Technical report, National Instruments Corporation. URL http://www.ni.com/datasheet/pdf/en/ds-355.

[160] Amal Abbadi. *Développement des réseaux de capteurs sans fil noyés dans le béton pour la surveillance des ouvrages de Génie Civil*. PhD thesis, Université Lille 1, September 2015.

[161] MSP430F23x0. Mixed Signal Microcontroller. Technical report, Texas Instruments, August 2011. URL http://www.ti.com.cn/cn/lit/ds/symlink/msp430f2370.pdf.

[162] nRF905 Low power Multiband Sub 1-GHz RF Transceiver. Technical report, Nordic Semiconductor. URL http://www.nordicsemi.com/eng/Products/Sub-1-GHz-RF/nRF905.

[163] H. Zhou, D. Luo, Y. Gao, and D. Zuo. Modeling of node energy consumption for wireless sensor networks. *Wireless Sensor Network*, 3(1):18–23, January 2011.

[164] Xinjie Chang. Network simulations with OPNET. In *Proceedings of the 31st Conference on Winter Simulation: Simulation—a Bridge to the Future - Volume 1*, WSC '99, pages 307–314, New York, NY, USA, 1999. ACM. ISBN 0-7803-5780-9. doi: 10.1145/324138.324232. URL http://doi.acm.org/10.1145/324138.324232.

[165] Li Daoliang Odey John Adinya. Transceiver energy consumption mmodel for the design of low power wireless sensor networks. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 3(4):135–139, April 2013.

[166] Roberto Verdone and Chiara Buratti. Modelling for wireless sensor network protocol design. In *Proc. IEEE International Workshop on Wireless Ad-hoc Networks (IWWAN), CD-ROM, London, United Kingdom*, 2005.

[167] Qin Wang, M. Hempstead, and Woodward Yang. A realistic power consumption model for wireless sensor network devices. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 1, pages 286–295, Sept 2006. doi: 10.1109/SAHCN.2006.288433.

[168] CC1000 Single Chip Very Low Power RF Transceiver - Datasheet. Technical report, Texas Instruments, 2007. URL http://www.ti.com/lit/ds/symlink/cc1000.pdf.

[169] D. Schmidt, M. Krämer, T. Kuhn, and N. Wehn. Energy modelling in sensor networks. *Advances in Radio Science*, 5:347–351, 2007. doi: 10.5194/ars-5-347-2007. URL http://www.adv-radio-sci.net/5/347/2007/.

[170] Waltenegus Dargie. Dynamic power management in wireless sensor networks: State-of-the-art. *Sensors Journal, IEEE*, 12(5):1518–1528, May 2012. ISSN 1530-437X. doi: 10.1109/JSEN.2011.2174149.

[171] Najmeh Kamyabpour and Doan B Hoang. Modeling overall energy consumption in wireless sensor networks. *arXiv preprint arXiv:1112.5800*, 2011.

[172] N. Kamyabpour and D.B. Hoang. A hierarchy energy driven architecture for wireless sensor networks. In *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, pages 668–673, April 2010. doi: 10.1109/WAINA.2010.46.

[173] A. Shareef and Yifeng Zhu. Energy modeling of wireless sensor nodes based on petri nets. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 101–110, Sept 2010. doi: 10.1109/ICPP.2010.19.

[174] Thomas N O Achia, Anne Wangombe, and Nancy Khadioli. A logistic regression model to identify key determinants of poverty using demographic and health survey data. In *European Journal of Social Sciences*, volume 13, 2010.

[175] F. Habyarimana, T. Zewotir, and S. Ramroop. Analysis of demographic and health survey to measure poverty of household in rwanda. *African Population Studies*, 29(1), 2015.

[176] Chih Lee, Ali Abdool, and Chun-Hsi Huang. PCA-based population structure inference with generic clustering algorithms. In *The Seventh Asia Pacific Bioinformatics Conference (APBC 2009), Beijing, China. 13Ű16 January 2009*, 2009. doi: 10.1186/1471-2105-10-S1-S73. URL http://www.biomedcentral.com/1471-2105/10/S1/S73.

[177] Nick Patterson, Alkes Price, and David Reich. Population structure and eigenanalysis. *PLoS Genet*, 2(12), 12 2006. doi: 10.1371/journal.pgen.0020190.

[178] K. Delac, M. Grgic, and S. Grgic. Statistics in face recognition: analyzing probability distributions of PCA, ICA and LDA performance results. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 289–294, Sept 2005. doi: 10.1109/ISPA.2005.195425.

[179] Huanhuan Yu, Rongda Chen, and Guoping Zhang. A {SVM} stock selection model within {PCA}. *Procedia Computer Science*, 31:406 – 412, 2014. ISSN 1877-0509. doi: http://dx.doi.org/10.1016/j.procs.2014.05.284. URL http://www.sciencedirect.com/science/article/pii/S187705091400461X. 2nd International Conference on Information Technology and Quantitative Management, {ITQM} 2014.

[180] Cecil C. Bridges Jr. Hierarchical cluster analysis. *Psychological reports*, 18(3):851–854, 1966.

[181] F. Husson, J. Josse, and J. Pagès. Principal component methods - hierarchical clustering - partitional clustering: why would we need to choose for visualizing data? *Technical Report - Agrocampus*, September 2010.

[182] S. Lê, J. Josse, and F. Husson. FactoMineR: an R package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, March 2008.

[183] W. N. Venables and D. M. Smith. An introduction to R. Technical report, R Core Team, 2015.

[184] F. Husson, J. Josse, S. Lê, and J. Mazet. *Package FactoMineR*, December 2014.

[185] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236 – 244, March 1963. doi: 10.1080/01621459.1963.10500845.

[186] Eric W. Weisstein. Normal Distribution. URL http://mathworld.wolfram.com/NormalDistribution.html.

[187] *FT232R Datasheet*. FTDI Chip, 2010. URL http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.