UNIVERSITY OF GENOA - ITALY

DIBE - Department of Biophysical and Electronic Engineering

Doctoral School: Science and Technology for Information and Knowledge

Ph.D. Course in Electronic, Computer Engineering, Robotics and Telecommunications

SSD: ING-INF/01 Elettronica

Ph.D Thesis

Power-Aware Design Methodologies for Embedded Wireless Sensors and Microsystems

By

Leonardo Barboni

Advisor: Ph.D Prof. Chiar.mo. Maurizio Valle

Coordinator: Chiar.mo Prof. Bruno Bianco

February 2010

Acknowledgments

First and foremost, I would like to thank God for all his numerous blessings bestowed upon me.

This work would not have been possible without the support of many people. Among this group, few individuals need special recognition:

First of all, I would like to express all my gratitude and love to my wife Lidia for being patient through the Ph.D study duration. She encouraged me to pursuit this work.

I am special grateful to Prof. Maurizio Valle who has been extremely supportive during my Ph.D activity in Italy. I thank him for his critical advices which provided me important guidelines that lent perspective to the performed research activity. And also for his investment of financial resources in my postgraduate education.

I wish to express my gratitude to Prof. Bruno Bianco. As Ph.D coordinator he has guided me through several aspects related to the Ph.D school, for instance, selection of courses and credits assignments. He offered invaluable assistance.

Deepest gratitude are also due to Prof. Paul Jespers, Prof. Piero Marietti, Prof. Gabriele Mosser and Prof. Michel Verleysen, who reviewed part of this work and for their contribution with rich comments, knowledge and suggestions.

To Lorenzo Ivaldi for his assiduous assistance in solving CAD problems.

Special thanks also to all anonymous reviewers who evaluated our articles for publications in conferences.

Thank a lot to the design house Canovatech for providing instruments and laboratory facilities.

I would like also to convey thanks to people from the IIE, Facultad de Ingenieria , Prof. Fernando Silveira, Prof. Conrado Rossi, Prof. Pablo Mazzara, Prof. Juan Pablo Oliver, Prof. Rafael Canetti and members of the SCAPA and CAP committee.
Thank you to the ANII committee for providing me the financial aid to buy an airline ticket (programm 720)

Thank a lot all.

Power-Aware Design Methodologies for Embedded Wireless Sensors and Microsystems

Summary

This thesis describes the work carried out, between January 2007 and December 2009, at the Microelectronics Lab, DIBE University of Genova.

Embedded microsystems for wireless sensors networks constitute a very powerful distributed and cooperative system for sensing. The increasing number of potential applications justify investment of research effort.
After a state of the art analysis, it has been identified several open issues and lack in methodologies to address research efforts in order to enhance the state of the art.
This dissertation explores novel methodologies and systems to enable embedded wireless sensors nodes applications capable of achieving low power consumption with reduced voltage supply.
In particular, the performed research activity has focused attention on: *i*) study of the hardware and embedded software interaction and its effects on the battery current consumption and lifetime; *ii*) study and development of power-aware signal conditioning circuitry and signal processing techniques.
The proposed goal has been searched through a systematic approach which goes from network level to node level because the power consumption must be reduced locally as well as globally. Therefore, we conducted the following strategies and research activity to successfully meet the goal:

It has been designed an electronic system based on a dedicated PCB to visualize node current consumption usage and charge extracted from the battery during node operating states. It has been study how software manages hardware resources and particularly, it has been observed the current consumption profile due to the radio transceiver and the associated protocol for communication.
Results can be used for detecting node operating modes that could generate current consumption profiles responsible of battery depletion acceleration. It is shown that sensor sampling activity and signal processing algorithms are very energy-demanding tasks, similarly with respect to the radio communications protocols. Experimental results are presented and discussed.
An experiment for battery full-depletion time estimation has been conducted in order to compare empirical results with theoretical battery lifetime estimations.

Moreover, the obtained information has been used to propose a heuristic algorithm for searching wireless sensors deployments. The algorithm predicts number of nodes, nodes distances and nodes output power level in order to give the deployment that reaches maximum battery lifetime compliant with user specifications.
Based on these studies, it has been observed that to effectively reduce power consumption, embedded wireless sensors networks can not work as simple set of wireless data loggers, instead, the cooperative and distributed signal processing capabilities must be taken as key advantage and nodes should be able to locally process information. The wavelet transform implementation into wireless sensors is then motivated by the node's need of performing local signal processing techniques in order to reduce power consumption. Such power consumption reduction is achieved because only signal features of interest would be transmitted, with reduced radio transceiver activity.
With this aim, we addressed the study of the digital Haar wavelet implementation into embedded

wireless sensors and its reliability to perform transients signal classification (or detection) *by thresholding*.

A discussion is also provided related to how the operating system TinyOs and the node as constrained hardware platform can yield together the better signal processing capability. Various issues that need to be considered for signal processing algorithms implementation with the operating system on general-purpose microcontrollers are discussed.
Moreover, this thesis describes a novel methodology for evaluating the signal-to-noise ratio (SNR) of data acquisition systems in wireless sensors microsystems. This methodology enables designers to evaluate the SNR regardless of non-stationary sensor signals behavior (with unknown signal shapes) or battery voltage decrease due to the depletion. The motivation arises because the use of data acquisition adaptive-circuits (with the purpose of achieving energy consumption reduction) can produce SNR variations.
Such variations, regrettably can jeopardize the digital signal processing performance. Hence, for the purpose of analyzing the SNR dependence on signal and data acquisition system features for general cases, a novel methodology for estimating the SNR has been deduced and tested. The methodology uses reduced hypothesis on the input signal and data acquisition features.

Chapter 1 gives detailed description of this thesis.

Finally, a critical evaluation of the work is offered at each chapter end, and final remarks and recommendations are presented in the conclusions section. A possible future direction related to this work is also presented.

This thesis has bee typeset in *documentclass [a4paper,11pt,oneside]* using LaTeX

Contents

List of Tables

Chapter 1

Thesis overview

Chapter Summary

This chapter covers the performed research activity description and the thesis organization.

1.1 Background

Wireless Sensors Networks (WSN from herein) are made by a large number of electronic devices named embedded wireless microsystems or sensor nodes. Depending on the application, a sensor node can measure physical variables such as temperature, pressure, sound, light intensity and so on. Measured data are transmitted through the network among nodes by implementing multihop communication protocols. Afterwards, data are collected by a centralized server or sink whose task is to infer or analyze from raw measured data, more structured spatial-temporal and higher-level information.

Generally speaking, a node is an electronic device built with limited hardware resources: one sensor node includes a general-purpose microcontroller, memory, sensors and circuitry for sensor signal conditioning, as well as AD converters, radio transceiver (frequently compliant with the standard IEEE 802.15.4 [**IEEE Standard 802.15.4-2006**]) and batteries for voltage power supply, as **Figure 1.1** shows.
Moreover, sensor node activity is controlled by the program that runs on the microcontroller under supervision of the operating system.

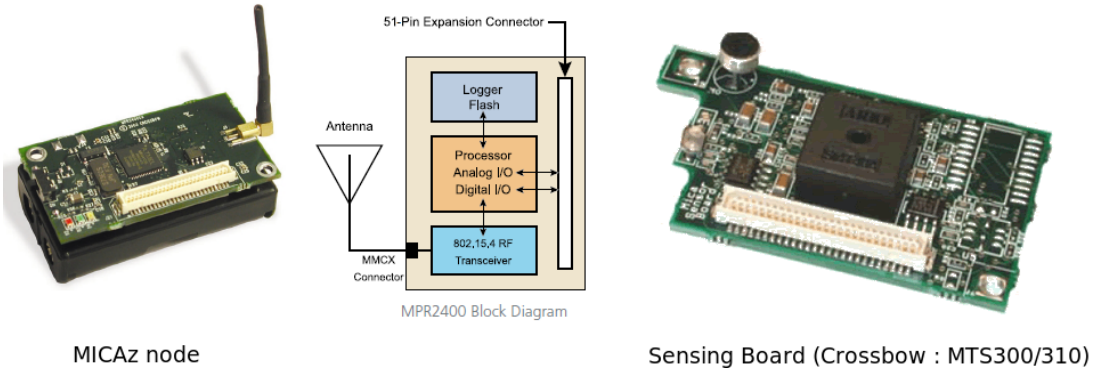


Figure 1.1: **Wireless sensor node example: the MICAz platform. Designed by Berkeley University of California and sold by Crossbow Inc. [MICAz (Crossbow)]. Key features: MCU Atmega128 [ATmega128, Datasheet], Clock 8MHz, 8 bit, AVR Core RISC 133 Instructions [AVR2002, App. Note], 128kB Flash-4KB EEPROM. RF Transceiver CC2420 [CC2420 Radio Device] 2.4GHz, IEEE 802.15.4 complaint, data rate: 250$kbps$, 2$MChips/s$. A 4-megabits serial Data Flash [AT45DB, Datasheet]. Operating System: TinyOS 1.x and TinyOS 2.x, open source and component-based, event-driven execution.**

1.1.1 Wireless sensors networks applications: state of the art

Wireless sensor networks constitute a very powerful distributed and cooperative system for sensing. An increasing number of potential applications for WSN justify the investment of effort in research.
An example of WSN application that shows the cooperative relationship between nodes is presented in [**Kim, S. (2007)**]. Authors describe a WSN implementation for structural health monitoring of the Golden Gate Bridge. The network is implemented by using 64 sensor nodes and each sensor node measures vibrations intensity in different points of the bridge.
Another example is presented in [**Lu, B. (2006)**]. In this case, the WSN is used for electrical machine energy monitoring and management in industrial environments. Wireless sensor devices are used in order to collect motor terminal voltages and currents.
Data are then sent through the network to the central collector that by using such information makes estimations on energy usage and structural engines conditions.
As authors mention, it is possible to implement on line non intrusive monitoring methods that allow to estimate the energy usage, motor load, average motor power and efficiencies.
We mention as another example the [**SLEWS Proyect**], which is a noteworthy example with important implications for surveillance in dynamic environments.
The WSN task is to monitor mass movements, so that an early warning alarm can be implemented in case of natural landslides. Geological data are measure in order to make probabilistic estimations of landslides risk. Data measured are water pressure, soil moisture, ground vibration, tilt, acceleration and strain in analog sensors buried under ground.

Wireless sensor networks bring great advantages for implementing forest monitoring applications. Not only researchers in disciplines of electronic and telecommunication, but also of biology, chemistry and geology highlight the advantages of using wireless sensor networks for forest monitoring.
It is evidenced by the large amount of recently reported works, among which we remark:
In [**Awang, A. (2007)**] it is deeply described the forest system monitoring named RIMBAMON. It has been developed by using WSNs technologies and the system is capable of constant monitoring of the conditions inside the forest. The lower cost with respect to satellite imaging techniques and the faster deployment implementation are remarkable features. Technical specifications are shown.
In [**Zhang, J.(2008)**], a wireless sensor network architecture for forest fire detection and monitoring is described. Authors highlight the catastrophic effects to the forest resources caused by forest fires.
A comparative analysis with respect to traditional techniques for forest fire detection is done, underlining the advantages of the wireless sensor network based on a ZigBee technique, which gives the possibility of implementing real time environmental parameters measurement. In this way, reactions for fire fighting or prevention could be rapidly implemented. The process of data transmission and radio link is presented in detail.
On the other hand, [**Laffea, L. (2006)**] is an article oriented to researchers in the discipline of bio-geochemistry. Authors offer the description of studies related to the carbon cycle .
A wireless sensor network is used to pick up information of several environmental parameters : $C0_2$, weather pressure, humidity, temperature . The system allows to researches to have accurate estimations of the forest-atmosphere $C0_2$ exchange due to the micro scale measurements and analysis provided by the distributed nature of the wireless sensor network.
In [**Porter, J. (2005)**] it is discussed how the greater and temporal resolution provided by wireless sensor networks enable ecologist and biologies to start new inquires in environmental sciences. Authors discuss the multiple useful features of WSN for environmental monitoring such as: unobtrusive observations, real time observation capability, scalable infrastructure for multiple goals and intensive or extensive sampling selection.

Technical considerations and issues are presented.
Finally, in [**Kim, Y. (2008)**] it is proposed a remote sensing and control of an irrigation system using a distributed wireless sensor network.

During the thesis reading, it is offered to the readers other real applications examples.

1.2 Identified research trends and open issues

After a study of the state of the art, it has been identified several open issues to address research efforts in order to enhance the state of the art.

In general, wireless sensor nodes are powered by limited energy resources (batteries) and one of the main concern of WSNs developers is to extend node lifetime. The node lifetime is correlated with the battery current usage profile that depends not only on hardware platform features but also on how the application software manages node hardware resources.
Moreover, the recent advances in nanotechnology, in MEMS and signal processing algorithms have meaningfully contributed to node current consumption reduction. Various issues that need to be considered have been highlighted in [**Chandrakasan, A. (2002)**], [**Rabaey, J.M. 2000**], [**Fang, Z. (2006)**],[**Aeschlimann, F. (2004)**], [**Bajwa, W. (2007)**] and [**Jain, A. (2004)**].
Among WSN designers, there exist a great interest related to the possibility of developing sensor nodes powered by energy harvesting systems in order to further extended node lifetime or to achieve self-powered devices. This means that the sensor node operation must be compatible with energy harvesting systems (for instance with systems that use piezoelectric materials for energy harvesting from mechanical vibrations) and at the same time it must fulfill user requirements.
Nonetheless, energy harvesting systems are effective to supply power to sensor nodes whereas the sensor node current and power usage profile are kept below very low thresholds, e.g. $100\,\mu W$.
Examples and case studies are presented in [**Sakunia, S. 2007**], [**Cantatore, E. (2006)**], [**Jaing, B. (2007)**] and [**Ammar, Y. (2008)**].
To have better insight, we briefly mention that nowadays, commercial radios often used for implementing wireless sensor nodes, consume $65\,mW$ for a packet transmission (29 bytes and transmission duration $6.44\,ms$) at the lowest radio output power level; this value accounts only for the radio transceiver power consumption [**Barboni, L. (2008)**].
Then, to implement sensor nodes energy-harvested supplied or to extend lifetime, it is necessary to develop appropriate strategies and wireless sensor nodes able to work not only with reduced-voltage power supplies (e.g. 1-1.2 V) but also with very low current consumption and consequently, low energy consumption.
Moreover, energy harvesting systems effectiveness depends on the availability of energy from the environment (under vibration or electromagnetic forms for instance). The amount and time availability of harvested energy is not known in advance and techniques to estimate as well as to control the node activity as function of the available energy have to be tailored.

Therefore:

The performed research activity has involved novel strategies to enable wireless sensor nodes and wireless sensor networks applications to operate at low-voltage and low-current consumption.

In particular, the performed research activity has focused attention on: *i*) study of the hardware and embedded software interaction and its effects on the battery current consumption in order to enhance the battery lifetime; *ii*) study and development of power-aware signal conditioning circuitry and signal processing techniques.

Signal conditioning circuitry involves not only the sensor current consumption but also current consumption of the associated interface circuits (including A/D converters). On the other hand, software implemented algorithms for signal processing should be oriented towards reduced resources hardware platforms (as wireless sensor nodes are).
For this, in order to obtain low-voltage and low-current consumption signal conditioning and processing, we propose to study how to implement dedicated algorithms capable of exploiting all hardware-software capabilities i.epictures/. by working with parameters that optimize tradeoffs in the signal conditioning circuitry (for instance, frequency sampling, voltage power supply that determines sensor dynamic range, resolution, memory usage, data representation) and which have impact in the algorithm performance.

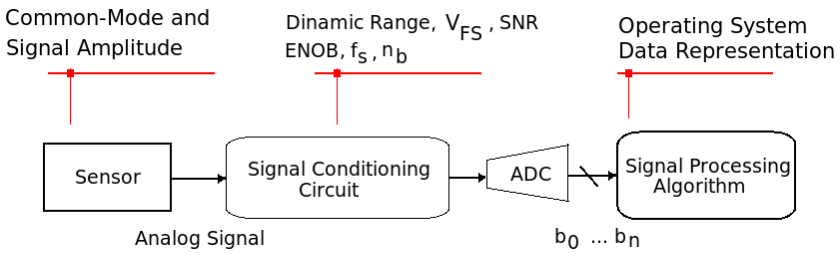


Figure 1.2: **Signal processing chain, block diagram. Current (or energy) consumption reduction can be accomplished by tailoring data acquisition system features and signal processing algorithms.**

The goal of the conducted research activity is to contribute to the development of novel methodologies and techniques for designing embedded wireless microsystems capable of operating at extremely low current and power consumption and to extend the node (i.e. battery) lifetime.

1.3 Addressed topics description

The proposed goal has been searched through a systematic approach which goes from network level to node level because the power consumption must be reduced locally as well as globally. Therefore, we conducted the following strategies and research activity to successfully meet the goal.

1. A board with commercial IC was built and current consumption measurements of wireless sensor nodes were done. The board is composed by three different circuits that having current monitors, amplifiers and coulomb counters, they can take detailed information of node current consumption profiles, becoming possible to study how software manages hardware resources. Particularly, it has been observed the current consumption profile due to the radio transceiver and the associated protocol for communication.
   Since writing optimized code is a way to avoid wasted energy, it is important to know if a program could be optimized proving to be beneficial for the node longevity.

2. Detailed study of battery models was performed, in order to understand the battery depletion process due to the node current consumption. The selected model is accurate enough as well as easy tractable. This kind of model will be further used to implement energy availability prediction and adaptive techniques for task scheduling.
   Even though the battery depletion process has been widely studied and reported in literature, in this work we follow a novel approach which consists in to introduce the battery voltage as function of the extracted charge for lifetime estimate.

3. Study of optimal deployments in WSN. Trade-off related to distance between nodes, power consumption and node activity have been studied and based on this, a design criterion and enhanced optimization techniques were proposed. The deployment customization for wireless sensor networks is a highly constrained optimization problem that involves multiple aspects. Designers have to reach the network layout that best meets specifications without violating any constraint, but reaching at the same time, the maximum lifetime before batteries become exhausted.
   The proposed methodology to handle this optimization problem, takes into account specific application constraints and hardware features which have been often reported in literature as vanished or assumed as ideal.
   Specifically, it has been taken into account the effects of the used protocol for radio communication and real radio propagation channels.

4. In order to reduce power consumption due to data transmission, the WSN can not work as a simple set of wireless data loggers, instead, the cooperative and distributed signal processing capabilities must be taken as advantage and nodes should be able to locally process information, with posteriori transmission only of the captured events and features of interest.
   Thus, information must be processed at node level, reducing the amount of transmitted data and consequently power consumption usage due to radio transceiver activity.
   Nevertheless, recently, sensor data acquisition and signal processing algorithms have started to be reported as a main power dissipation source, together with the RF front-end.
   We confirm this from our experience (see item 3); therefore, implemented algorithms must be aware to use sensors and data acquisition circuitry, without wasting energy.
   In wireless sensor nodes, the signal processing algorithm behavior depends on the software implementation and hardware platform features as well.
   For this, the implementation must be performed taking into account the fact that hardware and software work in conjunction.
   We also have studied how to implement algorithms capable of to work with the ADC resolution as low as possible as well as with adaptive sampling frequency, reducing the ADC sampling speed if signal features allow this.
   From the case study point of view, we propose to implement digital wavelet transform (DWT) based algorithms.
   The reference signal to be measured is the acceleration and sensors are basically accelerometers. The measured information can be used for structural health monitoring of engines, motors, buildings and so on or for detecting seismic micro-vibrations that adverts possible landslides.

5. The effects of voltage power supply reduction must be taken into account because it affects directly the signal feature extraction procedure. It is important to underline that it is reasonable to assume scenarios where there could be nodes for which the voltage power supply is not constant.
   It is due to, from one side, if the node is battery powered, the battery voltage experiences progressive droop. On another side, if the node uses a harvesting system, such system delivers energy to the node from a reservoir that must be recharged with the harvested energy. The voltage can also experiment progressive dropping if there is not energy available from the environment.
   The considered figure of merit is the SNR (signal-to-noise ratio). One consequence of voltage diminution is the reduction of the signal to noise ratio, degrading the signal conditioning circuitry and algorithms performance.
   Last but not the least, our SNR study is motivated by [**Luo, X. (2008)**] , where authors address the development of quantizers under energy constraints and the effects in the spatio-

temporal signal reconstruction at the fusion center. Optimum number of quantization bits for multiple collaborating sensor are analytically derived and evaluated.
Our work reveals tradeoffs between signal acquisition features and it has been also established the first attempt to build the theory for adaptive signal processing techniques (examples of acquisition features are sampling frequency, bit numbers for data representation, signal to noise ratio, voltage reference for the ADC and signal dynamic range).

1.4 Thesis organization and overview

The context in which the research is framed enables us to organize the thesis as follows:

Chapter 2: Wireless Sensors Node Current Consumption Behavior.

The wireless sensor node lifetime is correlated with the sensor node current usage profile and the associated effects on the battery voltage. From literature, it is possible to found sensor node current consumption models. Nonetheless, such models do not provide enough insight on how to implement reduced node current consumption applications.
This chapter offers a systematic approach to evaluate wireless sensor node current consumption for a given application so that to implement optimizations capable of enhancing battery lifetime. Results can be used for detecting node operating modes that could generate current consumption profiles responsible of battery depletion acceleration.
We implemented a prototype board and an experimental setup that enable to visualize wireless sensor node current consumption profiles and charge extracted from the battery. The system enables not only to gain insight on how the operating system and the user program manage hardware resources, but also to identify which hardware block creates current consumption bottlenecks and which can be subject of optimization for saving energy.
In addition to this, we selected several tests that represent tasks usually performed by wireless sensor nodes. Finally, an experiment for battery full-depletion time estimation has been conducted, in order to compare empirical results with theoretical battery lifetime estimations. It is shown that sensor sampling activity and signal processing algorithms are very energy-demanding tasks, similarly with respect to the radio communications protocols. Experimental results are presented and discussed.

Chapter 3: Lifetime Maximization in Wireless Sensors Deployments.

The node lifetime is defined as the time when the node works properly, before battery depletion. Distances between nodes determine the required radio output power level to ensure reliable connectivity among them, affecting then the node battery current consumption. If some node runs out, it could make unreliable network function.
For this, it is important to implement a deployment which assures longer node lifetime; however, this problem is a very constrained and non-lineal one.
In this chapter, we propose a heuristic algorithm for searching network deployments. The algorithm predicts number of nodes, nodes distances and nodes output power level in order to give a deployment that reaches maximum battery lifetime compliant with user specifications. The heuristic algorithm outperforms the genetic algorithms in the design procedure.

Chapter 4 : Digital Signal Processing on Wireless Sensors Nodes.

This chapter introduces implementation issues on digital signal processing.
The MICAz node (our reference hardware platform) is built with the microcontroller Atmega128

which features an integrated AD converter. In this chapter we present an overview of the main AD converter specifications, which are required to perform sensor sampling with a posteriori discrete-time processing stage.
As the signal processing algorithm performance (e.g. accuracy, speed and robustness) has dependence on its software implementation, we have also studied the data representation types available in TinyOS and how the program interacts with the hardware platform.
Measurements of computation operation time have been performed, i.e. the required time to perform multiplications and sums with different data representation that this software-hardware platform supports.
As case study we selected a $L-th$ band Hamming windowed filter, because such filter is representative of which are often used for implementing digital wavelets transforms (see summary of chapter 6). The filter is a high-pass band with sampling frequency $7.98\,kHz$ and it has been programmed into the node and tested with different input analog signals. Advantages and lacks of the operating system implemented on general-purpose microcontrollers are highlighted
We conclude that exist a clear separation between signal acquisition circuitry and signal processing techniques, which is unacceptable from the power consumption point of view.
In this scenario, novel approaches appears to be required.

Chapter 5: SNR Evaluation for Wireless Sensors Nodes and Embedded Systems.

This chapter proposes a novel methodology for evaluating the signal-to-noise ratio (SNR) of data acquisition systems in wireless sensors microsystems. This methodology enables designers to evaluate the SNR regardless of non-stationary sensor signals behavior (with unknown signal shapes) or battery voltage decrease due to the depletion.
Moreover, the proposed methodology provides criteria to modify signal features and data acquisition system parameters for achieving required SNR values (e.g. to assure numerical stability of the algorithms implemented into the embedded system).
Validation measurements and application examples are presented.

Chapter 6: Wavelets Assessment on Wireless Sensors Nodes: Case Study

For signal feature extraction techniques, we have studied the implementation of wavelets transforms, because they provide a methodology to represent a measured signal with components that show time and frequency signal features.
By using this methodology, the theory predicts that it is possible to better detect local features of measured signals, for instance, to detect where the signal contains sharp spikes and their associated time duration or to analyze signal frequency contents.
If we have to explore the frequency components of the signal and their time duration, we should use wavelets with not widespread spectrum. Unfortunately, these wavelets transform are not hardware amenable and the digital implementation is in general power demanding.
Nevertheless, there exist a very simple wavelets transform (Haar wavelets) whose digital implementation only requires few subtractions and additions, thus it has been selected as potential candidate to be implemented into wireless sensors nodes. Implementation issues are presented and discussed.
We have explored the detection capability *by thresholding* of the Haar wavelet. Transient signal identification *by thresholding* by means of the Haar wavelet could be very efficient from the current consumption point of view, however simulations have revealed the presence of enormous drawbacks.
Thus, to overcome such problems and to enhance the Haar wavelet detection capability, we proposed and analyzed a hardware system able to preprocess the input signal. It is defined as *pulse shaping*.

As final conclusion, the feasibility to establish wavelets transform into wireless sensor nodes is then discarded. Motivated by the previous discussion, we observed the arisen need to use other family of algorithms.

Chapter 7: Conclusions and Final Remarks

This chapter provides final ideas and remarks of the reported research activity, as well as it is discussed future issues to address that could have many implications in the field of low-voltage and low-current consumption embedded wireless sensor systems.

Chapter 8: Future trend: Bio-inspired Computation on Wireless Sensors Nodes.

In this chapter we offer an intriguing case study. A 1-neuron spiking neural network (SNN) is implemented into the wireless sensor node by using reduced code and minor computational efforts. The learning process is empirically performed with real data so that to detect user defined frequency changes in the input signal under analysis.
This approach focuses attention on the idea that the information could be encoded and contained in the transitions of the bits used for the data representation. Thus, we only use 4 bits for the sampled signal data representation.
Measurements are offered and discussed to demonstrate the approach feasibility. We have obtained important clues for further research activity in the field of signal processing for low-voltage and low-current consumption wireless sensor microsystems.

Finally, referring to interested readers, appendices highlight detailed issues which have been discarded from chapters with the only purpose of not to overwhelm the main text.

Chapter 2

Wireless Sensors Node Current Consumption Behavior

Chapter Summary

The wireless sensor node lifetime is correlated with the sensor node current usage profile and the associated effects on the battery voltage. From literature, it is possible to found sensor node current consumption models. Nonetheless, such models do not provide enough insight on how to implement reduced node current consumption applications. This work aims to implement methodologies to evaluate wireless sensor node current consumption for a given application so that to implement optimizations capable of enhancing battery lifetime. Results can be used for detecting node operating modes that could generate current consumption profiles responsible of battery depletion acceleration. We implemented a prototype board and an experimental setup that enable to visualize wireless sensor node current consumption profiles and charge extracted from the battery. In addition to this, we selected several tests that represent tasks usually performed by wireless sensor nodes. Finally, an experiment for battery full-depletion time estimation has been conducted, in order to compare empirical results with theoretical battery lifetime estimations. Experimental results are presented and discussed.

2.1 Introduction

Wireless Sensor Networks (WSN from herein) are composed by devices named embedded wireless sensor nodes which are supplied by a couple of batteries. Therefore, it is clear that being the quantity of charge limited, it is desirable to achieve the lifetime as longer as possible. On the other hand, recent works show how wireless sensor networks is an emerging technology which has started to be widely used for different applications due to the distributed sensing capability. The following review of the state of the art shows several examples.
In [**Lu, B. (2009)**] authors present an ad-hoc wireless sensor network for motor energy monitoring. The implemented system explodes the cooperative operation of the wireless sensor network and the distributed sensing feasibility and detailed discussion of the key challenges are performed. However, the node lifetime problem and firmware optimization methodologies are not discussed.
Moreover, in [**Gungor, Vehbi C. (2009)**] is presented a wireless sensor network for industrial automation applications. Technical aspects, standard and design principles, as well as advantages of using wireless sensor networks are remarked. Nevertheless, issues related to programming techniques and sensor node lifetime estimation are also omitted.

In [**Dondi, D. (2008)**] it is faced the problem of the node power consumption as the fundamental topic to be tackled in further wireless sensor applications. Authors proposed solar energy self-powered wireless sensor networks. Issues regarding to devices for energy storing and node energy consumption are discussed. In addition to this, in [**Agha, K. A. (2009)**] the need of reduced power consumption in order to maximizes battery autonomy in wireless sensor networks is highlighted.
Finally, another case study is offered in [**Ahn, H. (2009)**], where it is proposed a wireless sensor networks for pedestrian tracking and localization. The advantages with respect to other systems is remarked and despite the fact that algorithms and localization techniques are proposed, their

effects on the node current consumption have not been studied.

The problem we face can be formulated as follows: we need to determine when and the manner that the battery voltage reaches a threshold named V_{min} (see **Figure 2.1**) under which the wireless sensor node becomes inoperative, i.e. V_{min} is the minimum voltage power supply required for the sensor node operation (usually specified in datasheets, e.g. $2.7\,V$).
The time L is defined as node lifetime and our goal is to explore its dependence upon different sensor nodes operating modes and in which way it can be maximized. To do this, we have to know the node current consumption profile $i(t)$ and the equation that relates such node current consumption profile $i(t)$ with the battery voltage $V_B(t)$ as is depicted in **Figure 2.1**.

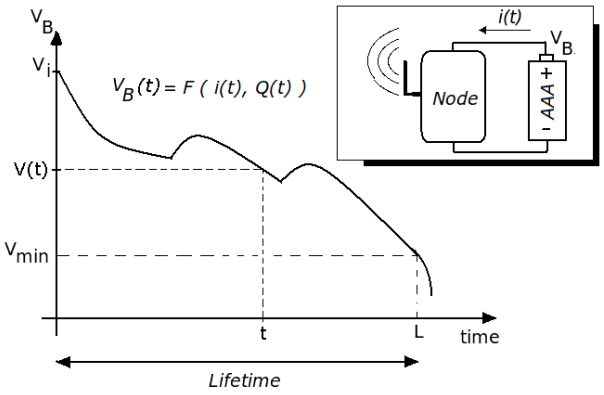


Figure 2.1: **Battery voltage profile. (Where $V_B(t)$ is the battery voltage, L is the lifetime, $i(t)$ is the current consumption profile and $Q(t)$ is the extracted charge)**

Therefore the following two issues deserve attention:

2.1.1 Node current consumption profile $i(t)$

The current usage profile is determined by the hardware platform features and the program that runs within the node, because it manages the hardware resources. There has been a lot of effort for building models capable of predicting embedded systems power consumption and current consumption, see for instance [**Saponara, S. (2007)**], [**Kan, B. (2007)**], [**Barberis, A. (2007)**] and also *Chapter 2-Single Node Architecture* in [**Holger, K. (2005)**].

The main problem related to the current consumption estimation is discussed by X. Jiang et. al. [**Jiang, X. (2007)**]. Authors evidence observations taken from real deployments, whereupon node's batteries often were depleted prematurely due to node activity originally not expected (i.e. unpredictable current consumption). Authors mention that for wireless sensor node applications it is not possible to assume a specific node current consumption profile a priori and the empirical validation of current consumption for every given application should be done.

Moreover, authors mentioned that static models for current and energy estimations could be inaccurate. Most of the static models presented in literature express the node current consumption or energy consumption in terms of equations with a large number of parameters, i.e. a defined state machine with the associated time duration and estimated energy consumption for each state and transition. See for instance in [**Kan, B. (2007)**] and [**Bougard, B. (2005)**] with reference to **Figure 2.2**, we have that the power consumption can be expressed as:

$$Total\ Energy_{nodo_i} = \sum_{state_j} P_{state_j}\, t_{state_j} + \sum E_{transitions} \tag{2.1}$$

On the other hand, for some hardware platforms, vendors provide lifetime calculators; they provide estimations of battery lifetime by using simplified node behavior models based

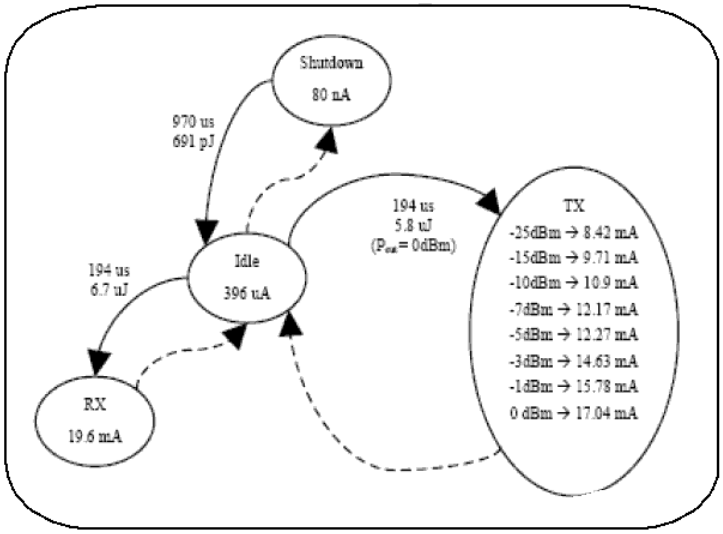


Figure 2.2: **Most used energy consumption models [Bougard, B. (2005)]. The index $state_j$ refers to the energy states of the radio: shutdown, idle, reception or transmission. In addition to this, P_{state_j} is the power consumed in each $state_j$ and t_{state_j} is the spent time in the corresponding state.**

on the node duty-cycle (the node duty cycle is defined as the ratio of active and not active node time intervals). The *Mote Battery Life Calculator from Crossbow Technology* [**MLC, Mote Battery Life Calculator**] is one of them (the user input data interface is depicted in **Figure 2.3**). Other examples regarding battery lifetime estimation are presented in [**Kyaw, Z.T. (2007)**] and [**AN053, App. Note**].

Despite the fact that the previous models and methodologies are theoretically correct, the main drawback lies in the fact that the required parameters have to be extracted from data not always available in datasheets. Moreover, the operating system effects on the power consumption are neglected or not modeled. It has enormous consequences on the accuracy to determine the duty cycle value required to implement the methodology proposed in [**MLC, Mote Battery Life Calculator**].

Node activity is controlled by the program that runs inside the microcontroller under supervision of the the operating system like TinyOS [**TinyOS**]. Most of the time, the operating system is asynchronously managing hardware resources in order to provide services to asynchronous events related to the wireless communication, (e.g. capturing input packets, channel assessment for transmissions among others) or sensors sampling requests. More specifically, an origin of this asynchronous behavior (among others) is given by the system used for channel assessment in most of MAC layer protocols (see *Chapter 5 MAC Protocols* in [**Holger, K. (2005)**]). It implements random waiting times slots for ready channel assessment. Therefore, generally speaking, the node is randomly awaking and sleeping and batteries are forced to deliver current patterns that are not easily predictable (see **Figure 2.4**).
The only way to overcome this problem is with accurate experimental hardware characterization and by assuming models of node behavior in order to perform node current consumption estimate.

We roughly mention that conversely as it is expected, we evidence in this work that node duty cycle is not a deterministic parameter, i.e. the duty cycle randomly varies in time. For this, proposed methods such as the mentioned in [**MLC, Mote Battery Life Calculator**], can only provide coarse node lifetime estimations.
Furthermore, proposed methods [**MLC, Mote Battery Life Calculator**], assume valid the current consumption superposition principle, however, it is not all true as we discuss in this work.

Expected battery life vs system current usage and duty cycle
(for two duty cycle models)

SYSTEM SPECIFICATIONS					
Currents			**Duty Cycles**		
	value	units	Model 1	Model 2	units
Micro Processor (Atmega128L)					
current (full operation)	6	ma	1	0.5	%
current sleep	8	ua	99	99.5	% Sum = 100%
Radio					
current in receive	8	ma	0.75	0.4	%
current xmit	12	ma	0.25	0.1	%
current sleep	2	ua	99	99.5	% Sum = 100%
Logger					
write	15	ma	0	0	%
read	4	ma	0	0	%
sleep	2	ua	100	100	% Sum = 100%
Sensor Board					
current (full operation)	5	ma	1	0.5	%
current sleep	5	ua	99	99.5	% Sum = 100%
Battery Specifcations					
Capacity Loss/Yr	3	%			

Computed mA-hr used each hour	**Model 1**	**Model 2**
uP	0.0679	0.0380
Radio	0.0920	0.0460
Flash Memory	0.0020	0.0020
Sensor Board	0.0550	0.0300
Total current(ma-hr) used	**0.2169**	**0.1159**

Computed battery life vs battery size	**Model 1**	**Model 2**
Battery Capacity (ma-hr)	Battery Life -months	Battery Life -months
250	1.58	2.95
500	3.16	5.91
1000	6.32	11.82
1500	9.48	17.73
2000	12.63	23.63
3000	18.95	35.45
	After Battery Capcacity Loss	After Battery Capcacity Loss
250	1.57	2.93
500	3.13	5.82
1000	6.22	11.47
1500	9.25	16.94
2000	12.24	22.24
3000	18.05	32.31

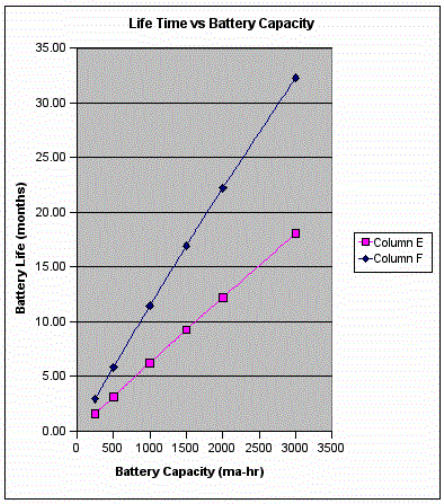


Figure 2.3: **[MLC, Mote Battery Life Calculator] User interface**

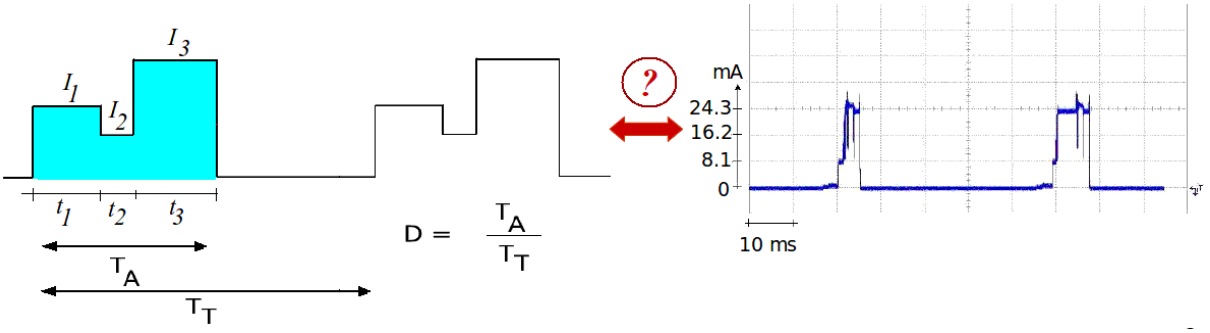


Figure 2.4: **Assumed current consumption profile (left) and a real one (right). There are a lot of open questions: how we can estimate time intervals, currents values, duty cycle ?. What about the current waveform, how we know that the staircase model is the best coarse approximation of the the real current usage profile ?. Are constant the required parameters, or they are time-varying during node activity ?.**

In addition to this, effect of sensors current consumption is often vanished in literature. It is argued that the radio is responsible for the battery depletion. Due to this, a lot of research has been oriented toward the design of efficient MAC protocols, reliable routing layers and algorithms for error-rate corrections due to unreliable radio links.

Only recently, new elements have been added to the list of causes of battery depletion acceleration in WSN scenarios. Data conditioning and acquisition systems have started

to be reported as a main source of power dissipation together with the RF front-end [**Dlugosz, R. (2007)**].
Furthermore, effects of the node current consumption for memory usage started to be underlined as another current consumption bottleneck [**Mathur, G. (2006)**].

2.1.2 Battery voltage behavior $V_B(t)$

Under the hypothesis that we can have a current consumption profile model, it remains open the problem of describing the battery voltage as function of such battery current discharge profile. Nevertheless, there is not provided in literature a closed treatment for analyzing the battery discharge phenomena. See for instance [**Rao, R. (2003)**], [**Rakhmatov, D.N. (2003)**], [**Rao, R. (2005)**], [**Panigrahi, D. (2001)**].

Most used of the proposed methodologies face the problem by calculating the charge Q, which is used by the node as function of its duty cycle. Once the charge Q has been estimated, the proposed methodology uses the battery capacity in order to estimate the battery lifetime for delivering the required charge Q at the given duty cycle.
However, the available charge that the battery can deliver differs from the nominal capacity specified in datasheets because it depends on the discharged current intensity; therefore, the battery capacity (or available charge) is dependent upon the node activity and it is an unknown variable within the lifetime estimation problem.
Moreover, the necessity of performing continuous monitoring of the battery voltage evolution in order to maximize node lifetime is discussed in [**Cho, Y. (2007)**], where authors propose adjusting the node frequency operation as function of the battery voltage variations. In addition to this, X. Jiang et. al [**Jiang, X. (2007)-2**] present a first customized hardware platform designed to implement in situ node power and energy consumption measurements. However, authors have not offered experimental results.

The need of using dedicated measurement systems to explore the node behavior becomes evident.

2.1.3 Chapter goal

The node lifetime estimation issue appears to be still open without a suitable methodology to be used as criterion to design WSN applications. In fact, nowadays, WSN designers cannot make accurate node lifetime predictions during the design stage.
Our target is then:

1. **We would like to understand what is happening:** there are evidence and observations taken from real deployments, whereupon node's batteries often were depleted prematurely with respect to the prediction due to node activity originally not expected.

2. **We would like to characterize the node current consumption:** models express the node current consumption in terms of equations with a large number of parameter. Such parameters have to be extracted from data not always available in datasheets and characterization is required.

3. **We would like to model current consumption features for new hardware:** data acquisition front ends have started to be reported as a main source of power dissipation together with the RF front-end. Node current consumption for memory usage started to be underlined as another possible current consumption bottleneck.

4. **We would like to optimize the application firmware:** for this, we need to visualize current consumption while the software is being built and modified (as an example during debug stages).

In this work we describe a methodology to face the described issues. Following a similar approach as it has been presented in [**Jiang, X. (2007)-2**] and [**Konstantakos, V. (2007)**], we present an electronic system based on a dedicated PCB. The board is a tool that enables to analyze the behavior of the node in real operating condition without jeopardizing the node lifetime when the board and node work together being powered by the same battery pack.
The measurement tool allows visualizing node current consumption usage and charge extracted from the battery during different node operating states; we highlight charge cost of some tasks usually performed in WSN scenarios.
The node under observation is the MICAz [**MICAz (Crossbow)**] that runs applications developed using nesC language [**Gay, P. (2003)**]. The TinyOS version is provided by the software platform MOTEWORK [**MOTEWORK Tool (Crossbow)**] and the radio transceiver is the Chipcon CC2420 [**CC2420 Radio Device**].

The scope of this work is to offer new highlights on how the application lifetime can be estimated as well as to present the designed board and its utility for enhancing the node longevity. Results of a full battery depletion experiment caused by the node activity are offered.

2.2 Experimental setup and methodology

The prototype PCB is based on the following devices: the High Side Current Shunt Monitor INA139 [**INA139 Datasheet**] and the Coulomb Counter LTC4150 [**LTC4150 Datasheet**]. In **Figure 2.5** the built board and the node under observation is depicted.

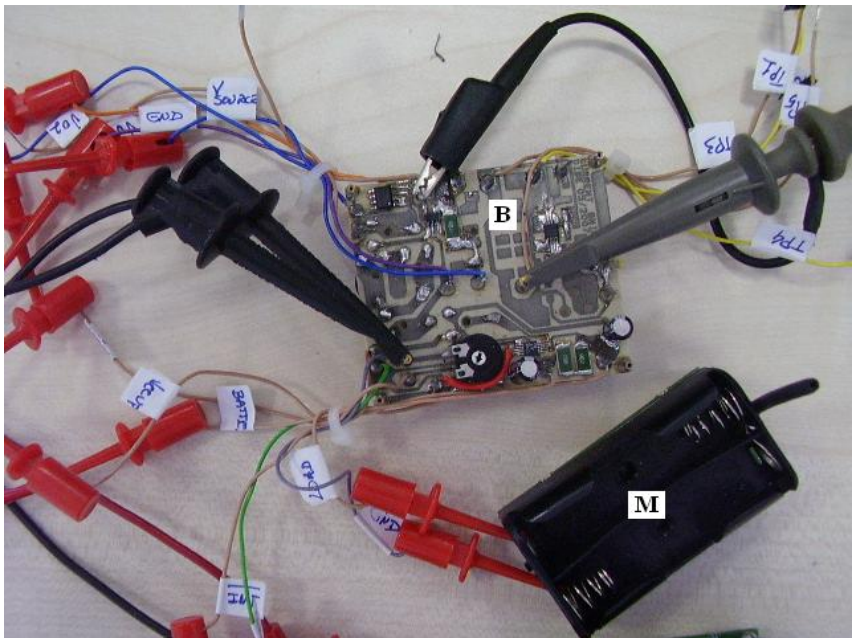


Figure 2.5: **Picture of the prototype board (B) and the MICAz node (M) (PCB layout and schematics are shown in appendix B).**

The INA139 amplifies the voltage across a shunt resistance between the battery and load. The measured voltage is proportional to the battery current consumption. The output voltage signal is connected to an amplification stage. Afterwards, the output is visualized in an oscilloscope, which provides a sampling rate of $4\,GSa/s$, and $1\,GHz$ of bandwidth. Consequently, the node current waveforms associated to the node operating state (such as transmission-reception states, sensor sampling and transceiver or microcontroller idle modes) can be analyzed.

We select the shunt resistance value for providing the maximum available INA139 output

voltage at the maximum node current consumption. In other words, the shunt resistance value has been selected so that the INA139 gain provides the maximum output range at the maximum node current consumption.
The wireless sensor node has the largest current consumption in the active mode (i.e. awake state). In such periods, it is normal to observe current consumption around $30 - 40\,mA$. Conversely, in sleep mode, the current consumption falls to around few microamperes.
The board is powered by a $3\,V$ voltage supply, which is independent of the node voltage supply. On the other hand, the charge counter characterizes the node in terms of consumed charge. In this case, the information is expressed in units of charge (mAh or μAh) flowed from the battery into the node. The LTC4150 device is mainly a voltage integrator with voltage-to-frequency converter. The output waveform consists of narrow pulses. The coulomb counter device forces to zero the output pin when a fixed quantity of charge Q_M has been measured. The value $Q_M = 5.2\mu Ah$ has been obtained by calibration. The time period between pulses is proportional to Q_M. **Figure 2.6** diagrams the experimental setup.

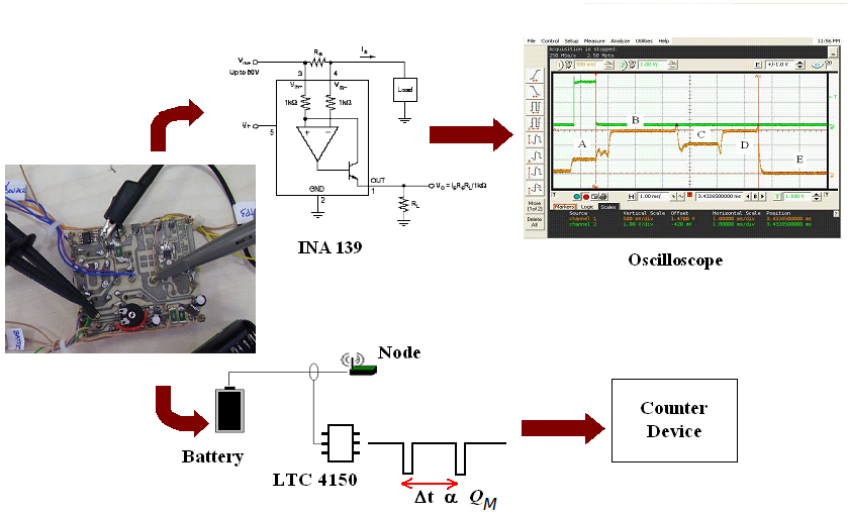


Figure 2.6: **Diagram block of the experimental setup. The board enables to visualize node current usage profile and charge extracted from the battery.**

The node's tasks under evaluation were sensors sampling, transmissions and signal processing. We assume that the node wakes-up in order to implement a task and afterwards, the node goes into sleep mode, having sleep mode negligible current consumption (interested readers can see in advance **Figure 2.14** that illustrates the process). Each task is executed at a rate R, expressed in tasks per second; we measured the charge used for implementing each task.

The procedure to get measurements from the coulomb counter device is an average counting process. If a number of K pulses is counted within a given time interval T (time observation window), then the amount of charge consumed by the load (i.e. the node from the battery) is given by $Q_T = K.Q_M$.
Finally, under the assumption that the node has executed N times the task under analysis during the time T, we calculate the average task charge cost Q_c as follows:

$$Q_c = \frac{Q_T}{N} = \frac{K\,Q_M}{N} \tag{2.2}$$

Where N is such that $N = RT$. The measurement ends when time T is reached and the task has been executed N times. In the worst case, the error due to (2.2) is given by (and because $\Delta K = 1$):

$$\Delta Q_c = \frac{\Delta K\,Q_M}{N} = \frac{Q_M}{N} \tag{2.3}$$

We use the following example to illustrate the methodology. If the node executes transmissions at the rate $R = 10\,Tx/s$ (transmissions per second), then, for a given time, e.g. $T = 3600\,s$ (one hour), the number of performed tasks (or number of packet transmissions) is $N = 36000$. If we count, for instance, $K = 1000$ pulses, by using (2.2), it is possible to calculate the charge cost associated to the transmission task and its associated error as follow:

$$Q_c = 1000\frac{5.2\mu Ah}{36000} = 144.4\,nAh \tag{2.4}$$

$$\Delta Q_c = \frac{5.2\mu Ah}{36000} = 144.4\,pAh \simeq 0.15\,nAh \tag{2.5}$$

The error becomes negligible for large N values. As a final remark, we mention that the best set $\{T, R\}$ to perform the experiment is such that can assure large N values, in order to meet the condition $\Delta Q_c << Q_c$.
The method of counting charge has the disadvantage that requires long time measurements. It depends on the selected R value and measurements could easily take several hours. A suitable T value (or N) is not known a priori, because the charge Q_c is obviously not known as well as the K value. This problem can be overcome with an iterative procedure, by performing several measurements. In general from our experience the longest measurement time which we have conducted was no more than 4 hours.

This method provides excellent accuracy for large N values, because we remark that is not necessary to assume a priori current consumption waveforms.
Moreover, the measurement takes into account the whole node activity. For instance, it takes into account the current consumption during hardware start-ups and node sleeping operating modes. It is frequently considered negligible in reduced node current consumption models, nevertheless, it could have accumulative effects during the whole time of node activity.

2.3 Experimental measurements.

Each task has been programmed using the TinyOS version as well as libraries provided by the software platform MOTEWORK; consequently we used the native HPL modules to manage the hardware platform (more specifically, the software versions are the following: ncc version 1.2.1, nescc version 1.2.8a and gcc version 3.3.3 cygwin special).

TinyOS, is based on the model of hardware abstraction layers (TEP 2 [**TinyOS**]), where each level gives services to upper layers through interfaces. The bottom layer is named Hardware Presentation Layer (HPL) (see TEP 2 in [**TinyOS**]) and it is responsible for accessing hardware resources.
By using the hardware abstraction layers architecture, designers can build their application programs using customized interfaces through which, lower hardware abstraction layers are invisible. In this way, the application is implemented at top level (Hardware Interface Layer or HIL), being somehow platform-independent and consequently reusable.

After providing a brief introduction about the software architecture, we present the tasks. The tasks represent applications usually implemented into wireless sensor nodes: sensor sampling (task 1), radio transmission (task 2), Exponentially Weighted Moving Average (EWMA) filter processing (task 3) [**Sangiacomo, F. (2007)**] and a FFT algorithm computation (task 4).
Task 1 involves the sampling of two sensors (temperature and light) at a given rate. Temperature and light sensors are provided by the board MTS300 [**MTS/MDA Board Datasheet**]. For each task we analyzed the behavior of the node through the current usage profile and the associated charge cost. Results are reported in **Table 2.1**.
It is evident that battery current consumption is primarily given by the FFT implementation followed by sensor sampling. For lifetime estimation, sensors current consumption should be seriously taken into account and we remark that this issue is frequently vanished in literature.

Table 2.1: **Energy and charge cost.**

Task	Charge Cost $Q_c(\mu Ah)$	Energy Cost (mJ) [1]
Task 1	0.520	5.616
Task 2	0.042	0.454
Task 3	0.005	0.054
Task 4	1.244 [2]	13.4

[1] Voltage power supply: $3\,V$
[2] Output Power: $0\,dBm$ (Maximum value for the radio CC2420). Payload size 25 bytes)

The energy required for computing the EWMA filter is the lowest.

On the other hand, few words related to the FFT algorithm deserve attention. In [**McIntire, D. (2006)**] the FFT performance implemented on MICA2 mote is reported; **Table 2.2** shows comparisons. Regardless of that the buffer size is twice than that of the work reported in [**McIntire, D. (2006)**], the execution time and energy consumption results show that our algorithm has poor performance.
In fact, the FFT algorithm has been implemented as a first prototype without taking into account optimization issues. Such algorithm has been provided by a company (see acknowledgements) and with the only purpose of testing current consumption. The difference between the results is mainly due to the fact that the algorithm presented in [**McIntire, D. (2006)**] has been optimally implemented in C language (using C compiler), instead of nesC language (using a nesC compiler).

Table 2.2: **Comparative EWMA and FFT computation results.**

	Node	Data size	Buffer size	Time (ms)	Energy (mJ)
FFT					
This work	MICAz	16-bits	256	552.73	13.4
[**McIntire, D. (2006)**]	MICA2	16-bits	128	15.4	0.934
EWMA					
This work	MICAz	32-bits	8	0.563	0.054

[1] MICA2 and MICAz nodes have the same microcontroller Atmega128L, working at the same clock frequency.

As we can see, compilers and programming implementation issues have strong influence on the node current consumption. Not withstanding the fact that programs build in nesC are based on customized interfaces and modules which provide modularity, encapsulation and re-usability features; such benefits are counterbalanced by high current consumption.

For achieving long node lifetime, the program implementation should be carefully optimized with meticulous measurements of the current consumption. **Metering boards such we present in this work becomes a powerful tool for enhancing WSN applications.**

2.4 Tasks results analysis

In this section, we present detailed analysis of current usage profiles for the task 1 (sensor sampling) and task 2 (radio transmission).

2.4.1 Task 1. transmission activity

Fist of all we analyze the radio transmission battery current consumption. In **Figure 2.7** we show the transmission current consumption profile where transmissions are taking place. It is

possible to identify the following intervals:

Interval A: Microcontroller wakes-up. The microcontroller starts to process commands to send a packet. While the microcontroller is working, the current consumption is $8mA$ according to the [**MICAz (Crossbow)**].

Interval B: Radio Transceiver wakes-up (oscillator start-up)

Interval C: The operating system manages the SPI bus for granting communication between the microcontroller and the radio. It is also implemented MAC layer functions, such as channel assessment and backoff-time calculations. In addition to this, a RX-calibration state takes place as it is indicated in the radio control state machine [**CC2420 Radio Device**].

Interval D: The transmission occurs. The current consumption is related to the output power level place the microcontroller current consumption (see **Table 2.3**).

Interval E: The radio transceiver is in reception mode according to the radio control state machine [**CC2420 Radio Device**]. Then, the node goes to Power Down state, waiting for the next timer interruption, which commands the next wake-up for the next transmission.

As we can see in **Figure 2.7**, the active period is $T_A = 6.44\,ms$.

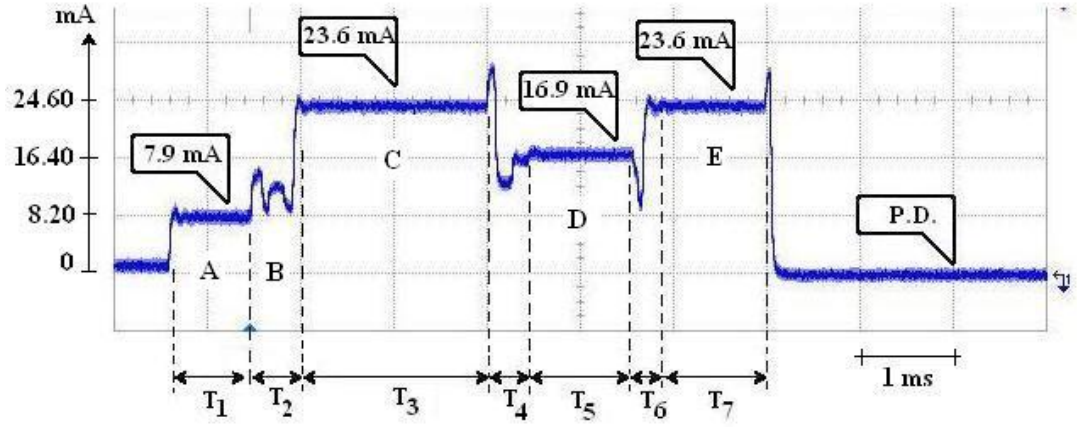


Figure 2.7: **Battery current waveform at $-25\,dBm$ output power. Time values are:** $T_1 = 0.84\,ms$, $T_2 = 0.58\,ms$, $T_3 = 2.00\,ms$, $T_4 = 0.49\,ms$, $T_5 = 1.1\,ms$, $T_6 = 0.36\,ms$, $T_7 = 1.07\,ms$. **The sum is the activity-time** $T_A = T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 = 6.44\,ms$.

Furthermore, **Table 2.3** shows that, to perform a packet transmission at a given output power level, the battery has to deliver the amount of charge indicated in the Charge Cost column.
The relative charge cost data show that selecting $0\,dBm$ as output power level, the battery has to deliver 1.083 times more charge that in the case of $-25\,dBm$ for implementing the same activity. Consequently, if the given application is working with a fixed duty-cycle value that at $0\,dBm$ reaches a lifetime of N days, at $-25\,dBm$ the node lifetime would be improved only 1.083 times, reaching 1.083 $N days$.
The improvement is quite negligible. The explanation could be obtained if we observe the current consumption curve during transmission in **Figure 2.7** and **Figure 2.8**. The current (and consequently charge) consumption actually is dominated by the current consumption during **intervals C and E**.

Table 2.3: **Battery charge and energy consumption for the output power levels available in the radio transceiver CC2420.**

Output Power Level	Current Consumption [1] ($m\,A$)	Charge Cost Q_c ($n\,A$)	Relative Charge Cost	Energy Cost ($m\,J$)
$0\,dBm$	17.4	41.9	1.083	0.453
$-1\,dBm$	16.5	41.4	1.070	0.447
$-3\,dBm$	15.2	40.8	1.054	0.441
$-5\,dBm$	13.9	40.3	1.041	0.435
$-7\,dBm$	12.5	39.7	1.026	0.429
$-10\,dBm$	11.2	39.2	1.013	0.423
$-15\,dBm$	9.9	39.1	1.010	0.422
$-25\,dBm$	8.5	38.7	1	0.418

[1] **[CC2420 Radio Device]**

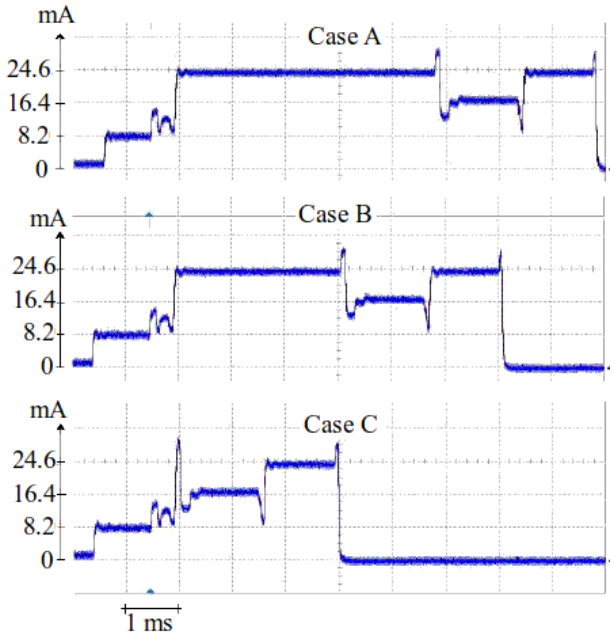


Figure 2.8: **Three different cases that show the time-varying battery current profile behaviour because of the random nature of backoff-time calculation for the MAC layer. Please, note how interval C (pre-transmission interval, see also Figure 2.7) varies.**

Variations in the current consumption level of **interval D** due to variations in the transmission output power level, actually do not modify significantly the whole cost. Finally, we can conclude that, with the given hardware/software platform, we could use the maximum output power for any application because it provides the highest radio coverage range at the expense of a negligible increment of charge cost; as it has been shown, output power levels only slightly modify node lifetime.

Moreover, according to **Figure 2.8**, we found that it is not possible to associate one fixed and predefined current consumption waveform during transmission.

Random values for backoff-time computation in compliance to the MAC protocol features are the origin of the non-deterministic behavior shown in **Figure 2.8**. Such issue makes the effort for having accurate current consumption models a very tough task. A statistical model looks to be the best approach.

For underlining the problem, we make numbers. If we assume for instance, that transmissions

occur at the rate of $R = 1Tx/s$ (transmissions per second), thus, the duty cycle varies as follow (and from **Figure 2.8**):

- For case A: $D \approx \frac{9.2\,ms}{1000\,ms}$ or 0.92%
- For case B: $D \approx \frac{7.5\,ms}{1000\,ms}$ or 0.75%
- For case C: $D \approx \frac{4.5\,ms}{1000\,ms}$ or 0.45%

The difference between duty cycles in case A and B is 48.9%.
Therefore, without a reliable determination of the duty cycle value, proposed methods such as [**MLC, Mote Battery Life Calculator**] can only give lifetime estimations very roughly. The coarse estimation is due to it is assumed static or non time-varying node activity cycles.

Finally, in **Figure 2.9**, it is shown the current consumption effects on the battery voltage. It is visualized directly with the oscilloscope probe in the battery terminals. The battery voltage is $2.21\,V$, and the node is near of becoming inoperative because it can work only with power voltage supplies above $2.1\,V$. We observe that the voltage suddenly falls due to the battery internal resistance. After finishing the transmission, the battery voltage returns to its previous value and the battery recovery effect starts.

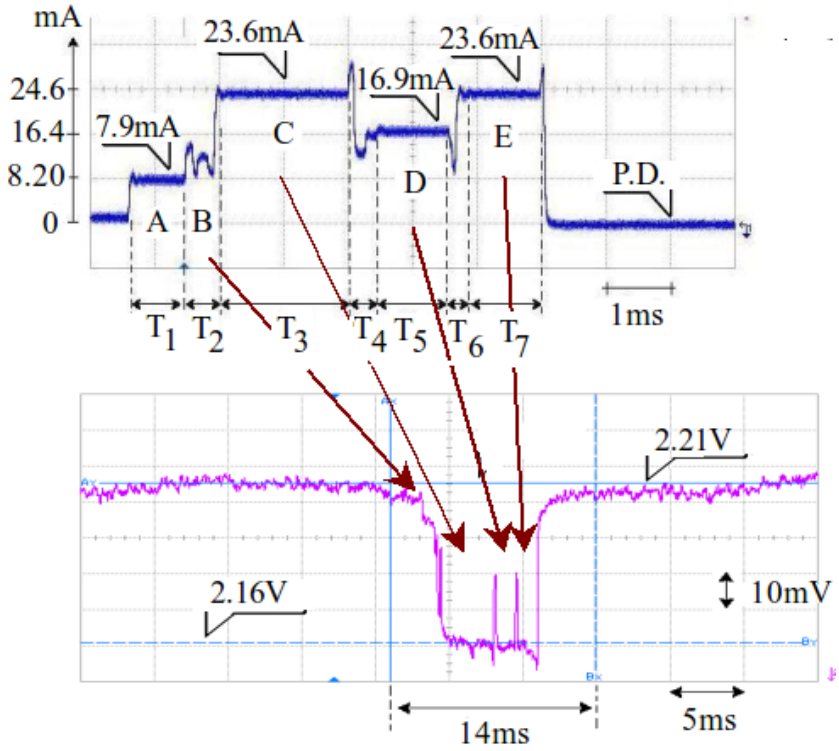


Figure 2.9: **Battery voltage variations during transmissions at** $-25dBm$.

2.4.2 Task 2. sensor sampling activity

The current consumption profile is drawn in **Figure 2.10**. The sensor sampling activity is programmed by using the provided MOTEWORK nesC modules. The only time that designers can set using the parametrized software interfaces is T_2. We selected $T_2 = 1\,s$.

We summarize each event marked in **Figure 2.10**, in order to explain the sensor sampling current consumption profile;

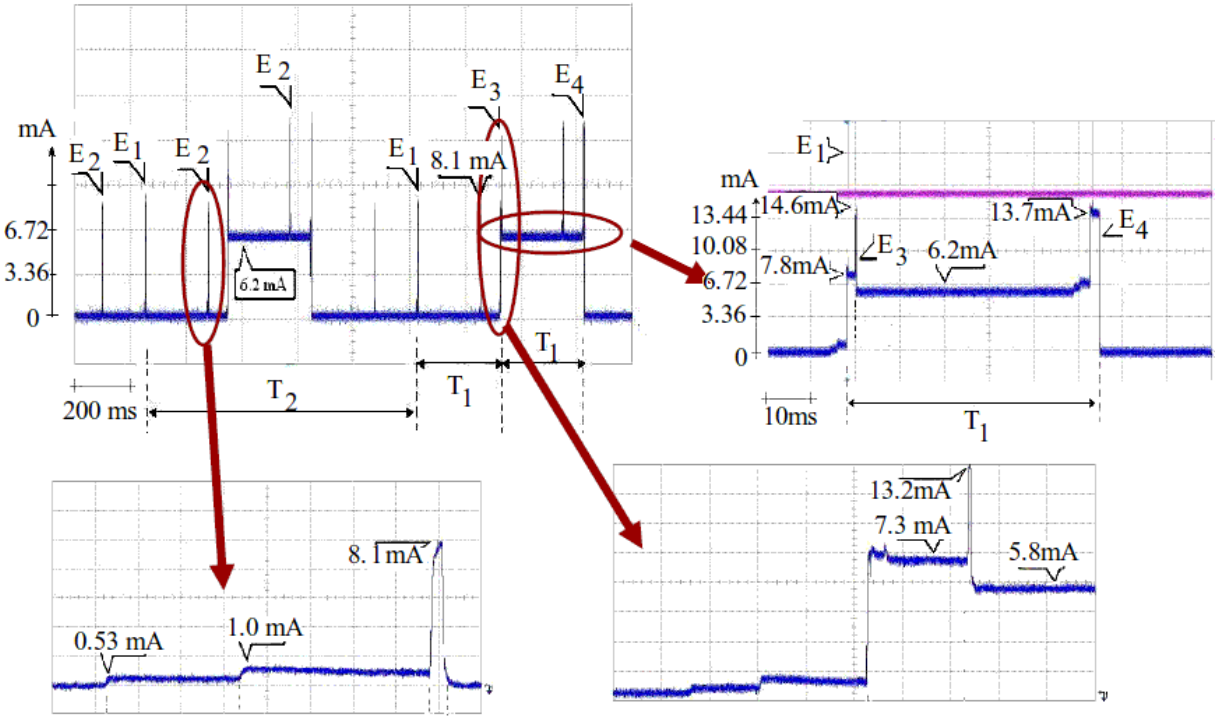


Figure 2.10: **Battery Current consumption profile for sensor sampling and zoom of the most relevant events. Time values are:** $T_1 = 300\,ms$ **and** $T_2 = 1\,s$**.**

E_1**:** The microcontroller wakes-up in order to perform sensor sampling. This request receives answer in E_3 and E_4. After performing sampling (light and temperature sensors) the microcontroller goes to sleep.

E_2**:** The TinyOs scheduler wakes-up the microcontroller and since the task-queue is empty, the microcontroller goes to sleep again.

E_3**:** The microcontroller wakes-up because of the time-out that was set in E_1. At this moment, the application has one light sample to process. The microcontroller goes to sleep but please, notice how the sensor board remains on, consuming $6.2\,mA$.

E_4**:** The microcontroller wakes-up once again because of the time-out that was set in E_3. At this moment, the application has one temperature sample to process. Afterwards, the whole system remains in sleep mode.

In **Figure 2.11**, it is depicted the current consumption profile and pieces of software code responsible for such behaviour. Analyzing the software module, which manages the sensor board, we found that with only few changes into the code, the charge cost could be reduced by almost 50% (time T_1 shown in **Figure 2.10** can be reduced). But, this optimization tip has to be implemented at low-level software modules or HPL (see TEP 2 in [**TinyOS**]). The new operating system version TinyOS 2.0.2 may solve this misbehavior. To verify this, further measurements have to be addressed. As a consequence, once again we highlight the necessity of current visualization tools for application optimization.

Finally, **Figure 2.12** shows the relationship between power and energy consumption for the

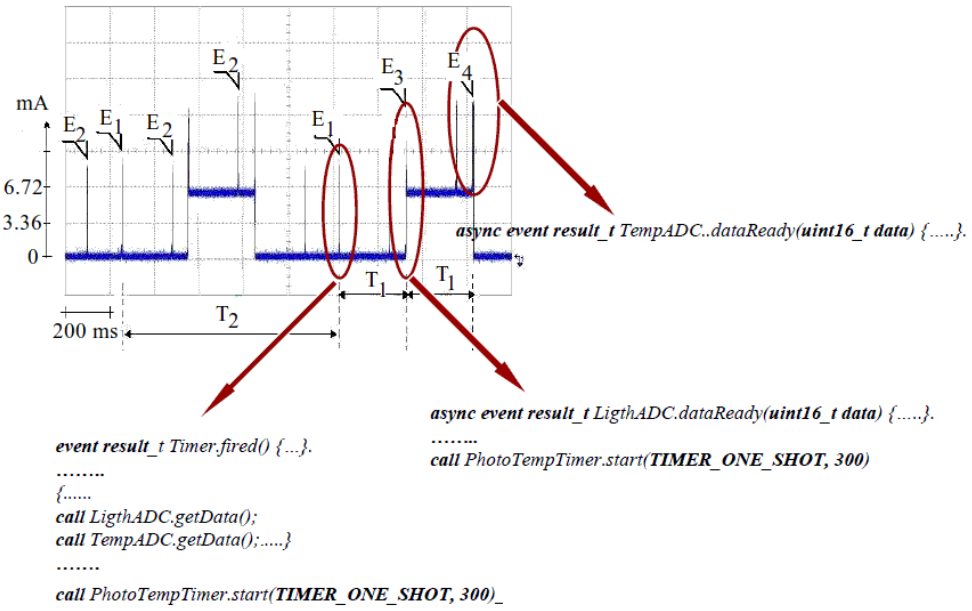


Figure 2.11: **Node Current consumption profile for sensor sampling and the main pieces of software code which create such current behavior.**

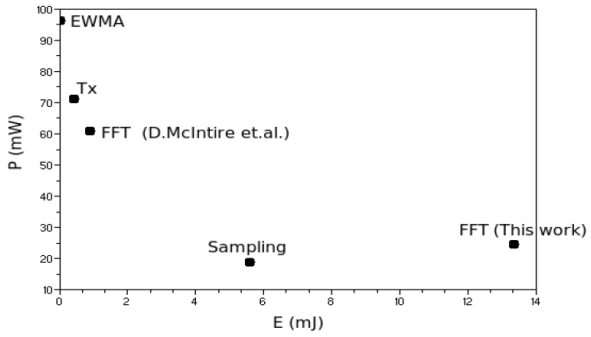


Figure 2.12: **Power-Energy Map for the analyzed tasks.**

analyzed test. For the case of transmission it is considered the case with output power level of $0\,dBm$ (marked T_x)

2.4.3 Non linear node current consumption effects

Table 2.1 shows charge costs for tasks working without correlation between them. Regardless of that,now we define the task 5 as sampling and transmission tasks working together, i.e. the node samples the sensor and then transmits the information.
The expected battery charge consumption cost should be equal to the sum of costs due to task 1 and task 2, because clearly task 5 is composed by task 1 and 2. To verify this assumption, we implemented the task 5 and experimental measurements have been performed. The result, presented in **Table 2.4**, is significantly different with respect to the expected one. As it is mentioned above (for the FFT case), it is evident that software implementation has a great impact on the node battery current consumption.
The behavior of the operating system and the way it manages hardware resources make unreliable the application of the superposition principle. From the charge consumption point of view, task 5 must be defined as a different task with respect to the activity performed in task 1 plus task

Table 2.4: **Task 5.**

Task	Charge Cost $Q_c(\mu\, Ah)$
Task 1+Task 2	$0.520 + 0.042 = 0.560$
Task 5	2

2. Despite the fact that task 5 and task 1 plus 2 perform the same final result (sampling and transmission).

We conclude that methodologies for lifetime estimation based on superposition (for instance shown in [**MLC, Mote Battery Life Calculator**]), should be carefully assessed. As regards node lifetime estimation, the approach must be holistic.

2.5 Battery lifetime estimation methodology.

2.5.1 Methodology

Battery models for depletion time estimation are available in literature. For instance, a noteworthy review of models are presented in [**Martin, T.L. (1999)**] and [**Rakhmatov, D.N. (2001)**]. Particularly, authors discuss the conditions and assumption on which the generalized Peukert's law (as is expressed by equation (2.6)), could be suitable for the battery lifetime estimation:

$$a = \left[\frac{1}{L_D}\int_0^{L_D} I(t)dt\right]^b L_D \tag{2.6}$$

The parameter L_D represents the battery depletion time and $I(t)$ the battery current. Moreover, $a > 0$ and $b > 1$ are constants that depend on battery features (i.e. type of chemical reactions and sizes). Both values have to be estimated by experimental battery characterization. However, according to [**Martin, T.L. (1999)**] for most of the batteries, b typically varies from 1.2 up to 1.7.

Nevertheless, in literature there is not consensus on how to use the equation and its accuracy is guaranteed in literature only for particular and controlled current discharge profiles.

The node lifetime L is defined as the time when the battery voltage reaches the voltage threshold V_{min}, below which the node cannot work properly (see **Figure 2.1**). The main drawback of equation (2.6) is that L_D is defined as the time when the battery becomes depleted, and a criteria to define *depleted battery* is when the battery voltage achieves values closed to the *depleted voltage* value provided in datasheets, for instance it is $0.8V$ for non-rechargeable batteries alkaline IEC-LR6 type, size AA, nominal voltage $1.5\,V$. Then $L_D \neq L$ in most of the cases.

The problem we face is then clear, in the case of sensor nodes, the V_{min} value could be different with respect to the *depleted voltage* used in equation (2.6). In fact, the sensor node under study is supplied by two non-rechargeable batteries alkaline IEC-LR6 type, size AA, with nominal capacity $C = 2000\,mAh$. According to **Table 2.5**, the node could be considered out-of-order when the battery voltage falls below $V_{min} = 2.7V$, i.e. when each battery voltage achieves a value of $1.35V$, instead of $0.8V$ (i.e. depleted voltage).

The node can become inoperative without depleting the battery and $L_D \neq L$.

Table 2.5: **Node voltage power supply characteristics.**

Device	Min.	Max.
CC2420	$2.1\,V$	$3.6\,V$
Atmega128L	$2.7\,V$	$5.5\,V$

For this, we propose a novel approach to determine L. It consists in using a simplified version

of the well known Nernst's equation, which quantifies the voltage created during electrochemical reactions as follows:

$$V_B(t) = V_B(t_o) + \lambda_1 \ln\left[\lambda_2 - Q(t)\right] + \lambda_3 \ln\left[\lambda_4 + Q(t)\right] \tag{2.7}$$

Where:

- $V_B(t)$: is the battery voltage at time t
- $V_B(t_o)$: is the initial battery voltage
- λ_1, λ_2, λ_3, λ_4 : are constants which must be determined by fitting experimental battery discharge characterization.
- $Q(t) = \int_{t_o}^{t} I(\tau) d\tau$: is the extracted charge at time t
- It is assumed a current profile $I(t)$ such that: $|\mathrm{I(t)}| < I_{max}$ and such that the battery does not experiment relaxation. For an alkaline battery, $I_{max} = 40\,mA$. Such value is estimated as follows: data sheets available from different vendors always show the curve *Capacity (C) vs. Discharge Rate ($I(t)$)*. On this graph, I_{max} determines the current interval values (from 0 up to I_{max}) within the following expression holds:

$$\frac{d\,C}{d\,I(t)} \simeq \texttt{constant} \tag{2.8}$$

We use this model to estimate L with the following methodology based on (2.7):

- The battery voltage variation during the node lifetime is $\Delta\,V_B = V(t_o) - V(L)$ (for a packet of 2 batteries, the node experiments a voltage variation of $2\,\Delta\,V_B$
- According to equation (2.7), the battery experiments $\Delta\,V_B$ when an amount of charge has been furnished, this charge is defined as Q_B. It is important to underline that each time that the battery delivers Q_B, the same $\Delta\,V_B$ is achieved (it depends on the integral of $I(t)$). Please, observe that is not necessary to perform estimations of the parameters in the equation (2.7).
- The charge Q_B related to $\Delta\,V_B$ is estimated by means of a controlled experimental battery characterization.
- We assume that the node periodically performs the same task at rate of R tasks per time-unit .
- Each task consumes an amount of charge Q_c (as it has been characterized in this work, see **Table 2.1**). The charge consumption rate is then $Q_T = Q_c\,R$
- Since equation (2.7) relates $\Delta\,V_B$ with Q_B independently of the discharge process, the node lifetime is estimated as: $L = \dfrac{2\,Q_B}{Q_T}$.
  The reason for the factor 2 lies in the following idea: the battery voltage variation is $0.25\,V$ for delivering Q_B, however the charge consumed for the node should be twice, since is powered by two batteries connected in series.

2.5.2 Case study and experimental results.

For our study case, the battery characterization is described in **Figure 2.13** for a non-rechargeable alkaline IEC-LR6, size AA, with nominal capacity $C = 2000\,mAh$. It has been performed by analyzing the battery voltage profile while is being depleted through a resistance $R_{LOAD} = 54\,\Omega$.
This value was selected in order to obtain current values, during the overall process, below $30\,mA$. This value is the upper bound of the maximum current value observed during transmissions, as it is shown in **Figure 2.14** and also meets the condition (2.8). The current across the resistance is proportional to the battery voltage, which goes down while depleting and the result is depicted in **Figure 2.13**.
Once the charge Q_B has been delivered, the battery voltage drops from 1.60 to $1.35\,V$ and consequently two batteries cannot provide the required voltage above $2.7\,V$ as power voltage supply.

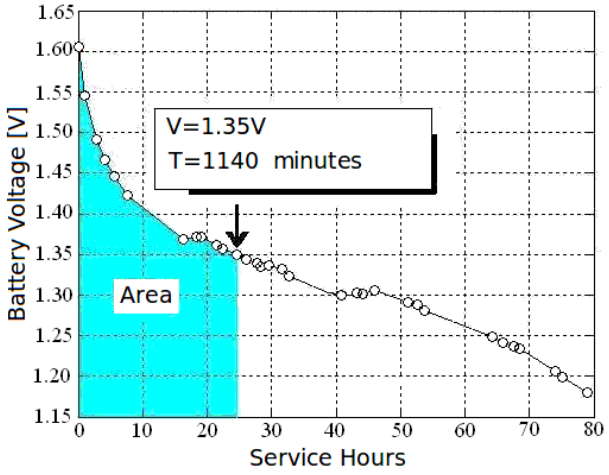


Figure 2.13: **Battery discharge characterization. Test Conditions:** $R = 54\,\Omega$ ($I \in [25.0 \div 29.6]\,mA$).

By numerical area integration, we can compute the charge Q_B that came out of the battery through the resistance until the battery voltage $1.35\,V$ is reached; $Q_B = 622.2\,mAh$ (see **Figure 2.13**) in $T = 1440$ minutes:

$$\int_0^T I(t)dt = Q_B = 622.2\ mAh \quad \texttt{for} \quad \Delta V_B = 0.25\,V \tag{2.9}$$

As a consequence, when the node voltage source reaches $2.7V$, each battery suffered $\Delta V_B = 0.25\,V$ and the node will have used $Q_{B,Total} = 2\,Q_B = 1244.4\,mAh$ of the $4000\,mAh$ available.
We reinforce the idea previously described. According to **Table 2.5**, the node could be considered out-of-order when the voltage power supply falls below $2.7\,V$. In this situation and starting with a couple of fresh batteries (the initial battery voltage is $1.6\,V$) it means that each battery experimented $\Delta V_B = 0.25\,V$ after furnishing $Q_B = 622.2\,mAh$ and the final battery voltage is $1.35\,V$.

According to **Table 2.1**, the charge cost for sending one packet at the output power level of $0\,dBm$ is $Q_c = 0.042\,\mu Ah$. Since the node has been configured for delivering 20 packets per seconds, the charge consumption rate is $Q_T = 0.84\mu Ah/s$ or $Q_T = 3.024mAh/h$.
Figure 2.14 shows the current usage profile in this case. If a zoom is implemented on a spike, we would face a current profile very similar to the one presented and analyzed in detail in **Figure 2.7**. However, the main difference lies in the time T_3 (it is the pre-transmission state time marked as interval C in **Figure 2.7**).

The expected lifetime is calculated as: $L = \frac{2\,Q_B}{Q_T} = \frac{1244.4\,mAh}{3.024 mAh/h} = 411$ hours, roughly 17 days.

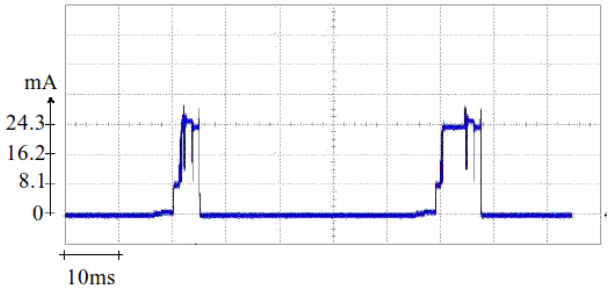


Figure 2.14: **Current usage waveform during transmission activity. Two current spikes corresponding to transmissions states are shown. The rest of the time, the node sleeps.**

The battery voltage as well as the transmitted packets have been monitored throughout the experiment with the CC2400DK Development Kit with the SmartRF Studio [**SmartRF Studio**] (see **Figure 2.15**).
The measurements have been done in order to verify that the node was working correctly during the experiment.

Time (us)	Length	Frame control field								Sequence number	Dest. PAN	Dest. Address	MAC payload	RSSI (dBm)	FCS
		Type	Sec	Pnd	Ack req	Intra PAN									
+241308													04 7D 9D		
=2196350	15	DATA	0	0	0	0				0x9D	0xFFFF	0xFFFF	41 01 00	-24	OK
Time (us)	Length	Frame control field								Sequence number	Dest. PAN	Dest. Address	MAC payload	RSSI (dBm)	FCS
		Type	Sec	Pnd	Ack req	Intra PAN									
+244155													04 7D 9E		
=2440505	15	DATA	0	0	0	0				0x9E	0xFFFF	0xFFFF	41 01 00	-24	OK
Time (us)	Length	Frame control field								Sequence number	Dest. PAN	Dest. Address	MAC payload	RSSI (dBm)	FCS
		Type	Sec	Pnd	Ack req	Intra PAN									
+246515													04 7D 9F		
=2687020	15	DATA	0	0	0	0				0x9F	0xFFFF	0xFFFF	41 01 00	-24	OK

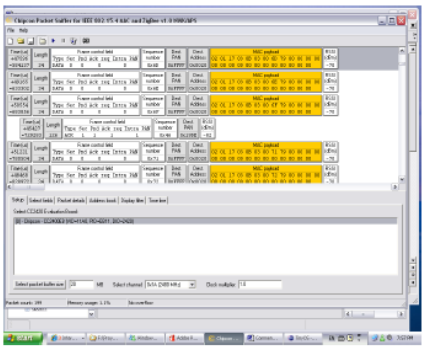


Figure 2.15: **SmartRF Studio [SmartRF Studio] user interface. In the Figure is depicted different packets captured by the tool. In this case, and with the purpose of illustration, we see a payload of 3 bytes.**

In **Figure 2.16** it is depicted the battery voltage behavior during the experiment with the node. This application had a lifetime of 15 days.
Please note that the given transmission rate value is high for usual real wireless sensor networks applications.The transmission rate value was selected to be able to perform the experiment in a reasonable time (i.e. some days i.e. no more than one month).
Despite the fact that batteries reached the threshold of $2.7\,V$ as indicated in **Table 2.5**, the node worked until day 35, when batteries voltage reached $2.1\,V$ (marked as B). This is the minimum required supply voltage value for the radio transceiver (as it is indicated in **Table 2.5**).

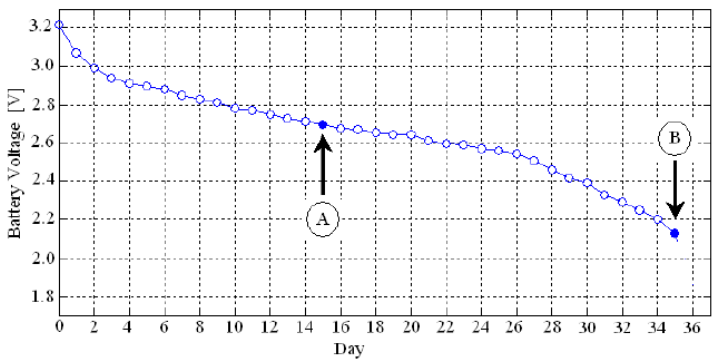


Figure 2.16: **Battery voltage during node activity in the battery depletion test.**

The experiment terminated when battery voltage reached $2.1\,V$ instead of $2.7\,V$. Despite the fact that this value is the minimum voltage supply required by the microcontroller, from this experiment it is quite clear that information taken from datasheets should be confronted with measurements. However, if we use the voltage $2.7\,V$ as voltage threshold, the lifetime estimation has 2 days of error in 15 days.

2.6 Final considerations

The main objective of this chapter was to present an electronic system based on a dedicated PCB to visualize node current consumption usage and charge extracted from the battery during node operating states.
The second objective was to make reliable battery lifetime estimation for wireless sensors network applications.

We selected tasks that represent usual WSN applications and node current consumption has been experimentally analyzed. Battery current consumption consequences for node lifetime have been clearly highlighted.
We conclude that first order models for node current consumption often used in literature (for instance in[**Kan, B. (2007)**]) can not model the measured behavior presented in **Table 2.3** and further research activity must be addressed to this aim.

Improved node current consumption models should take into account the whole activity of the node, more specifically; models have to take into account the effects of the hardware-software interaction. The radio transceiver is often reported as a main power dissipation source.
Regardless of that, we found that the processing and sensor data sampling are mainly responsible of battery depletion. Moreover, battery lifetime experiments have been performed. Specifically, the node has been daily under observation during one month and results show that battery lifetime estimation could be correctly addressed with the battery voltage model presented in this work.

2.6.1 Special acknowledgement

The authors wish to thank to Ing. Alessandro Balvis from SolTec IS3 S.r.l. ($www.soltec-is3.it$) who provided us the nesC module that implements the FFT algorithm used in this work.

Chapter 3

Lifetime Maximization in Wireless Sensors Deployments

Chapter Summary

Wireless Sensor Networks (WSNs) are composed by battery supplied nodes. Node lifetime is defined as the time when the node works properly, before battery depletion. Distances between nodes determine the required radio output power level to ensure reliable connectivity among them, affecting then the node battery current consumption. If some node runs out, it could make unreliable network function. For this, it is important to implement a deployment which assures longer node lifetime; however, this problem is a very constrained and non-lineal one. In this chapter, we propose a heuristic algorithm for searching network deployments. The algorithm predicts number of nodes, nodes distances and nodes output power level in order to give a deployment that reaches maximum battery lifetime compliant with user specifications. The heuristic algorithm outperforms the genetic algorithms in the design procedure and it is used real data obtained by the methodology presented in the previous chapter.

3.1 Introduction

Network node deployment has a great impact on the battery current consumption. The output power level has to guarantee reliable radio coverage for at least, the distance between nodes. Therefore, the battery current consumption is conditioned by how nodes are deployed, among other factors.

The growing complexity of these systems has brought new challenges for WSN designer and methodologies for efficient network deployment implementations are key issues to address. In fact, nowadays WSNs are widely used for monitoring environmental parameters as temperature, humidity, solar radiation and soil moisture among others [**Montepaldi FARM WSN TestBed**]. In this framework, application specifications and constraints make the node deployment a non-lineal and multi-objective optimization problem; regardless of that, there are applications which give insight on how the deployment can be settled.

For instance, one of the most used topology is the *in-line* array of nodes that build networks with grid appearance as it is shown in **Figure 3.1**.

Figure 3.1: **Farms: example where in-line deployments are present. This type of deployment frequently uses multihop protocols as it is depicted in Figure 3.2**

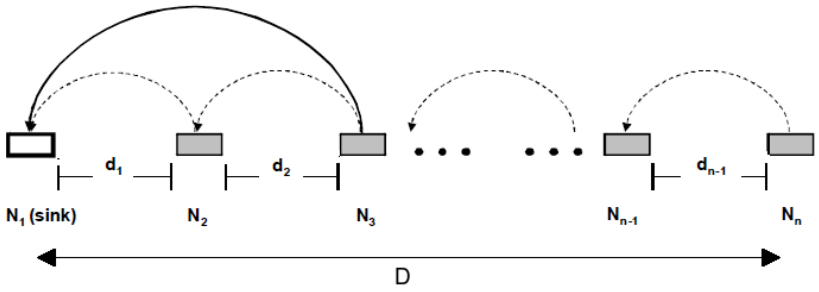


Figure 3.2: **In-line deployment. The multihop protocol is used and packets from far nodes are able to arrive to the sink.**

Among the different deployment implementations, the *in-line* deployment represents a good first candidate to investigate the energy usage minimization problem (which is also the lifetime maximization). Under certain hypothesis, there are expressions mathematically calculated (e.g. expression 3.1 from [**Shelby, Z. (2005)**]) so that it is phttp://www.google.com/ossible to make comparison between simulated (or empirical) results and theoretical results.
Moreover, the *in-line* deployment could be considered as a *building-net* for others more extensive deployments.

A study of the state of the art on nodes deployments has been performed and available literature such as [**Ferentinos, K. P. (2006)**], [**Xing, G. (2005)**] and [**Jiang, X. (2007)**] gives evidence that the energy usage minimization problem appears to be still open and there is not consensus about reliable methodologies.
Main weaknesses of published works are related to energy consumption models. These models depend on hardware models which are based on a large number of parameters not always available from datasheets. In addition to this, in [**Jiang, X. (2007)**] authors presented experimental observations from real world deployments, where nodes often run out prematurely, with respect to the predicted lifetime.
Authors claim that static current estimation models could be inaccurate and not suitable to use for node lifetime estimation. Authors argue that node current consumption depends on the application type and implementation features; for this, experimental hardware characterization should be performed and this must be taken into account for the deployment implementation.
Moreover, in literature it is often presented the unrealistic hypothesis of optimal power control; see for instance [**Shelby, Z. (2005)**], where the best deployment, i.e. the configuration of distances between nodes that assure the maximum lifetime is given by the following expression (and with reference **Figure 3.2**):

$$d_i = \frac{D}{(n-i)\sum_{p=1}^{n} p^{-1}} \tag{3.1}$$

(It is observed that the distance d_i gradually decreases in order to reduce the output power level at the antenna. The increment of traffic overwhelm the nodes near the sink, then such nodes have to transmit more packets by using reduced output power level in order to avoid the battery depletion caused by the large energy consumption.)
Conversely, the real situation is that commercial nodes have a discrete set of output power levels software selected (i.e. the antenna output power level is not a continuous variable inside the minimization problem). In addition to this, it is often claimed that the radio transceiver is the main responsible for battery depletion and most results presented in literature are based on this hypothesis.

However, in Chapter 2 it has been shown that node lifetime is mainly conditioned by sensors current consumption. Another issue is that proposed methodologies don't provide a way to incorporate energy usage profile models for sensors and because of this, the methodology scalability is vanished.
Moreover, the effects of the operating system and how it manages hardware resources have a great impact in the current consumption as we have studied in chapter 2, but until now it has not been taken into account.

3.1.1 Chapter goal

The main goal is to provide to designers an efficient tool to design real deployments with the only use of data from datasheets and few empirical hardware characterization as it has been done in chapter 2. The methodology enables designers to build a deployment that fulfills specification and constraints with longer lifetime.
The approach is supported by the methodology presented in chapter 2 and it enhances the state of the art based on the following:

- A heuristic algorithm for effectively exploring the deployment design space has been designed with good convergence capability. An efficient searching strategy is proposed. The heuristic algorithm has been designed after testing genetic algorithms without achieving relevant results.
- A concrete novel procedure is developed to handle the deployment design with real nodes. (All information is obtained from datasheets or from experimental hardware characterization and the algorithm is highly successful in meeting WSN designers needs).
- We added into the algorithm the real sensor current/charge consumption effects.
- We introduced a stochastic radio channel propagation model to increase robustness.
- A battery model is added inside the algorithm.

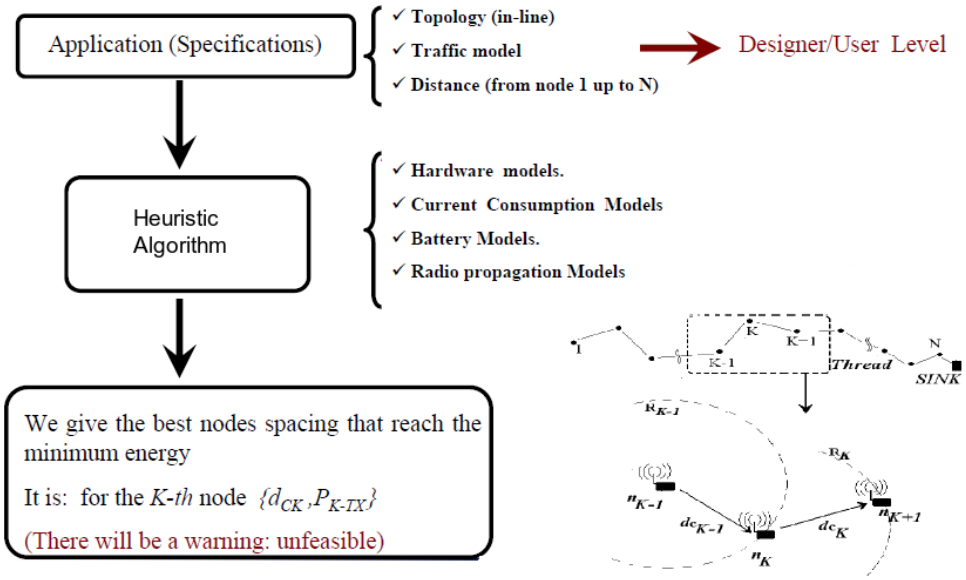


Figure 3.3: **Block diagram of the algorithm from the input-output point of view.**

3.2 Considered deployment and problem definition

The network deployment is considered to be an array of N nodes. The generic node n_K is defined from herein as the Kth node where K varies from 1 up to N (see **Figure 3.4**). The distance between n_K and n_{K+1} is dc_K

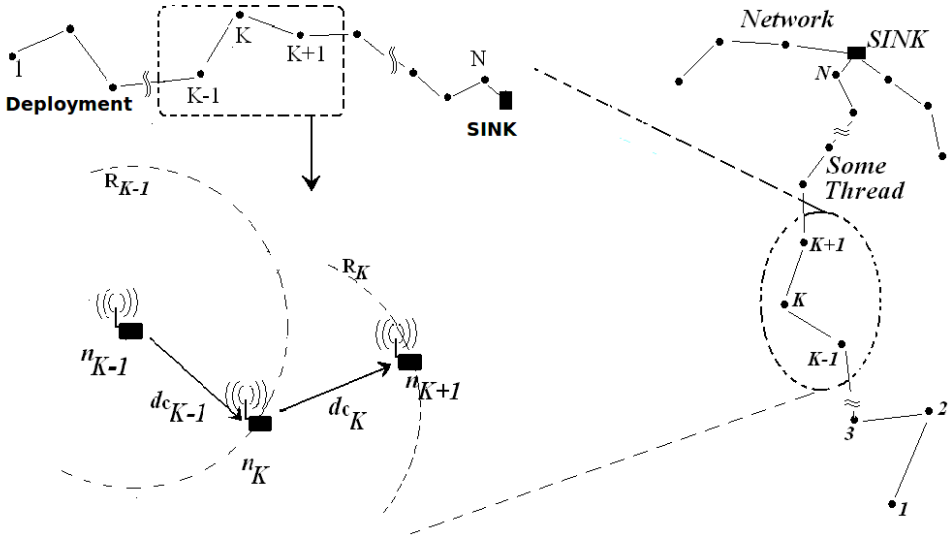


Figure 3.4: **Considered deployment topology and nodes n_{K-1}, n_K and n_{K+1}. The parameter R_{K-1} is the radio of coverage of node n_{K-1} (distance dc_{K-1}) and R_K is the radio of coverage of node n_K (distance dc_K).**

The node n_{N+1} is the sink. We assume that node n_K executes two types of tasks during its lifetime. One task consists in sensor sampling, data encapsulation and packet radio transmission. During the node lifetime defined as T_K, this task is executed M times.
The node is sleeping and periodically (the time period is named T_A) wakes up for performing the sensor sampling and information transmission. The value of M is correlated with T_K as follows:

$$T_K = M\,T_A \tag{3.2}$$

Another task is the multihop implementation. Packets from node n_{K-1} use the node n_K in order to reach the node n_{K+1} and so on, in order to reach the sink.
During the node lifetime, the number of packets transmitted from the node n_K to n_{K+1} is then equal to $K\,M$. We assume that the MAC is ideal i.e. a schedule system avoids packets collisions (and consequent wasted energy).
The deployment coverage range is D, whose value is given by user specification. It holds that:

$$D \leq \sum_{K=1}^{N} dc_K \tag{3.3}$$

Each node has λ possible transmission output power levels that can be software selected. The output power level P_{Tx} of node n_K can be selected from the set Ls that is defined as:

$$P_{Tx}^K \in [L_1, L_2, \cdots, L_\lambda] \tag{3.4}$$

For instance, in the case of [**CC2420 Radio Device**], $\lambda = 8$ and L_s is:

$$L_s = [-25, -15, -10, -7, -5, -3, -1, 0] \quad dBm \tag{3.5}$$

We define one individual of the deployment as the vector:

$$C = [J_1, J_2, \cdots, J_K, \cdots, J_N] \tag{3.6}$$

The individual of the deployment is one possible deployment topology among the large number of possible topologies .The vector has size N and it means that the deployment is made with N nodes. The value J_K is an index that can vary from 1 up to λ and it is a pointer to elements that belong to L_s. The output power level for node n_K is then expressed as:

$$P_{Tx}^{K} \in L_s(J_K) \tag{3.7}$$

Likewise that the index J_K has an associated output power level, it also has an associated radio coverage value dc_K. (Interested readers can see in advance how this relation is given by **Table 3.1**).
The following example is used to gain insight.
We consider $D = 10\,m$ and the radio CC2420 parameters. One deployment individual could be:

$$C_1 = [4, 7, 6] \tag{3.8}$$

This means that C_1 has 3 nodes. Node 1 uses the output power level 4 ($-7\,dBm$ from (3.5) and radio coverage $dc_1 = 3.1\,m$ from **Table 3.1**). Similarly, node 2 uses $-1\,dBm$ with radio coverage $dc_2 = 4.4\,m$ and node 3 uses $-3\,dBm$ with radio coverage $dc_3 = 4.3\,m$. The coverage range is $D = 11.8\,m$.
Readers can verify that other individuals can be:

$$C_2 = [4, 1, 7, 4]\,; \quad C_3 = [8, 1, 7, 1]\,; \quad C_4 = [8, 8]\,; \quad C_5 = [8, 7, 6]\,; \tag{3.9}$$

The vector C has variable size N. Please, note that there could be a lot of different individuals with the same size N. However, there is a maximum size for that is N_{max}. It is given by the minimum integer value that mets:

$$dc_{min}.N_{max} \geq D \tag{3.10}$$

The distance dc_{min} is the minimum coverage distance that could be found using the lowest output power level from Ls, i.e. L_1.

How many individuals we can find ?. The generic individual C_i, (with $i = 1, 2, 3, \cdots, Nc$) belongs to the set of N_c possible deployments compliant with specifications. This number N_c of different suitable deployments can be calculated as follows:

$$N_c = \lambda^{N_{max}} + \sum_{h=1}^{N_{max}-1} \lambda^{N_{max}-h} \tag{3.11}$$

That can be reduced as follows:

$$N_c = \lambda \frac{\lambda^{N_{max}}}{\lambda - 1} \tag{3.12}$$

e.g. taking into account (3.5) where $\lambda = 8$ and if we assume, for instance, $D = 50\,m$ then from (3.10) it results $N_{max} = 100$; therefore we have that: $N_c = 2.33 \times 10^{90}$ .This value is huge and is quite clear that the problem cannot be solved by analyzing all possible individuals. An algorithm for searching the network deployment has to be implemented.
Moreover, the n_K radio transceiver current consumption during transmission in the generic individual C_i is defined as:

$$I_{Tx,K} = I_{Tx,s}(J_K) \tag{3.13}$$

Where the vector $I_{Tx,s}$ is the associated current consumption for each output power level during transmission modes.

$$I_{Tx,K} \in I_{Tx,s} = [I_{Tx}(L_1), I_{Tx}(L_2), \cdots, I_{Tx}(L_\lambda)] \tag{3.14}$$

For the radio transceiver (as case study) we use [**CC2420 Radio Device**] where it results $\lambda = 8$ and $I_{Tx,s}$ is given by:

$$I_{Tx,s} = [8.5, 9.9, 11.213.9, 15.2, 16.5, 17.4] \quad mA \tag{3.15}$$

The whole amount of consumed charge during the n_K lifetime is:

$$Q_{A,K} = M\ (Q_{Tx,K} + Q_s) \tag{3.16}$$

Where: $Q_{Tx,K}$, is the charge used by node n_K for transmitting packets and Q_S is the charge used for sensing and processing. The charge $Q_{Tx,K}$ can be estimated as follows:

$$Q_{Tx,K} = I_{Tx,K}\ \tau \tag{3.17}$$

Where: τ is the transmitted packet time length. The Q_S value has to be experimentally measured (see **Table 3.2**). Using (3.2), expression (3.16) becomes:

$$Q_{A,K} = (I_{Tx,K}\ \tau + Q_s)\ \frac{T_K}{T_A} \tag{3.18}$$

In addition to this, each node executes packet re-transmission in order to implement the multihop protocol (see **Figure 3.5**).
The charge drawn from batteries to perform retransmission during n_K node lifetime is:

$$Q_{MH,K} = (K-1)\,(I_{Rx} + I_{Tx,K})\,\frac{T_K}{T_A}\,\tau \tag{3.19}$$

Where I_{Rx} is the current consumption in radio reception mode.

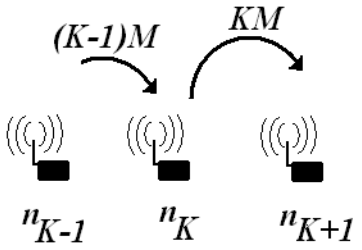


Figure 3.5: **Multihop. Node n_k transmits packets from n_{k-1} in order to reach the sink.**

3.3 Battery current consumption model

Each node in the network has its own energy source, i.e. a set of batteries; hence, the prediction about battery lifetime must be used in the deployment searching process.
The battery charge or capacity can be computed as expression (3.20), where C_{k-1} is the capacity before the time interval $[t_{k-1}, t_k]$ with $\Delta\ t_k = t_k - t_{k-1}$, moreover, $I(t)$ is the battery current and L is the battery charge leakage factor per time unit. Batteries datasheets quantify L by defining either the self-life time (named S_L) or battery charge losses per unit time.

$$C_k = C_{k-1} - \int_{\Delta\ t_k} I(t)dt - L\Delta\ t_k \tag{3.20}$$

Please, note that this model is widely used for node lifetime estimation. This is due to the fact that current consumption ranges available in commercial nodes makes negligible battery discharge rate dependency [**MLC, Mote Battery Life Calculator**],[**OMNeT++**].
The node current workload consists of periodic peaks between long idle periods with negligible current consumption. If during the node lifetime T_K the node performs periodically the same task M times, the following condition is then satisfied:

$$MQ_K + LT_K \leq C_{Batt} \tag{3.21}$$

Where Q_K is the charge cost (in mAh) of the task performed during time Δt_k and C_{Batt} is the initial battery capacity.

$$Q_K = \int_{\Delta\, t_k} I(t)dt \tag{3.22}$$

Finally, the relationship between node activity and battery capacity is as follows: the node n_K, is active while the following condition is held:

$$Q_{A,K} + Q_{MH,K} + LT_K \leq C_{Batt} \tag{3.23}$$

Hence, for the node n_K, its lifetime can be expressed as:

$$T_K = \frac{C_{Batt}T_A}{(K-1)\left(I_{Rx} + I_{Tx,K}\right)\tau + \left(I_{Tx,K}\,\tau + Q_s\right) + LT_A} \tag{3.24}$$

This lifetime is the cost function to be maximized as we soon explain during the algorithm description.

3.4 Radio propagation model

One of the most used radio propagation models in WSN applications is the Log-Normal Shadowing [**Zuniga, M. (2004)**] where radio channel loss is modeled as follows:

$$L(d)(dB) = L_o + 10n\log_{10}\left(\frac{d}{d_o}\right) + X_\sigma \tag{3.25}$$

The received signal strength can be expressed in dB as:

$$P_{Rx} = P_{Tx} - L(d) \tag{3.26}$$

The power P_{RX} represents the expected signal strength (in dBm) at the receiver placed at distance d (in meters) from the transmitter which delivers P_{TX} as output power (in dBm). The channel attenuation $L(d)$ (in dB) is given at the distance d, L_o is the attenuation (in dBm) at the distance d_o, n is the path loss exponent and finally X is a zero mean Gaussian random variable with variance σ_X^2 (in dB).
We have performed measurements in order to verify the model reliability. A measured example is offered in **Figure 3.6**.
This model has been extensively studied in literature and it is often used to project radio links, however, the main drawback lies in the large quantity of statistical data as well as accurate experimental arrangements required to estimate the statistical parameters.
As interest in our case, it is possible to overcome the problem through another approach for the radio channel propagation models which is based on the parameter named packet reception rate or also known as p_Q (it means the probability to have a reliable link, in terms of packets successfully received).
To highlight the packet reception rate modeling richness, we mention that the empirical radio

channel characterization can be performed by using two nodes instead of an elaborated and expensive experimental setup (interested readers can see in advance **Figure 3.7** which shows a performed characterization).

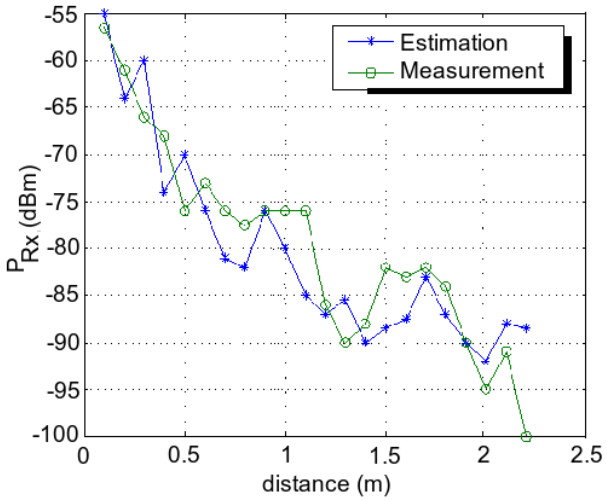


Figure 3.6: **The estimation is done by means of the expressions (3.25) and (3.26).The used output power level was $P_{out} = -25\,dBm$ and at $d_o = 0.1\,m$ the measured attenuation was $55dB$. Hence, $L(do)$ was set $30\,dB$. It also used $n = 2.5$ and $\sigma_X^2 = 4$. The input power level (RSSI) at the reception has been measured by using the tool [SmartRF Studio].**

The signal is received or incoming packets are successfully received if condition (3.27) is verified, when the Signal-to-Noise ratio (SNR) must be grater than a given threshold SNR_{min}:

$$SNR = P_{Rx} - P_{noise\ \ floor} \geq SNR_{min} \tag{3.27}$$

It is possible to show that the probability p_Q of successfully packet reception is given by:

$$p_Q = 1 - \frac{1}{2}\texttt{erfc}\left(\frac{P_{Tx} - L_o + 10n\log_{10}\left(\frac{d}{d_o}\right) - SNR_{min} - P_{noise\ \ floor}}{\sqrt{2}\sigma_x}\right) \tag{3.28}$$

The relationship between distance and radio output power is then given by the probability p_Q to have a reliable link, as it is expressed in (3.28).
It is assumed that the fully connected region [**Zuniga, M. (2004)**], is bounded by the distance d_c where $p_Q \geq 0.9$ i.e. $p_Q < 0.9$ or $\forall d$ such that $d \geq d_c$ means an unreliable link. A measured example of the p_Q (or PRR) behavior conducted for this work is depicted in **Figure 3.7**.
The main shortcoming in (3.28) is the presence of parameters that depend on the environment. For this reason, radio channel characterization has been conducted in the outdoor environment where the network has to be deployed [**Boero, F. (2007)**].
Results are shown in **Table 3.1**, where: $P_{Tx}\,(dBm)$ is the output power level of the radio transceiver CC2420 [**CC2420 Radio Device**] and $d_c(m)$ is the boundary or fully connected region distance, inside which $p_Q \geq 0.9$.

Table 3.1: **Radio channel characterization. Output power level and maximum coverage distance d_c.**

$P_{Tx}\ (dBm)$	-25	-15	-10	-7	-5	-3	-1	0
$d_c\ (m)$	0.5	1.7	1.9	3.1	3.9	4.3	4.4	5.0

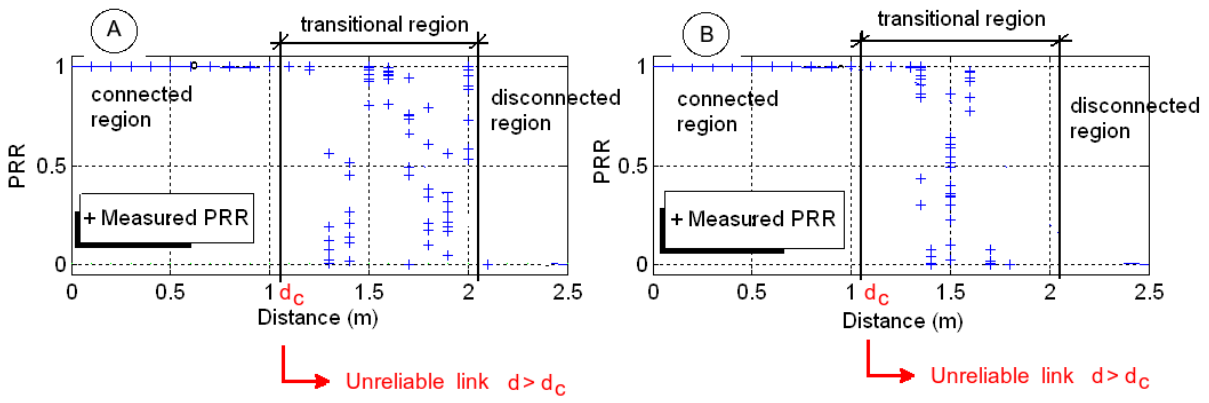


Figure 3.7: **Example of radio link characterization by using the PRR (or p_Q). In both figures are depicted the measured PRR vs. distance for the MICAz node. The measurements have been performed in two different days (case A and B). Features: output power level $-25dBm$, $2.4GHz$, indoor environment. The relationship between distance and T_x output power is then given by the probability p_Q to have a reliable link. The fully connected region is bounded by the distance d_c where $p_Q \geq 0.9$ (or $PRR \geq 0.9$)**

It is important to underline that height between node and ground has a great impact on the radio coverage. At the frequency of $2.4GHz$ Fresnel zones must be taken into account. It is important to underline that all measurements (**Table 3.1**) have been performed with nodes placed at $5\,cm$ above ground and we use such information in the algorithm.
Our algorithm is easily scalable and the height on ground for each node can be added in the optimization process if the need arises.

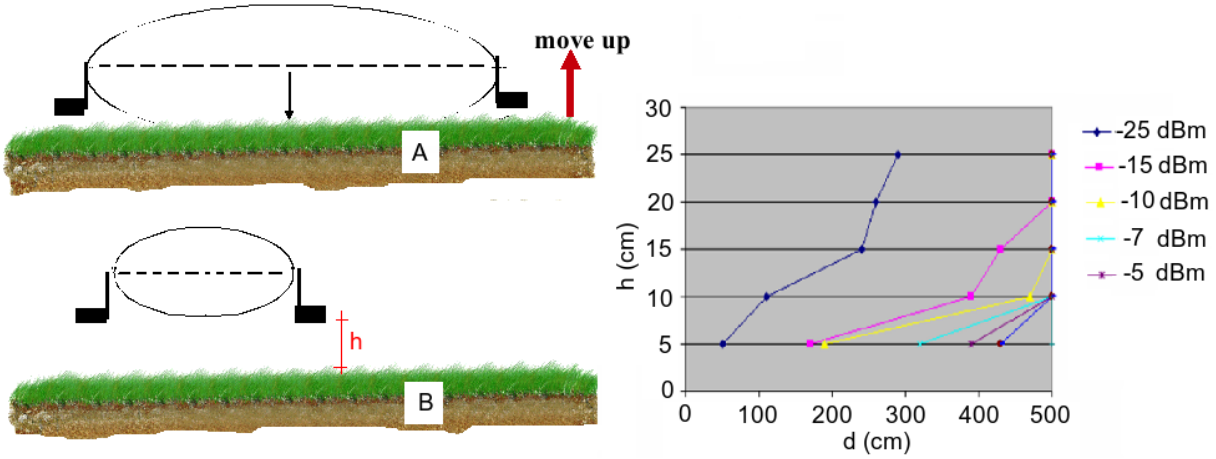


Figure 3.8: **The d_c value such that, for a given P_{out} (output power level) the $p_Q = 0.9$ depends on h. Measurements from [Boero, F. (2007)]. It is shown that for higher h values and at the same output power level, the radio coverage d_c increases.**

Instead of previous works, we argue that experimental radio channel characterization is a very important issue and for this, we use experimental measurement in the algorithm.

3.5 Optimization algorithm

We present the summary of variables and parameters as well as the algorithm goal and implementation.

- **Variables or output algorithm data:**

$$\left[N, \{d_{c,k}, P_{Tx,K}\}_{K=1\cdots N}\right] \tag{3.29}$$

The design goal consists in the computation of the number of nodes N, the deployment distances set $\{dc_K\}_{K=1\cdots N}$ and the output power set $\{P_{T_x,K}\}_{K=1\cdots N}$. The calculated deployment assures to be compliant with the given user specifications but meeting the largest deployment lifetime.
Giving a generic deployment C_i, each node n_K lifetime is named T_K. The deployment lifetime is defined as the minimum node lifetime.
The distance between nodes is an algorithm output variable constrained by the expression:

$$D \leq \sum_{K=1}^{N} d_{c,K} \qquad d_{c,K} \leq d_{max} \quad \forall\, K = 1, 2, \cdots, N \tag{3.30}$$

(environmental parameters often have to be accurately measured within a given spatial resolution).

- **User specifications (known parameters):**

$$[D, T_A, d_{max}] \tag{3.31}$$

- **Hardware-software features data (known parameters):**

$$[I_{Tx,S}, I_{Rx}, L_s, \tau, Q_s, \lambda, S_L, C_{Batt}] \tag{3.32}$$

Algorithm 1 describes the algorithm procedure and its main components. The algorithm implements a heuristic searching to find the deployment. It is mainly based on the random generation of deployments and lifetime evaluation. Once the deployment is evaluated, the algorithm modifies the deployment generation procedure accordingly with previous results.
The algorithm runs until a given number of iterations are executed. This number in the **algorithm 1** is named Max_Iterations. Below, and with reference to the **algorithm 1**, we highlight the main features.

- **Step 1:** The algorithm computes the first deployment C_{init}. The generation of suitable deployments or individuals to be evaluated is a task that has direct impact on the convergence speed.
The deployment generation is made by the function named Deployment_Generator. The first calculated deployment is taken as the best (C_{best}) as well as its lifetime (L_{best}).

  Description of the function Deployment_Generator **:**

  The function works as follows: first of all, the function creates three intervals named from herein U_1, U_2 and U_3. The creation of U_1, U_2 and U_3 is implemented with an uniform probability density function (PDF).
  By using uniform probability, the function randomly select two numbers; q_1 and q_2 within the interval [0,1] (the lowest value is named q_1).
  Therefore, intervals are defined as follows: $U_1 = [0, q_1]$, $U_2 = [q_1, q_2]$ and $U_3 = [q_2, 1]$. After this, the function selects a value for the random variable X by using an uniform PDF over the interval $[0, 1]$. Then, the selected value X belongs to a specific interval U_1, U_2 or U_3.
  According to this, the function takes the decision of which PDF must be used for extracting

output power values from L_s. If X belongs to U_1, the function randomly takes out values from L_s by using a designed PDF which generally favours higher output power levels (see **Figure 3.9**).
Equivalently, if X belongs to U_2, the function randomly takes out values from L_s by using another designed PDF which generally favours lower output power levels. Finally, in a similar manner to which described above, if X belongs to U_3 the function uses a uniform PDF, hence; each output power level has the same probability to be selected.

Algorithm 1 Algorithm Summary

{step 1}
Initialization (C_{init} is created)

`call Deployment Generator`
$C_{best} = C_{init}$; $C_{old} = C_{init}$;
$L_{best} = L_{init}$; $L_{old} = L_{init}$;
$Count = 0$;

Main loop
while $Count \leq Max\_Iterations$ **do**
{step 2}
`call Deployment Generator` (it generates a new deployment set: C_{new})
`it is calculated` C_{new} `lifetime:` L_{new}

{step 3}
if $L_{old} < L_{new}$ **then**
$C_{old} = C_{new}$
$L_{old} = L_{new}$

if $L_{best} < L_{new}$ **then**
$C_{best} = C_{new}$
$L_{best} = L_{new}$

`Modify size of` U_1, U_2 `and` U_3
$Count = Count + 1$;
{step 4}
Show results
Algorithm end

{**Function Deployment Generator**}

`Select a random value for` X
if $X \in U_1$ **then**
`use` PDF_1
if $X \in U_2$ **then**
`use` PDF_2
otherwise `use` PDF_3
`Calculate the individual C`
=0

Every time that an output power level is selected from L_s, the coverage distance $d_{c,K}$ is estimated by means of a function that implements the relation given by **Table 3.1**.
The process is repeated N times, using the same PDF to extract output power levels, until expression (3.3) is hold: then a new deployment C with length N has been defined and the function returns the control to the algorithm.
It is important to underline that not every time the function Deployment_Generator is called to modify the intervals U_1, U_2 and U_3. It is done when the algorithm requests such action, in **Step 3**, otherwise, intervals are not changed and every new deployment is calculated with the same intervals previously used.

- **Step 2:** The new deployment is evaluated. The algorithm calculates the lifetime and compares it with the best found deployment lifetime until this moment. If lifetime improvement is detected, the new deployment is updated in **Step 3**, otherwise, the algorithm requests modifications on the intervals and the function Deployment_Generator is called to implement such modifications.
- **Step 3:** Deployment lifetime features evaluation, update and modifications of the intervals U_i.
- **Step 4:** When the iteration number reaches the value Max_Iterations the algorithm stops.

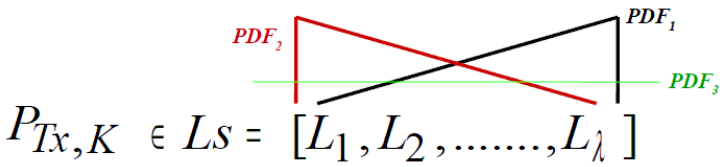


Figure 3.9: **With an uniform probability density function (PDF) we generate two numbers; q_1 and q_2 over the interval $[0,1]$. Therefore, intervals are defined as: $U_1 = [0, q_1]$, $U_2 = [q_1, q_2]$ and $U_3 = [q_2, 1]$. Finally if $X \in U_1$ it is used PDF_1, if $X \in U_2$ it is used PDF_2 and otherwise PDF_3.**

3.6 Case study: simulation results

The heuristic algorithm for searching the deployment has been implemented in MATLAB®.
The software-hardware platform refer to the Crossbow MICAz node. **Table 3.2** shows the used parameters values for the preliminary study and **Table 3.3** as well as **Table 3.4** summarize the resulting deployment for three cases-study.
We analyzed the algorithm performance in three cases: $D = 200\,m$, $300\,m$ and $350\,m$ respectively with $T_A = 60\,s$ and $d_{max} = 3m$. **Table 3.3** summarize results.
For the case $D = 200\,m$, the deployment that fulfills specifications and maximizes lifetime can be built with 68 nodes. The lifetime is 1127 hours (47 days) and the effective coverage range is 200.5 meters.
The algorithm takes a run-time of 525 seconds with `Max_Iterations=3000` on a computer with CPU AMD Athlon(tm) of $1.79GHz$ and $512\,MB$ RAM.
Nodes lifetime values vs. node number for the case $D = 200\,m$ is depicted in **Figure 3.10**. It is of interest to note that the node which determines the network lifetime is always the node nearest to the sink, because this node runs out first. The same effects is observed for the cases $D = 300\,m$ and $D = 350\,m$.
Since the algorithm has no theoretical expression to estimate the convergence rate, its convergence capability has to be studied by performing statistical study of the different algorithm trials results.

Table 3.2: **Parameter Values Summary.**

Symbol	Value	Note or Reference
T	1.1 ms	([1])
T_A	60 s	example of user requirement
Q_S	$0.52\mu\, Ah$	See chapter 2
I_{Rx}	19.7 mA	[**CC2420 Radio Device**]
L_s	$[-25, -15, -10, -7, -5, -3, -1, 0]$ dBm	[**CC2420 Radio Device**]
$I_{Tx,S}$	$[8.5, 9.9, 11.2, 13.9, 15.2, 16.5, 17.4]$ mA	[**CC2420 Radio Device**]
λ	8	[**CC2420 Radio Device**]
C_{Batt}	3000 mAh	([2])
D	$200\,m$ $300\,m$ $350\,m$	3 cases studies
d_{max}	$3\,m$	example of user requirement
S_L	7 years at 21 ℃(80% of initial capacity)	Battery $E91$ ([3])

[1] We use the IEEE 802.15.4 Frame Format standard compliant [**CC2420 Radio Device**]. We assume that the packet is 34 bytes long. The radio CC2420 transmits at 250 Kbps.
[2] Two non-rechargeable batteries type alkaline AA with voltage $1.5V$. Nodes use two batteries.
[3] Product Datasheet E91: http://data.energizer.com/PDFs/E91.pdf.

Table 3.3: **Deployments Results.**

D (m)	Lifetime (Hours)	Number of nodes (N)	Achieved Coverage (m)
200	1127	68	200.5
300	958	102	301.3
350	881	120	350.4

Table 3.4: **Topology Configuration Results.**

D (m)	Distance set $\{dc_K\}_{K=1,\cdots,N}$	Output Power Level $\{P_{Tx,K}\}_{K=1,\cdots,N}$
200	$dc_K = 3$ for all K except for: for $K = 1, 30 \rightarrow dc_K = 1.9\,m$ for $K = 22 \rightarrow dc_K = 1.7\,m$	$P_{Tx,K} = -7\,dBm$ for all K except for: for $K = 1, 30 \rightarrow P_{Tx} = -10\,dBm$ for $K = 22 \rightarrow P_{Tx} = -15\,dBm$
300	$dc_K = 3$ for all K except for: for $K = 3, 26 \rightarrow dc_K = 1.9\,m$ for $K = 37 \rightarrow dc_K = 0.5\,m$	$P_{Tx,K} = -7\,dBm$ for all K except for: for $K = 3, 26 \rightarrow P_{Tx} = -10\,dBm$ for $K = 37 \rightarrow P_{Tx} = -25\,dBm$
350	$dc_K = 3$ for all K except for: for $K = 23, 28, 84 \rightarrow dc_K = 1.9\,m$ for $K = 30 \rightarrow dc_K = 1.7\,m$ for $K = 29, 57 \rightarrow dc_K = 0.5\,m$	$P_{Tx,K} = -7\,dBm$ for all K except for: for $K = 23, 28, 84 \rightarrow P_{Tx} = -10\,dBm$ for $K = 30 \rightarrow P_{Tx} = -15\,dBm$ for $K = 29, 57 \rightarrow P_{Tx} = -25\,dBm$

To do this, we implemented 100 runs for the three cases presented above. Lifetime results and statistical information are offered in **Table 3.5**.
In closing this section we mention several remarks. This algorithm converges to its steady-state faster than other algorithms we have previously tested, such as Annealing and Genetic Algorithms.
On the other hand, it is important to underline that the computational complexity increases non-linearly with the number of parameters such as L_s and $I_{Tx,S}$.

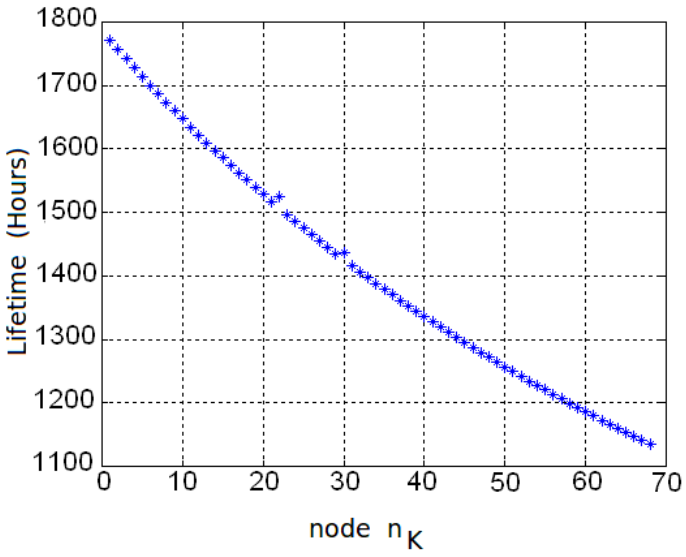


Figure 3.10: **Node lifetime for each node: (Case $D = 200\,m$).**

Table 3.5: **Statistical Convergence Capability Evaluation. Obtained Lifetime (hours) for different runs and the associated statistical parameters.**

Case	Mean	Minimum obtained value	Maximum obtained value	Standard Dev.
$D = 200\,m$	1121	1100	1135	8.67
$D = 300\,m$	946	926	960	6.98
$D = 350\,m$	879	866	888	5.24

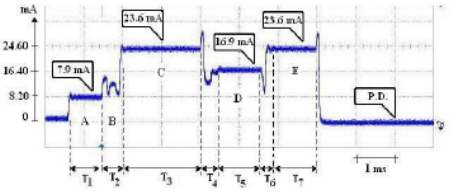
3.7 Final considerations

The deployment lifetime maximization is a problem that can not be solved by exploring all possible combinations that variables and specifications generate in real applications. The main chapter's objective was to solve this constrained optimization problem and in this framework we developed a heuristic method for searching the wireless sensors deployment with longer lifetime. The algorithm can efficiently explore the deployment design space and the best deployment is reached with acceptable convergence capability and significant saving in CPU time.
Moreover, the algorithm is highly successful in meeting the WSN designer's needs and we consolidated the relationship among the network parameters.
Several tests must be conducted in order to assure that the deployment solution is the optimal deployment, i.e the deployment with higher lifetime. However, it is important to underline that novel resulting deployments arise when hypothesis on WSN models are relaxed toward handling a network placement problem more related with real world specifications and hardware features. Moreover, more trials with different parameters values have to be implemented in order to test the convergence properties and robustness.
Results provided by the algorithm have been presented for three cases. A noteworthy and novel observation is that results presented in **Table 3.4** reveal that deployment solutions, conversely with results presented in [**Shelby, Z. (2005)**], have the tendency of taking the equal-distance topology.

In our case, with constrained distances, real energy (charge) consumption models and a discrete set of output power levels (hypothesis which have not been assumed in [**Shelby, Z. (2005)**]) make the deployment solution seems to prefer to use the mesh user specified d_{max} distances in

order to maximize the network lifetime. We define this as the natural-mode of the deployment. The explanation of such behavior deserves attention.

It can be resolved by taking into account the measurements performed in **Chapter 2** (see **Figure 3.11**). From this perspective, it is noted that the power consumption (and energy consumption) is dominated by the whole protocol and not for the antenna output power as it is often claimed in literature.
Case A is frequently assumed as the real situation. This is used into the optimization methodology, nevertheless it is not correct at all.
The implementation of the communication protocol required dedicated hardware which implements the radio control states depicted in **Figure 3.12**.
Therefore, it is clear that in such situation, the minimization of the energy consumption (i.e. lifetime maximization) is achieved by reducing the traffic of packets, then the algorithm uses reduced number of nodes and tries to maximize the distance between them.
Of course, as distance between nodes increases, the output power level at the antenna increases also, but it does not have significant impact on the overall node current consumption. Reduced number of nodes placed at the maximum possible distance between them is an attractive deployment for the algorithm because it reduces energy consumption by reducing traffic.



Charge Consumption for each output power level available in the radio transceiver CC2420

Output Power Level	Current Consumption (mA) [8]	Charge Cost (nAh)	Relative Charge Cost	Energy Cost (mJ)
0 dBm	17.4	41.9	1.083	0.453
-1 dBm	16.5	41.4	1.070	0.447
-3 dBm	15.2	40.8	1.054	0.441
-5 dBm	13.9	40.3	1.041	0.435
-7 dBm	12.5	39.7	1.026	0.429
-10 dBm	11.2	39.2	1.013	0.423
-15 dBm	9.9	39.1	1.010	0.422
-25 dBm	8.5	38.7	1	0.418

$$P_{out} = P_{circuitry} + K P_{Tx\,out}^{n}$$

A $P_{circuitry} << K P_{Tx\,out}^{n}$

B $P_{circuitry} >> K P_{Tx\,out}^{n}$

Figure 3.11: **Case A: conventional models assume that the output power at the antenna is the dominant power consumption. Instead, with the methodology presented in Chapter 2, we have found that case B is the correct behavior . The power consumption due to the associated radio circuitry and the current consumption required to implement the protocol are dominant in the optimization process. The parameters K and n depend on the circuit and the radio channel propagation environment.**

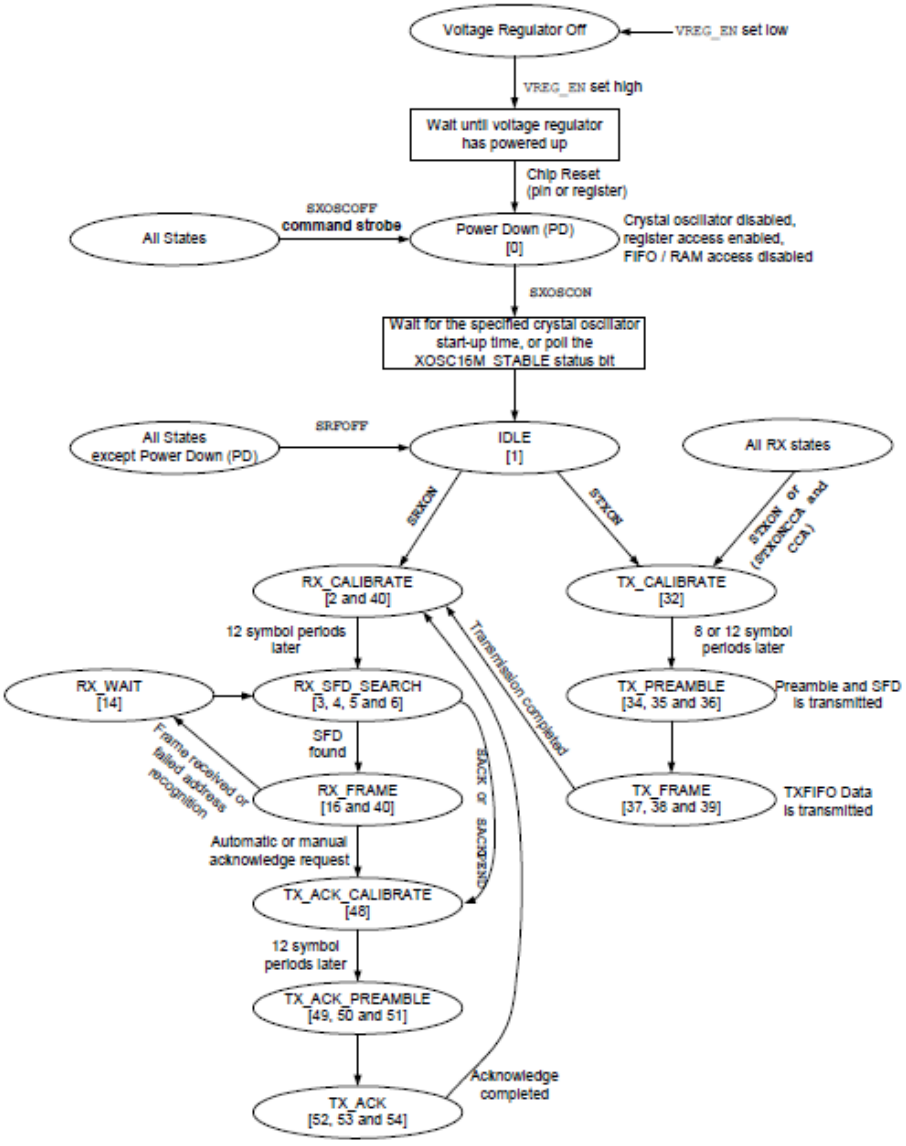


Figure 24. Radio control states

Figure 3.12: **CC2420 radio transceiver state machine used to implement protocol features (IEEE 802.15.4 and ZigBee systems) [CC2420 Radio Device].**

Chapter 4

Digital Signal Processing on Wireless Sensors Nodes

Chapter Summary

The MICAz node is built with the microcontroller Atmega128 which offers an integrated AD converter. In this chapter we present an overview of the main AD converter (ADC) features and among the information provided by datasheets, we select suitable sampling frequencies in order to perform sensor sampling with a posteriori digital filtering stage. The selected filter is a $L-th$ Band Filter Hamming Windowed because such filter is representative of which are often used for implementing Digital Wavelets Transforms. The selected sampling frequency was $7.98\,kHz$. In order to implement the filter that best matches the hardware capabilities, we have studied the different data representation types available in TinyOS. Moreover, measurements of computation operation time have been performed, i.e. the required time to perform multiplications, sums and divisions with different data representation. The filter has been programmed into the node and tested with different input analog signals, having bandwidth up to $4\,kHz$. Advantages and lacks of digital signal processing algorithms under the operating system implemented on general-purpose microcontrollers have been studied.
We conclude that does not exist an optimized interaction between signal acquisition circuitry, general purpose microcontrollers, signal processing techniques and the operating system TinyOs 2.x, which is unacceptable from the power consumption point of view.

4.1 Introduction

Wireless sensor nodes are often built with 8 and 16-bits general-purpose microcontrollers that offer very constrained hardware resources, therefore, nodes are not capable of implementing high performance filtering process and signal processing techniques as Digital Signal Processors (DSP) can do, and for which are usually selected.
However, the main advantages that the micontrollers meet are: low cost, reduced size, low power consumption with architectures widely tested, and nowadays, the possibility of complete integration into the same radio transceiver die (or silicon chip).
All are very desired properties for implementing wireless sensor nodes platforms; due to this, the study and review of signal processing techniques as well as methodologies to enhance sensor sampling and signal processing capabilities under low power consumption and low voltage (i.e. $1-1.2\,V$), are reasonable and it has perfect sense within the wireless sensor node framework.

4.1.1 Chapter goal

The chapter introduces to you how the operating system TinyOs together to the node as constrained hardware platform can yield the better signal processing capability. This study is essential as preliminary step in order to implement ad-hoc algorithms for signal processing and to highlight or to take the *right road* from the beginning of the application implementation. The chapter aims to examine advantages and lacks of digital signal processing with the operating system implemented on general-purpose microcontrollers. Due to this study and for the rest of the performed activity some programming techniques have been avoided. Final recommendations are offered to developers in the concluding section.

4.2 The Atmega128 AD converter

The microprocessor Atmega128 on which is based the MICAz node, has an ADC converter whose features are described in this section. We point out features which are important for our sampling requirements and signal processing techniques. For this, we offer a summarized overview of the technical data shown in [**ATmega128, Datasheet**].
The sampling process is performed with a $10-bit$ successive approximation ADC, that converts the analog input signal into digital samples through the successive approximation technique. As is mentioned in the datasheet, the ADC has selectable reference voltage, 8 multiplexed single ended input channels, 7 differential input channels and 2 differential input channels with optional gain of $10x$ and $200x$. For implementing conversions, the ADC requires certain clock frequency between $50\,Hz$ and $200\,kHz$. Therefore, the first issue that deserves attention is that we must identify available sampling rates, because we must select one sampling frequency among a discrete set of possible values.

According to technical data, the AVR CPU is directly clocked by the clock source which has been selected for the microcontroller, for instance an external crystal, no internal clock division is used. In the case of the MICAz node, this clock source is a crystal of frequency $7.3828\,MHz$. Three bits named *ADC Prescaler Select Bits* (ADPSx in the register ADCSRA) determine the ADC input clock frequency as a division factor of the crystal frequency that drives the microcontroller. A diagram block with the ADC clock sources is depicted in **Figure 4.1**. Moreover, the same figure also shows the sampling timing diagram.
In **Table 4.1** the ADC typical conversion times are presented. All time are referenced with respect to the ADC clock period or cycle.

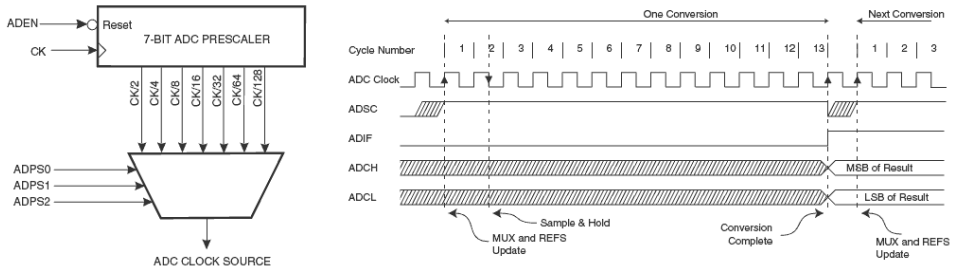


Figure 4.1: **(Left) ADC Prescaler, Figure 109 from [ATmega128, Datasheet]. (Right) ADC Timing Diagram, Single Conversion, Figure 111 from [ATmega128, Datasheet]. The ADSC signal starts the conversion process. Once a conversion is completed, the signal ADIF (AD interrupt flag) is set and data registers ADCH and ADCL are updated with the fresh measured signal.**

Table 4.1: **Conversion Times (Table 95 from [ATmega128, Datasheet]).**

Condition	Sample and Hold	Conversion Time (Cycles)
First Conversion	14.5	25
Normal conversion (single ended)	1.5	13
Normal conversion (differential)	1.5/2.5	13/14

Finally, in **Table 4.2** we estimate the different available options to be selected as ADC clock source.
On the other hand, according to the ADC timing Diagram , in **Table 4.3** it is calculated time between two consecutive conversions. This estimation is very important because it determines the sampling rate (in samples per seconds). We assume that for the worst case, the conversion time takes 14.5 clock cycles, according to the information provided by the **Table 4.1**.

Table 4.2: **Division factor: CPU clock vs. ADC clock [ATmega128, Datasheet].**

Bit ADPS2	Bit ADPS1	Bit ADPS0	Division Factor	ADC Clock Source
0	0	0	2	3.6914 MHz ($T_{ck} = 0.271\,\mu s$)
0	0	1	2	3.6914 MHz ($T_{ck} = 0.271\,\mu s$)
0	1	0	4	1.8457 MHz ($T_{ck} = 0.542\,\mu s$)
0	1	1	8	922.85 kHz ($T_{ck} = 1.08\,\mu s$)
1	0	0	16	461.43 kHz ($T_{ck} = 2.17\,\mu s$)
1	0	1	32	230.71 kHz ($T_{ck} = 4.33\,\mu s$)
1	1	0	64	115.36 kHz ($T_{ck} = 8.67\,\mu s$)
1	1	1	128	57.68 kHz ($T_{ck} = 17.34\,\mu s$)

According to the datasheet, if the ADC converter works at frequencies above $200\,kHz$ or higher sample rate, the resolution is lower than 10 bits. The number of bits determines the accuracy of the sampled signal as well as the processing algorithm complexity due to the data size for processing. It is important to take into account that most of the used microcontrollers to build wireless sensor nodes, are 8-bits based microcontrollers, and then, if higher data size need to be managed, it have to be implemented at software level, incrementing energy consumption.

Table 4.3: **ADC Clock Source and maximum sampling rate in free running conversion mode.**

Division Factor	ADC Clock Source	Maximum Sampling rate ($kSamples/s$)
2	$3.6914\,MHz$ ($T_{ck} = 0.271\,\mu s$)	254.45 ($T_s = 3.93\,\mu s$)
2	$3.6914\,MHz$ ($T_{ck} = 0.271\,\mu s$)	254.45 ($T_s = 3.93\,\mu s$)
4	$1.8457\,MHz$ ($T_{ck} = 0.542\,\mu s$)	127.38 ($T_s = 7.85\,\mu s$)
8	$922.85\,kHz$ ($T_{ck} = 1.08\,\mu s$)	63.86 ($T_s = 15.66\,\mu s$)
16	$461.43\,kHz$ ($T_{ck} = 2.17\,\mu s$)	31.78 ($T_s = 31.47\,\mu s$)
32	$230.71\,kHz$ ($T_{ck} = 4.33\,\mu s$)	15.93 ($T_s = 62.79\,\mu s$)
64	$115.36\,kHz$ ($T_{ck} = 8.67\,\mu s$)	7.95 ($T_s = 125.72\,\mu s$)
128	$57.68\,kHz$ ($T_{ck} = 17.34\,\mu s$)	3.98 ($T_s = 251.43\,\mu s$)

According to the technical data, the resulting sampled data is placed in the ADC Data Register. The sampled signal has 10 bits.The sampled signal should be place into two 8-bit registers, named ADCH and ADCL. The register ADCH (or ADC Data Register High Byte) is mapped in the address 05_{hex} (25_{hex}) and the ADCL (or ADC Data Register Low Byte) is mapped in 04_{hex} (24_{hex}). Both registers conform the ADC Data Register.

The presentation of the ADC conversion can be controlled by the bit ADLAR bit. It adjusts the result to right or left side as is depicted in **Figure 4.2**. In configuration A , the register ADCL must be read first. However, when precision less than 8 bits could be enough, then, the configuration B it is more efficiently from the software implementation point of view, because manage only one byte, instead, configuration A requires to handle 2 bytes .

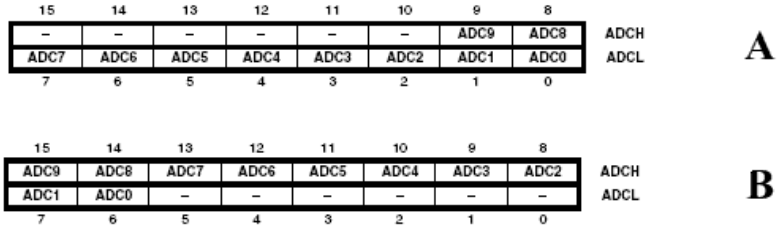


Figure 4.2: **The presentation of the ADC conversion.**

4.3 Conflict between TinyOs and *user-configured* interruptions

In **Tabel 4.4** we offer important observations. By using the piece of code immediately shown after the **Tabel 4.4**, in the **Listening 4.1** we have measured the time between two consecutive events SET_BIT(PORTB,4). It sets the pin 4 of the PORTB to the logical level 1 and the measured time is the sampling time (column 3).
For the cases *Division Factor* (column 1) from 128 to 16 it is observed a mismatch between the expected and measured time. However, interruptions take place regularly and clearly identified in the oscilloscope.
Instead, for cases from 8 to 2 (marked with Yes in the column 4), it has been impossible to take measurements. The signal corresponding to the event SET_BIT(PORTB,4) suffers instability and the signal can not be triggered by the oscilloscope.
The measurements in **Table 4.4**, for which conflicts have been observed (and marked with Yes), can be performed without problem only after **changing the TinyOs interruption commands used for the internal scheduler clock; it has been canceled**. At this point, the operating system TinyOs becomes inoperative and the AD converter works in free running mode having its associated interruptions perfectly observed in the oscilloscope screen.
At this moment is not completely known the cause, probably this misbehavior is caused by bugs in the TinyOs scheduler configuration. But this simple experiment evidences drawbacks in the operating systems because it is not possible to use the complete hardware capability , i.e. the maximum speed for sensor sampling.
It is required to implement modifications inside the operating system kernel as well as the design of ad-hoc drivers according to the application's needs.

Table 4.4: **Measured sampling rate and TinyOs conflict.**

Division Factor	Maximum Sampling rate ($kSamples/s$)		Measured T_s	Observed conflict
2	254.45	($T_s = 3.93\,\mu s$)	$T_s = 3.91\,\mu s$	Yes
2	254.45	($T_s = 3.93\,\mu s$)	$T_s = 3.91\,\mu s$	Yes
4	127.38	($T_s = 7.85\,\mu s$)	$T_s = 7.08\,\mu s$	Yes
8	63.86	($T_s = 15.66\,\mu s$)	$T_s = 14.1\,\mu s$	Yes
16	31.78	($T_s = 31.47\,\mu s$)	$T_s = 28.1\,\mu s$	No
32	15.93	($T_s = 62.79\,\mu s$)	$T_s = 55.8\,\mu s$	No
64	7.95	($T_s = 125.72\,\mu s$)	$T_s = 112.0\,\mu s$	No
128	3.98	($T_s = 251.43\,\mu s$)	$T_s = 228.0\,\mu s$	No

Listing 4.1: **Used NesC code section for the measurements offered in Table 4.4**

```
1  /**
2  *AD interrupt handlers
3  */
4
5  AVR_ATOMIC_HANDLER(SIG_ADC){
6
7  SET_BIT(PORTB,4);
8  data=ADC;
9  sei();   // global set interuptus I-bit SREG
10          // equal __nesc_enable_interrup();
11 CLR_BIT(PORTB,4);
12 }
```

4.4 Data representation types in TinyOS

Data types definitions (also named as typedefs) are specified in the files <stdint.h>, <limits.h> and <float.h>, within the TinyOS directory.

These file are also headers libraries in the C standard library, introduced by the Programming languages-C International Standard **[IEC 9899:TC2]**. Instead of standard C names such as int, long, or char, the TinyOS operating system and its programing language NesC, use formats which are declared in the language-C International Standard.
The name convention defines [u]intN_t integer types, which have specific width. The integer type intN_t is used for signed integers and uintN_t is used for unsigned integers. In addition to this, the header library defines the limits for each integer value.
The reason for using this definition lies in the need of to avoid hardware platform-specific definitions. In some hardware platform, a variable declared as int could be represented and stored in a 16-bits register, but for another, it would be represented and stored in a 32-bits register.

Thus, definitions for data with fixed widths are as follows:

- typedef signed char int8_t
- typedef unsigned char uint8_t
- typedef signed int int16_t
- typedef unsigned int uint16_t
- typedef signed long int int32_t
- typedef unsigned long int uint32_t
- typedef signed long long int int64_t
- typedef unsigned long long int uint64_t

Table 4.5: **The commonly used types in TinyOS and their associated ranges.**

Data Type	Sign	Maximum	Minimum
int_8 (1 byte)	signed	127 ($7F_{hex}$)	−128
uint_8 (1 byte)	unsigned	255	0
int_16 (2 byte)	signed	32767 ($7FFF_{hex}$)	−32768
uint_16 (2 byte)	unsigned	65535	0
int_32 (4 byte)	signed	2 147 483 647 ($7FFFFFFF_{hex}$)	−2 147 483 648
uint_32 (4 byte)	unsigned	4 294 967 295	0
int_64 (8 byte)	signed	9 223 372 036 854 775 807 ($7FFFFFFFFFFFFFFF_{hex}$)	−9 223 372 036 854 775 808
uint_32 (4 byte)	unsigned	18 446 744 073 709 551 615	0

TinyOS supports floating point numbers. The arithmetic is implemented in software rather than hardware and it is defined in the header libraries <math.h> and <float.h>. The specified floating

point type for arithmetic is described in the header libraries: <float.h> and <limits.h>. The representation is as follows **[IEC 9899:TC2]**:

$$x = s\, b^e \sum_{k=1}^{p} f_k b^{-k} \tag{4.1}$$

Where, as is defined in the standard:

- s: sign (value -1 or 1)
- b: base or radix for the exponent representation (value $b > 0$)
- e: exponent ($e_{min} < e < e_{max}$)
- p: precision (the number of base-b digits in the significant digits)
- f_k: non negative integers less than b (the significant digits)

To have better insight, we show the an example which is provided in **[IEC 9899:TC2]** (more specifically, in Section 5.2.4.2.2 Characteristics of floating types <float.h>)

$$x = s\, 16^e \sum_{k=1}^{6} f_k 16^{-k} \tag{4.2}$$

According to the notation used in the header library <float.h>, the following parameters are introduced for (4.2) (the explanation is extracted from Section 5.2.4.2.2 Characteristics of floating types <float.h> **[IEC 9899:TC2]**):

- FLT_RADIX; value: 16
  ... *"radix for the exponent representation, b "* ...
- FLT_MANT_DIG ; value: 6
  ... *"number of base-FLT_ RADIX digits in the significand (or also mantissa), p "* ...
- FLT_EPSILON; value: $9.53674316e-07F$
  ... *"the difference between* 1 *and the least value greater than* 1 *that is representable in the given floating point type,* $b^{(1-p)}$ *"* ...
- FLT_DIG; value: 6
  ... *"number of decimal digits, q, such that any floating-point number with q decimal digits can be rounded into a floating-point number with p radix b digits and back again without change to the q decimal digits;* $p \log_{10}(b)$ *if b is a power of* 10, $floor[\ (p-1)log_{10}(b)\ ]$ *otherwise "* ...
- FLT_MIN_EXP; (e_{min}), value: -31
- FLT_MIN_10_EXP; value: -38
  ... *"minimum negative integer such that* 10 *raised to that power is in the range of normalized floating-point numbers;* $ceil[\ log_{10}(b^{(e_{min}-1)}\ ]$ *"* ...
- FLT_MAX_EXP ; (e_{max}) value: $+32$
- FLT_MAX ; value $3.40282347e+38F$
  ... *"maximum representable finite floating-point number;* $(1-b^{-p})b^{e_{max}}]$ *"* ...
- FLT_MAX_10_EXP ; value $+38$
  ... *"maximum integer such that* 10 *raised to that power is in the range of representable finite floating-point numbers;* $floor[\ log_{10}((1-b^p)b^{e_{max}})]$ *"* ...
- FLT_MIN ; value $2.93873588e-39F$
  ... *"minimum normalized positive floating-point number;* $b^{(e_{min}-1)}$ *"* ...

4.4.1 Floating types characteristics for TinyOs 2.x

For our case, parameters' values for the TinyOs 2.x are listed in the **Table 4.6** . To obtain the information, it must be written in the shell the following commands in order to display macros' definitions:

$: touch empty.c
$: avr-gcc -E -Wp,-dD,-dM empty.c

Table 4.6: **Parameter values used in the TinyOS 2.X version.**

Parameter	Value
FLT_RADIX	2
FLT_MANT_DIG	24
FLT_EPSILON	$1.1920929e-7F$
FLT_DIG	6
FLT_MIN_EXP	-125
FLT_MAX_EXP	$+128$
FLT_MIN_10_EXP	-37
FLT_MAX_10_EXP	$+38$
FLT_MAX	$3.40282347e+38F$
FLT_MIN	$1.17549435e-38F$

Note: rpms: /bin/rpm: nesc-1.3.0-1.fc9.i386, avr-libc-1.4.7-1.i386, avr-gcc-3.4.3-1.i386, automake-1.10.1-2.noarch, imake-1.0.2-6.fc9.i386, automake14-1.4p6-15.fc7.noarch, avrdude-tinyos-5.6cvs-1.i386, automake17-1.7.9-11.noarch, make-3.81-12.fc9.i386, tinyos-deputy-1.1-1.fc9.i386, automake16-1.6.3-14.noarch, avr-binutils-2.15tinyos-3.i386, avarice-2.6-2.fc9.i386, tinyos-2.1.0-1.fc9.noarch, automake15-1.5-23.noarch, tinyos-tools-1.3.0-1.fc9.i386

4.5 Computation time measurement

Nodes in general are built with general-purpose microcontrollers without dedicated hardware such as hardware multipliers [**AVR201, App. Note**]. For this reason it is important to measure the operation computation time.
As regards digital filter implementation it is possible to find a huge number of specialized literature such as [**AVR223, App.Note**]. Nevertheless, due to the compilers nature, it has to be good practice to perform a hardware-software characterization to assure that the compiler makes the correct compilation from the NesC language to assembler language. Performance analysis it is also important because does not exist literature which enable developers to estimate computation and algorithms execution time.
In the **Listening 4.2** it is shown the NesC piece of code which has been used for performing the measurements of **Tables 4.7, 4.8 and 4.9**

Listing 4.2: **NesC code section which has been used for the measurements offered in Tables 4.7, 4.8 and 4.9**

```
1  void MilliTimer.fired() {
2      call MilliTimer.stop();
3      while(0==0) {
4        pointer=5;
5      __asm ("nop");
6      __asm ("nop");
7       SET_BIT(PORTB,4);
8            yt=h1*xs[pointer] ;
9       CLR_BIT(PORTB,4);
10     __asm ("nop");
11     __asm ("nop");
12       if (yt == yexp){
13           call Leds.led0On();
14           yt=0;
15                       } }
16 }
```

Table 4.7: **Measured time for operation execution. Case: multiplication**

Case	Operation	NesC Example (declarations)	Execution Time
	Multiplication		
1	$yt = (\text{int16\_t})h_1 * x_s$	int8_t $x_s[p]$; int16_t const $h_1 = -100$; int16_t yt;	$1.67\mu s$
2	$yt = (\text{int16\_t})h_1 * x_s$	uint8_t $x_s[p]$; int16_t const $h_1 = -100$; int16_t yt;	$2.58\mu s$
3	$yt = (\text{int32\_t})h_1 * x_s$	uint8_t $x_s[p]$; int16_t const $h_1 = -15123$; int32_t yt;	$8.60\mu s$
4	$yt = (\text{int32\_t})h_1 * x_s$	int8_t $x_s[p]$; int16_t const $h_1 = -15123$; int32_t yt;	$8.60\mu s$
5	$yt = h_1 * x_s$	uint16_t $x_s[p]$; int16_t const $h_1 = -15123$; int16_t yt;	$3.08\mu s$
6	$yt = h_1 * x_s$	int16_t $x_s[p]$; int16_t const $h_1 = -15123$; int16_t yt;	$3.08\mu s$
7	$yt = (\text{int32\_t})h_1 * x_s$	uint16_t $x_s[p]$; int16_t const $h_1 = -1097451$; int32_t yt;	$8.70\mu s$
8	$yt = (\text{int32\_t})h_1 * x_s$	int16_t $x_s[p]$; int16_t const $h_1 = -1097451$; int32_t yt;	$9.20\mu s$
9	$yt = h_1 * x_s$	int16_t $x_s[p]$; int32_t const $h_1 = -150923621$ int32_t yt;	$9.88\mu s$
10	$yt = (\text{int64\_t})h_1 * x_s$	int16_t $x_s[p]$; int32_t const $h_1 = -150923621$ int64_t yt;	$122\mu s$
11	$yt = h_1 * x_s$	int16_t $x_s[p]$; int64_t const $h_1 = -8007199254740991$ int64_t yt;	$122\mu s$
12	$yt = h_1 * x_s$	int16_t $x_s[p]$; float const $h_1 = -5.3E-12F$ float yt;	$253\mu s$
13	$yt = h_1 * x_s$	float $x_s[p]$; float const $h_1 = -5.3E-12F$ float yt;	$556\mu s$
14	$yt = (\text{float})x_s$	int16_t $x_s[p]$; float yt;	$240\mu s$

Table 4.8: **Measured time for operation execution. Case: sum**

Case	Operation	NesC Example (declarations)	Execution Time
	Sum		
1	$yt = x_{s1} + x_{s2}$	int8_t $x_{s1}[p]$, $x_{s2}[p]$ int8_t yt; $yt = x_{s1}[p] + x_{s2}[p]$	$2.6\mu s$
2	$yt = x_{s1} + x_{s2}$	int16_t $x_{s1}[p]$, $x_{s2}[p]$ int16_t yt; $yt = x_{s1}[p] + x_{s2}[p]$	$3.75\mu s$
3	$yt = x_{s1} + x_{s2}$	int32_t $x_{s1}[p]$, $x_{s2}[p]$ int32_t yt; $yt = x_{s1}[p] + x_{s2}[p]$	$5.56\mu s$
4	$yt = x_{s1} + x_{s2}$	int32_t $x_{s1}[p]$, $x_{s2}[p]$ int64_t yt; $yt = x_{s1}[p] + x_{s2}[p]$	$7.23\mu s$
5	$yt = x_{s1} + x_{s2}$	int64_t $x_{s1}[p]$, $x_{s2}[p]$ int64_t yt; $yt = x_{s1}[p] + x_{s2}[p]$	$24\mu s$
6	$yt = x_{s1} + x_{s2}$	float $x_{s1}[p]$, $x_{s1}[p] = -15.36E - 10F$ float $x_{s2}[p]$, $x_{s2}[p] = 21.2E + 2F$; float $yt = x_{s1}[p] + x_{s2}[p]$	$18.4\mu s$

Table 4.9: **Measured time for operation execution. Case: division**

Case	Operation	NesC Example (declarations)	Execution Time
	Division		
1	$yt = x_{s1}/x_{s2}$	float $x_{s1}[p]$, $x_{s1}[p] = -15.36E - 10F$ float $x_{s2}[p]$, $x_{s2}[p] = 21.2E + 2F$; float yt $yt = x_{s1}[p]/x_{s2}[p]$	$78.2\mu s$
2	$yt = x_{s1}/x_{s2}$	float $x_{s1}[p]$, $x_{s1}[p] = -15.36E - 10F$ float constant $h_1 = -5.3E - 12F$; float yt $yt = x_{s1}[p]/x_{s2}[p]$	$66.7\mu s$
3	$yt = x_{s1}/x_{s2}$	int32_t $x_{s1}[p] = -1147411002$ int32_t $x_{s2}[p] = 247481902$ int32_t yt $yt = x_{s1}[p]/x_{s2}[p]$ result: $yt = -4.636343073$ NesC truncated result $yt = -4$	$100\mu s$
4	$yt = x_{s1}/x_{s2}$	int32_t $x_{s1}[p] = -1047411002$ int16_t $x_{s2}[p] = 32152$ int16_t yt $yt = x_{s1}[p]/x_{s2}[p]$ result: $yt = -32576.853757154$ NesC truncated result $yt = -32576$	$100\mu s$
5	$yt = x_{s1}/x_{s2}$	int16_t $x_{s1}[p] = -32766$ int16_t $x_{s2}[p] = 32152$ int16_t yt $yt = x_{s1}[p]/x_{s2}[p]$ result: $yt = -1.01909679$ NesC truncated result: $yt = -1$	$35.5\mu s$

Table 4.10: **Example of code for the operation case 3 in Table 4.7.**

Main Program	Subroutine
⋮	**00000832 < mulsi3 >:**
movw r24, r28	832: 62 9f mul r22, r18
eor r26, r26	834: d0 01 movw r26, r0
sbrc r25, 7	836: 73 9f mul r23, r19
com r26	838: f0 01 movw r30, r0
mov r27, r26	83a: 82 9f mul r24, r18
movw r22, r24	83c: e0 0d add r30, r0
movw r24, r26	83e: f1 1d adc r31, r1
call 0x832	840: 64 9f mul r22, r20
movw r26, r24	842: e0 0d add r30, r0
movw r24, r22	844: f1 1d adc r31, r1
⋮	846: 92 9f mul r25, r18
	848: f0 0d add r31, r0
	84a: 83 9f mul r24, r19
	84c: f0 0d add r31, r0
	84e: 74 9f mul r23, r20
	850: f0 0d add r31, r0
	852: 65 9f mul r22, r21
	854: f0 0d add r31, r0
	856: 99 27 eor r25, r25
	858: 72 9f mul r23, r18
	85a: b0 0d add r27, r0
	85c: e1 1d adc r30, r1
	85e: f9 1f adc r31, r25
	860: 63 9f mul r22, r19
	862: b0 0d add r27, r0
	864: e1 1d adc r30, r1
	866: f9 1f adc r31, r25
	868: bd 01 movw r22, r26
	86a: cf 01 movw r24, r30
	86c: 11 24 eor r1, r1
	86e: 08 95 ret

4.6 Digital filter implementation test

This work has been performed together with the study described in **Chapter 6** because it was crucial to get insight about the behavior of the filter implemented into reduced and constrained hardware platforms. Our intention is to detect problems and drawbacks which could be object of enhancements and future optimization processes.
In order to design the digital filter which will be implemented into the node, we have used a tool named *ScopeFIRTM* which is provided by the *Iowegian International Corporation* as free 30-day trial period version available from `http://www.iowegian.com/download/loadfir.htm`
The ScopeFIRTM. distribution features graphical user interface for Finite Impulse Response (FIR) filter design.
By using this software, it is possible to design low-pass, high-pass or band-pass filters as well as Windowed Sinc, Raised Cosine, Maxflat, Hilber transforms and differentiators.
The software tool implements the Parks-MacClellan algorithm which allows us to estimate optimized filter coefficients values, which always depend on the pass-stop frequencies bands, bands edge frequencies, sampling frequency, band attenuation responses, rolloff factor and so on.
We have decided to implement a filter which features are described in **Table 4.11** and the filter magnitude of the frequency response is depicted in **Figure 4.3**.
It is a L-th band filter because is representative of the most used filters to implement digital wavelets transforms (interested readers can see in advance **Figure 6.9 in Chapter 6**).

The incoming signal after sampling is represented by an integer `uint16_t` because the maximum sampled value that the AD converter features with 10 bits is 1024.

Table 4.11: **Implemented Filter Features.**

Type	L-th Band Filter (or Nyquist Filter)
Sampling frequency	7.95 $KSamples/s$ ($Ts = 125.72\mu s$)
Number of taps	15
Upper Stop band	1.175 kHz
$F_s/2L$ ($L = 2$)	1.988 kHz
Lower Passband	2.8 kHz
Filter coefficients	$h_0 = 0.003651453990227175$, $h_1 = 0$
	$h_2 = -0.016179253392499472$, $h_3 = 0$
	$h_4 = 0.068411776788146486$, $h5 = 0$
	$h_6 = -0.304947517341087220$,
	$h_7 = 0.501872920089573850$,
	$h_8 = -0.304947517341087220$, $h_9 = 0$
	$h_{10} = 0.068411776788146486$, $h_{11} = 0$
	$h_{12} = -0.016179253392499475$,$h_{13} = 0$
	$h_{14} = 0.003651453990227175$

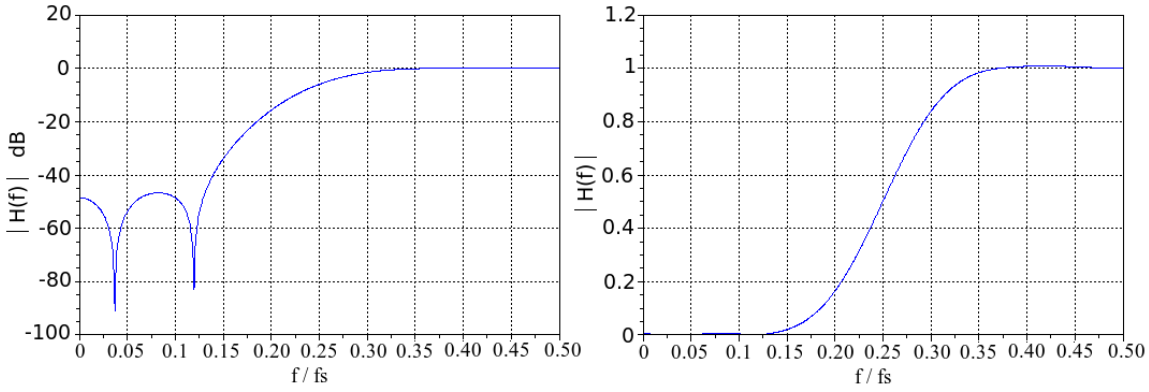


Figure 4.3: **FIR frequency response magnitude.**

As we have to implement the filter into the node, filter coefficients must be represented by using suitable format. Such process is done by multiplying each coefficient by certain constant and then rounding.
Table 4.12 shows the adapted filter coefficients. The higher tap is defined as $h_{i7} = 32767$ which is the maximum allowable value for the `int16_t` representation.
In **Figure 4.4** is depicted the filter frequency response $H(f)$ for both cases, the filter originally calculated (see **Table 4.12**) and the frequency response of the adapted filter implementation, i.e. the filter after coefficients modification.
As we can see , in **Figure 4.4**, the difference between them in frequency response for the `int16_t` data representation is quite negligible.

4.6.1 FIR test

The implemented FIR is tested with real sine input signals. Below we describe briefly the procedure for one performed test.
The first step consist in the selection of two arbitrary frequencies and for such frequencies we calculate the frequency response as follows:

- $|H_i(f)| = 65750 \quad @f = 3.18kHz$, i.e. $f/f_s = 0.4$
  $|H_i(f)| = 10750 \quad @f = 1.59kHz$, i.e. $f/f_s = 0.2$
  (see **Figure** 4.4)

An input sine waveform is selected:

Table 4.12: **Modifications implemented for the FIR implementation .**

filter	$y(n) = h(n) * x(n)$
input signal $x(n)$	$x(n)$ is represented by **uint16_t** (maximum value 1024)
coefficients $h(n)$	$h(n)$ is represented by **int16_t**
filter output $y(n)$	$y(n)$ is represented by **int32_t**
multiplication factor	65289
multiplication factor estimation	$32767/h_7 = 65289$
Filter coefficients	$h_0 = 0.003651453990227175$, $h_1 = 0$ $h_2 = -0.016179253392499472$, $h_3 = 0$ $h_4 = 0.068411776788146486$, $h5 = 0$ $h_6 = -0.304947517341087220$, $h_7 = 0.501872920089573850$, $h_8 = -0.304947517341087220$, $h_9 = 0$ $h_{10} = 0.068411776788146486$, $h_{11} = 0$ $h_{12} = -0.016179253392499475$,$h_{13} = 0$ $h_{14} = 0.003651453990227175$
Adapted filter coefficients	$h_{i0} = 238$, $h_{i1} = 0$ $h_{i2} = -1056$, $h_{i3} = 0$ $h_{i4} = 4467$, $h_{i5} = 0$ $h_{i6} = -19910$, $h_{i7} = 32767$, $h_{i8} = -19910$, $h_{i9} = 0$ $h_{i10} = 4467$, $h_{i11} = 0$ $h_{i12} = -1056$,$h_{i13} = 0$ $h_{i14} = 238$

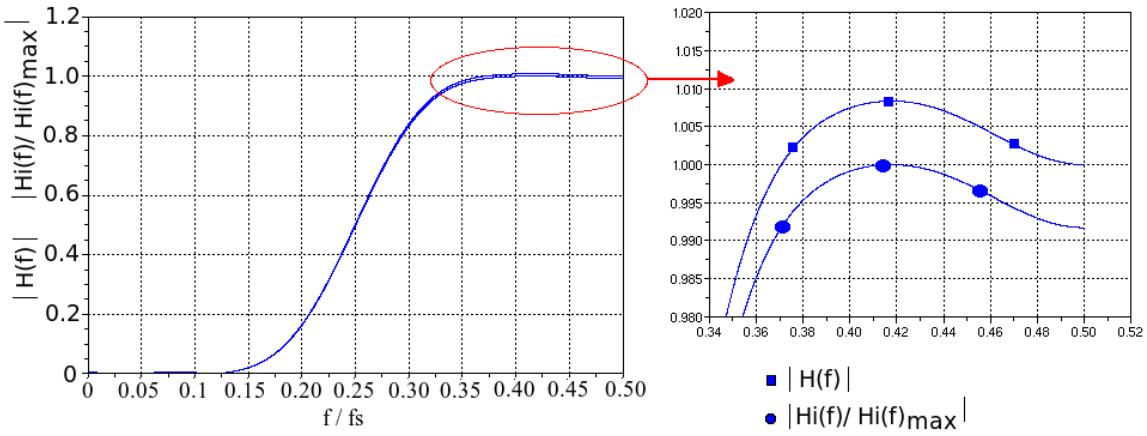


Figure 4.4: **Frequency responses (magnitude) of both FIR . The filter named $H(f)$ is the frequency response of the original filter. The filter $H_i(f)$ is the frequency response of the filter implemented with the modified coefficients.**

- **input signal:** for both frequencies cases the input signal is a wave $x(t)$ with the following features:
  $x(t) = Asin(2\pi f_o t) + x_{DC}$ with $A = 1.5V$ (quantized to 200_{hex} or 512 with $V_{REF} = 3V$)
  $x_{DC} = 1.5V$ (quantized to 200_{hex} or 512 with $V_{REF} = 3V$)

  Two frequency cases studies are separated:
  case 1: $f_1 = 3.18kHz$
  case 2: $f_2 = 1.59kHz$

Finally, we estimate the FIR output for both cases, i.e. the range of values to which $y(n)$ belongs.

- Expected output FIR: $y(n) = x(n) \star h(n)$ or $y(n) = A_f sin\left(2\pi\frac{f_o}{f_s}n + \Phi_f\right) + y_{DC}$
  case 1: $A_f = A.65750 = 33664000$ and **case 2:** $A_f = A.10750 = 5504000$

For both cases $y_{DC} = x_{DC} \sum_i h_i = x_{DC}.245 = 125440$

Therefore we have that:
case 1: the range results: $y(n) \in B_1 = [-33538560, 33789440] \quad \forall n$
case 2: the range results: $y(n) \in B_2 = [-5378560, 5629440] \quad \forall n$

For test the FIR behavior, we implemented a simple program that controls when $y(n)$ goes outside the expected interval B_1 or B_2 when the input signal frequency is changed. A red led acts as warning and no misbehavior has been observed.
Another important issue is the execution time. The execution time was estimated by using the measured computation time previously described. It results: $T_{ex} = 73.5\mu\, s$
The measured execution time results: $T_{ex,m} = 70.1\mu$

Through the filter implementation into the wireless sensors node, we have earned lessons to manage concepts related to the microncontroller architecture, the operating system and signal processing, referring the interested readers to the summary section for recommendations.

4.7 Final considerations

The main objective of this chapter was to get feeling about the computation capability available in wireless sensors nodes.
This work has been performed together with the study described in the **Chapter 6** because we need to implement digital filters that best match the hardware capabilities.
Different data representation types available in TinyOS have been studied. Moreover, measurements of computation operation time have been performed, i.e. the required time to perform multiplications, sums and divisions with different data representation.

Earned lessons are as follows:

- TinyOS supports floating point data representation but we don't recommend its use. There is a clear gap between the software implementation and the hardware capability. The floating point usage is excessively power demanding and no accuracy is gained with respect to more coarse (or moderate low-resolution) data representation during the filtering stage.
  Of course, at the end, the best data representation depends on the application-specific knowledge and specifications but we evidence that `int16_t` or `int32_t` could be enough for most of the WSN applications.
- General purpose microcontrollers are not suitable. The TinyOS does not use properly the hardware resources. In order to reduce power consumption, much closer cooperation between hardware and software is required. For instance, the implementation of multipliers, adders and parallel processing into wireless sensor nodes could be an emerging and interesting research field. In addition to this, it could gain great commercial success.
  Interested readers can take a look on `www.parallax.com` where the propeller chip which features eight processors that operate simultaneously, independently or cooperatively is presented. It could be an interesting new approach for building wireless sensors nodes.
- TinyOS is an event-oriented operating system with an internal clocked scheduler that interrupts the signal sampling process, making unreliable the signal processing algorithm. The event-oriented programming methodology furnish versatility and scalability to the WSN application, however this configuration provides disadvantages from the signal processing point of view.
- Some considerations related to the implemented filter deserve attention. An ad-hoc driver has been built to manage the ADC because we prefer to make our own driver rather than use HPL layers (*Hardware Presentation Layer* [**TinyOS**]) provided by TinyOS because we intend to have absolute control on the hardware. As we have previously discussed, for higher frequency sampling configurations some problems have been found.
- With regards to programming techniques, after performing this preliminary evaluation related to the wireless sensor node capability of signal processing, we decide to use for further developments *more C oriented* software structures.

More specifically, we decided don't use nesC defined *events*. Instead, we adopt *i)* inline defined functions; *ii)* asynchronous commands; *iii)* pointers to memory positions that every module can use and finally; *iv)* we call functions like other traditional C/C++ programs.
Nevertheless, we realized that to implement optimizations, a deep study about the compilation model is necessary. The compilation process is in fact intriguing, the nesC compiler invokes the cross compiler (avr-gcc) and produces the executable main.exe, therefore, in fact we have two compilers working; nesC to C and C to assembler.

Chapter 5

SNR Evaluation for Wireless Sensors Nodes and Embedded Systems

Chapter Summary

This chapter proposes a novel methodology for evaluating the signal-to-noise ratio (SNR) of data acquisition systems in wireless sensors microsystems. This methodology enables designers to evaluate the SNR at the electronic front end regardless of non-stationary sensor signals behavior (with unknown signal shapes) or battery voltage decrease due to the depletion. Moreover, the proposed methodology provides criteria to modify signal features (e.g. amplitude and common mode)and data acquisition system parameters (e.g. full scale range variation) for achieving required SNR values (e.g. to assure numerical stability of the algorithms implemented into the embedded system). Validation measurements and application examples are presented.

5.1 Introduction

Wireless Sensor Networks (WSNs) are composed by embedded electronic systems named wireless sensor nodes. Each node is built with a microcontroller, AD converters, radio transceiver, batteries for power supply, sensors and the associated signal conditioning circuits.
Input sensor signals are processed by the node. To guarantee that the numerical properties and rate of convergence of the digital signal processing algorithms don't deviate with respect to the expected behavior, it is often required SNR values above a given threshold.
In wireless sensors nodes the data acquisition system SNR depends mainly on the quantization noise introduced by the ADC.
Unfortunately, the SNR degradation is a problem that inherently arises related to the efforts for achieving low-power and low-voltage node operating modes due to the following two issues which deserve special attention: *Reduced voltage power supply* and *Strategies for adaptive data acquisition systems.*

5.1.1 Reduced voltage power supply

Due to recent advances in CMOS technology scaling, the tendency is to implement the entire node on a single chip. The use of deep-submicron CMOS technologies results in the design of embedded mixed-signal systems powered by reduced voltages (e.g $1.8V$, $1.2V$). Since the battery voltage progressively decreases,circuits powered by reduced voltages can obtain the maximum battery lifetime.
Unfortunately, the penalty for the reduced voltage supply use, could be the SNR degradation in the data acquisition system before the digital processing **[Chiu, Y. (2005)]**,**[Matsuzawa, A. (2007)]**.
On the other hand, due to energy consumption constraints, integrated voltage regulators are only used for RF circuits voltage regulation (e.g. the MicaZ node). If the battery voltage is used as full-scale voltage reference (V_{FS}), sensors and signal conditioning circuits experiment the progressive battery voltage supply decrease due to the battery depletion. The effect on the SNR is not clearly analyzed in literature.

5.1.2 Strategies for adaptive data acquisition systems

Embedded wireless sensor nodes must be aware to use data acquisition circuitry, without wasting energy (acquisition features are: sampling frequency, bit numbers for data representation, signal to noise ratio, ADC voltage reference and input signal amplitude). Strategies for implementing adaptive power-aware circuits are being increasingly reported and benefits on energy consumption reduction arise.
For instance, in **[Dlugosz, R. (2007)]** authors present a low-power 8-bit current mode interleaved successive approximation ADC. The ADC can adapt the sampling rate and the number of bits in order to

reach different power consumption levels. A noteworthy paper that highlights system tradeoffs exploration benefits is **[Dallago, E. (2007)]**, which proposes a sigma-delta modulator ratiometric with respect to the power supply. It is discussed the effects of using the system power supply as reference voltage instead of the usual band-gap circuit for reference voltage. The energy dissipated by reference buffers and band-gap circuits is then saved (for interested readers, the ratiometric conversion theory is well explained in [**Anderson, R. (2004)**]).
Another recent example related to adaptive quantization is presented in [**Virtanen, K. (2005)**], where the adaptive quantization is used to fulfil demands of video compression standards with lower power consumption.
In [**O'Driscoll, S. (2005)**] authors propose a low-power variable resolution ADC for neuro-prosthetic implants. It is explained that the ADC power consumption increases with the resolution (number of bits) and it is also discussed how the resolution is selected by considering signal-to-signal ratios,dynamic range and signal-to-noise ratio. The proposed ADC can adapt the number of bits from 3 to 8 bits in 1 bit steps for power saving.
Finally, a particularly significant and motivating article is [**Verma, N. (2007)**], where it is described a resolution-rate scalable ADC for micro-sensor networks. The ADC has two resolution modes (12 bit and 8 bit), and its sampling rate is also scalable at a constant figure-of-merit.

More specifically, to obtain better insight we discuss the following example. Approaches to achieve reduced power consumption can be obtained by means of expression (5.1). It is a power estimator for CMOS Nyquist-rate ADC top-down system exploration from [**Toumazou, C. (2002)**], Chapter 21.
Where: V_{dd} is the voltage power supply (or voltage full scale), $ENOB$ effective number of bits, L_{min} minimal transistor channel length, F_{signal} is the signal frequency (the sampling frequency is $2F_{signal}$) and F_{sample} is the frequency of the clock to produce the quantization after sampling.
Clearly, the power consumption P decreases by reducing V_{dd} and (or) the $ENOB$.

$$P = \frac{V_{dd}^2 L_{min} \left(F_{sample} + F_{signal}\right)}{10^{-0.1525ENOB+4838}} \tag{5.1}$$

With respect to power consumption, it is clear from (5.1) that system S_2 in **Figure 5.1** exhibits better power efficiency than system S_1, however, it is not known in advance if such benefit could be translated in terms of SNR degradation at the input of the digital signal processing algorithm.

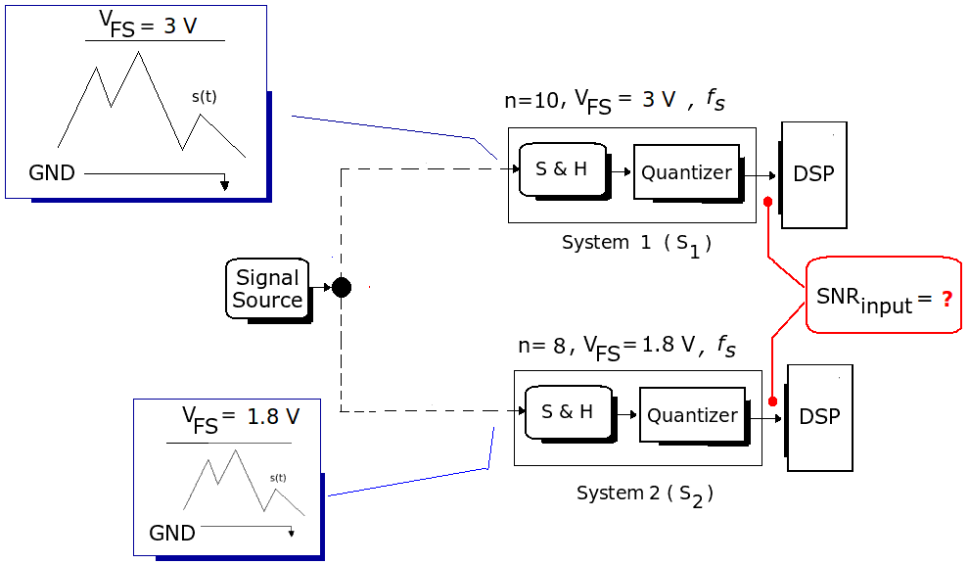


Figure 5.1: **Two different proposed data acquisition systems. Different input signal amplitude and number or bits yields lower power consumption of the system S_2. It should be noted that in order to reach the lowest power consumption level, we can produce SNR variations with adverse effects on the DSP, therefore, SNR analysis is required. It must be evidenced the SNR dependence on input signals and data acquisition system parameters for general cases.**

Each above mentioned power-aware or power-adaptive technique has been implemented rather ad hoc.

Even more serious, the use of data acquisition adaptive-systems (with the purpose of achieving node energy consumption reduction) produces SNR variations. Such variations regrettably can jeopardize the digital signal processing performance and it can not be properly analyzed due to the absence of models that capture the SNR dependence on the data acquisition system and signal features for general cases.

5.1.3 Chapter goal

The chapter's fundamental goal and novel contribution lies in the development of a low-complexity methodology to evaluate at system level, the SNR of data acquisition systems. The stochastic undertaken approach, with minimal assumptions, leads to a SNR expression that provides useful information to visualize the SNR dependence on signal and data acquisition system features for general cases. The core of the chapter is not only the SNR expression deduction but also to explore the expression capability for solving problems that inherently arise in wireless sensor nodes applications. We offer application examples and the accuracy of the SNR expression has been experimentally verified.

5.2 Random signals: overview

We consider that signals in WSN applications are generated by random electrical processes from sensors or radio transceivers.Generally speaking, it can be known beforehand only a few signal features (e.g. signal bandwidth, maximum signal amplitude but not the whole temporal behaviour).Therefore, the statistical approach for signal modeling seems to be the best procedure. Detailed studies of random signals are offered by a large amount of literature.
We present the theory as well as definitions and notations offered by **[Carlson, B. (1986)]** and **[Haykin,S. (1996)]**. Main concepts are summarized to be used as background material through this thesis dissertation.

Random processes can be represented as follow: $s_i(t) = s(t, X_i)$ where X_i is a particular value of the random variable x with probability density function $p(x)$ (the random process can be considered also as the experiment of the random variable x). This random variable x represents a signal feature, e.g. amplitude, phase or noise.
If we consider an instant of time t, the value $s_i(t) = s(t, X_i)$ is the mapping of the random value onto the real function of time $s(t, x)$.
The signal $s_i(t)$ is defined as member of all possible sample functions or **collection** or **ensemble**. Such member $s_i(t)$ of the collection is a random function and also is named **sample function**.
For instance, at time t_0 it holds $s_i(t_0) = s(t_0, X_i)$, where X_i is one particular value for the random variable x. It is possible to study the statistical properties of the random function $s(t_0, x)$ and it is defined as **ensemble's statistic**.

In random processes it is not possible to know which sample function (among the ensemble) is observed, because it is not possible to know beforehand the X_i value. Moreover, in literature for reducing notation the random value x is often omitted and the random process is expressed only as $s(t)$, but must be indeed as $s(t, x)$.
By using the previous definitions it is possible to calculate the most used statistical features, for instance, equation (5.2) is the **ensemble or statistical average** of the signal $s(t, x)$ at arbitrary time t **[Carlson, B. (1986)]**.

$$\overline{s(t,x)} \triangleq E[s(t,x)] = \int_{-\infty}^{\infty} s(t,x)p(x)dx \tag{5.2}$$

A more formal mathematical expression is:

$$\overline{s(t,x)} = \int_{-\infty}^{\infty} x f_{s(t)}(x)dx \tag{5.3}$$

The autocorrelation function is (5.4):

$$R_s(t_1,t_2) \triangleq E[s(t_1,x)s(t_2,x)] = \int_{-\infty}^{\infty} s(t_1,x)s(t_2,x)p(x)dx \tag{5.4}$$

A more formal mathematical expression is:

$$R_{s_1,s_2}(t_1,t_2) = \int\int_{\mathbb{R}^2} xy f_{s_1(t),s_2(t)}(x,y)dxdy \tag{5.5}$$

The mean square value is then: $R_s(t,t)$.
It is important to underline that the variable t (time) in the foregoing expression is constant during operations, however, the final value depends on time. Expressions above can be generalized to the case of random processes which depend on more than one random variable.For instance, for two random variables the autocorrelation becomes (5.6):

$$R_s(t_1,t_2) = E[s(t_1,x,y)s(t_2,x,y)] = \int_{-\infty}^{\infty} s(t_1,x,y)s(t_2,x,y)p(x,y)dx \tag{5.6}$$

Finally, the average or time-average operation over a interval of duration T is defined as equation (5.7):

$$\langle s(t,x)\rangle \triangleq \frac{1}{T}\int_{-T/2}^{T/2} s(t,x)dt \tag{5.7}$$

5.2.1 Definitions

In this section, we summarize from **[Carlson, B. (1986)]** the most important concepts widely used through this chapter.
The signal power of a **sample function** $s(t,x)$ within the time interval $[-\frac{T}{2},\frac{T}{2}]$ is defined as the following random variable (5.8):

$$P_T(x) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} s(t,x)^2 dt \tag{5.8}$$

The total signal energy is (5.9) :

$$\lim_{T\to\infty} TP_T(x) = \int_{-\infty}^{\infty} s(t,x)^2 dt \tag{5.9}$$

The average signal power is then $\overline{P} = \lim_{T\to\infty} E[P_T(x)]$, and under the hypothesis of ergodic process, the time integration and expectation can be interchanged, the expression for the average signal power becomes (5.10):

$$\overline{P} = \lim_{T\to\infty}\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} E[s(t,x)^2]dt = \langle E[s(t,x)^2]\rangle \tag{5.10}$$

To conclude this section,we observe two definitions provided by **[Carlson, B. (1986)]**.

Stationary random process: the signal can be considered as stationary random process when the signal features do not change over all time. The main properties of a wide-sense stationary process are:

- The autocorrelation only depends on time differences: $E[s(t_1,x)s(t_2,x)] = R_s(t_1-t_2)$. It is often expressed as $E[s(t,x)s(t-\tau,x)] = R_s(\tau)$.
- The mean value does not depend on t (time): $E[s(t,x)] = m_s$, (we observe abuse of notation in literature because it is expressed as $E[s(t)] = m_s$ but formally means $E[s(t,x)] = m_s$).
- The mean square value are constant and $\overline{s(t,x)^2} = E[s(t,x)^2] = R_s(0)$.
- The average power of the signal is constant and $\overline{P} = R_s(0) = \overline{s(t,x)^2} = \sigma_s^2 + m_s^2$ (where σ_s^2 is the variance and σ_s is the standard deviation).

Ergodic random process: the signal can be considered as an ergodic random process when the time averages and ensemble averages are equal. It can be expressed as: $\langle s(t,x)\rangle = E[s(t,x)]$, $\langle s(t,x)^2\rangle = E[s(t,x)^2]$ and $\langle s(t,x)s(t-\tau,x)\rangle = E[s(t,x)s(t-\tau,x)]$ ($\langle.\rangle$ indicates time average operation). An ergodic process is stationary but vice versa is not guaranteed, the stationary property does not assure ergodicity properties. Moreover, if the process is ergodic the following relations hold **[Carlson, B. (1986)]**:

- The signal mean value is the DC component $\langle s(t,x)\rangle = m_s$.
- The mean square value $\overline{s(t,x)^2} = \langle s(t,x)^2\rangle$ is the average power.
- The variance σ_s^2 represents the AC power of the time-varying signal.
- The standard deviation σ_s represents the RMS value.

5.2.2 Spectrum and filtered random signals

In order to complete this overview, we mention that the power spectral density for random stationary signals can be calculated as:

$$G_s(\omega) = \int_{-\infty}^{\infty} R_s(\tau) e^{-j\omega\tau} d\tau \tag{5.11}$$

And

$$R_s(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G_s(\omega) e^{j\omega\tau} d\omega \tag{5.12}$$

Finally, when a random signal $s(t, X_i)$ is filtered by a LTI filter $h(t)$, the resulting output $y(t, X_i)$ has autocorrelation and power spectral density as:

$$R_y(\tau) = h(\tau) * R_s(\tau) * h(-\tau)^* \tag{5.13}$$

And

$$R_y(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mid H(\omega) \mid^2 G_s(\omega) d\omega \tag{5.14}$$

5.3 A brief AD converter model overview

Many articles and books have been published describing the ADC functionality. It is not possible to include all of them, however, we select few references in order to be used as guide for this chapter. Particularly, for this section we use **[Plassche, R. (1987)]** as main guide.
As it is depicted in **Figure 5.2**, the process of analog-to-digital signal conversion involves two stages:

- sample and hold of the input signal $s(t)$.
- quantization and codification of the previous sampled signal.

During the analog-to-digital conversion, the input signal is first sampled and then quantized. The sample and hold process is executed at a given period Ts, where $Ts = 1/f_s$ (f_s is the sampling frequency). During the quantization process, the ADC selects the bits for which the binary word codification is nearest to the sampled signal. In other words, the process of quantization consists in rounding the sampled value into a finite number of discrete values. Such discrete values are defined in literature as codes.

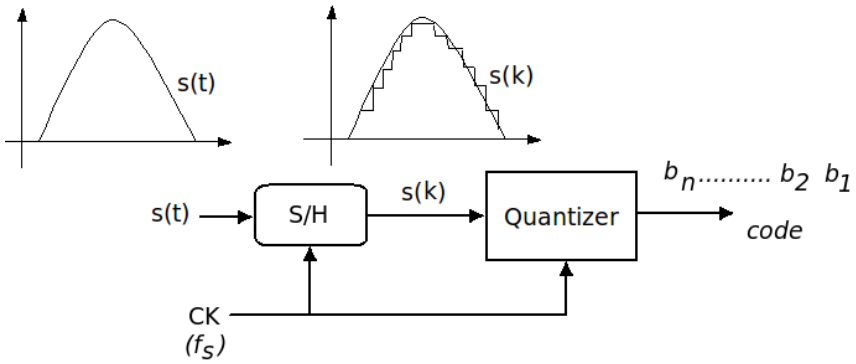


Figure 5.2: **The two stages of the analog-to-digital conversion process. The analog signal $s(t)$ is sampled and then quantized. The sampling frequency is f_s.**

The digital representation of the sampled input signal, named from herein s_k ($s_k = s(kT_s)$) is as follow:

$$s_k = \frac{V_{FS}}{2^n} \sum_{j=1}^{n} b_{j,k} 2^{j-1} + e_k \tag{5.15}$$

Where:

- n is the number of bits for data representation.
- V_{FS} is the full-scale voltage reference.

- e_k is the error or uncertainty that the quantization process introduces.This quantization error depends on the number of used levels for the quantization and in general such error is assumed uniformly distributed in the interval $[-q_s/2, q_s/2]$ ($e_k \in [-q_s/2, q_s/2]$).
- $q_s = V_{FS}/2^n$ is the quantization step.

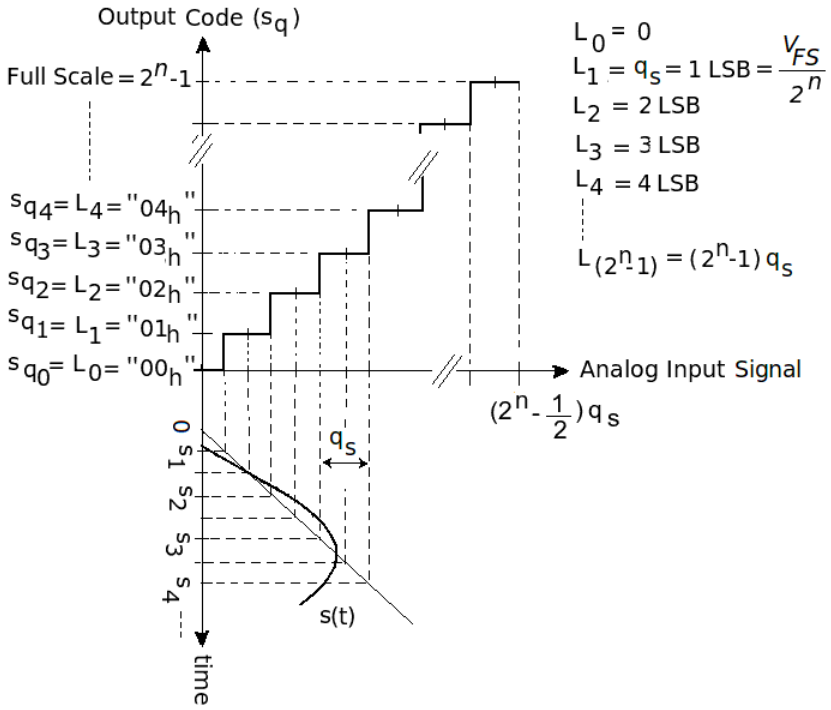


Figure 5.3: **Representation of the quantization process. The signal $s(t)$ is sampled and codified according to the quantizations levels. See also Figure 5.4**

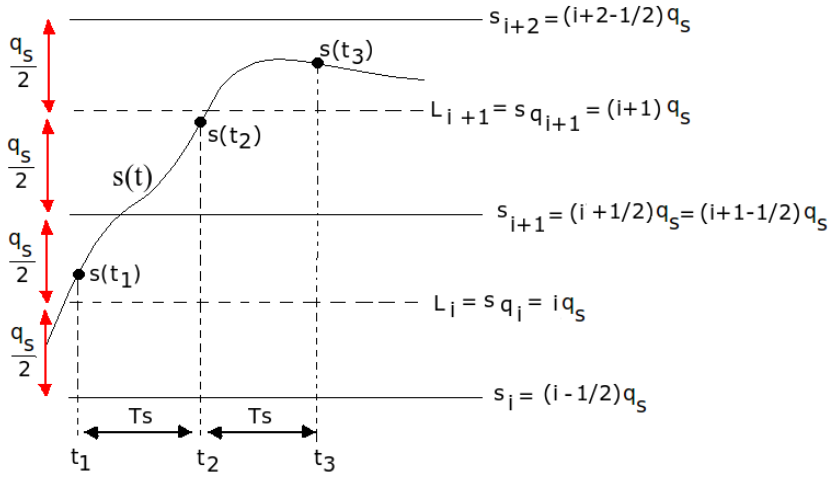


Figure 5.4: **Representation of the quantization process. The signal $s(t)$ is sampled and codified according to the quantizations levels.**

As is depicted in **Figure 5.4**, the signal $s(t)$ is quantized into the level $s_{q\,i}$ when $s_k \in [s_i, s_{i+1}] = [s_{q\,i} - q_s/2, s_{q\,i} + q_s/2]$. For instance, when $t = t_1$, $s(t_1) \longrightarrow s_{q\,i}$ (see **Figure 5.4**). However,when $t = t_1 + T_s = t_2$, $s(t_2) \longrightarrow s_{q\,i+1}$ and similarly when $t = t_1 + 2T_s = t_3$, $s(t_3) \longrightarrow s_{q\,i+1}$.

Under the hypothesis that q_s is small enough so that the number of quantization levels is large, the quantization error can be assumed additive,uniformly distributed and uncorrelated with s_k **[Plassche, R. (1987)]**, **[Gunst, A.W. (2004)]** i.e. white noise with uniform probability density function. Then, it is possible to express the mean-square quantization error as:

$$\sigma_q^2 = \int_{-q_s/2}^{q_s/2} e_k^2 \frac{1}{q_s} de_k = \frac{q_s^2}{12} \tag{5.16}$$

And the power density spectrum is:

$$S_e(f) = \frac{q_s^2}{12 f_s} \tag{5.17}$$

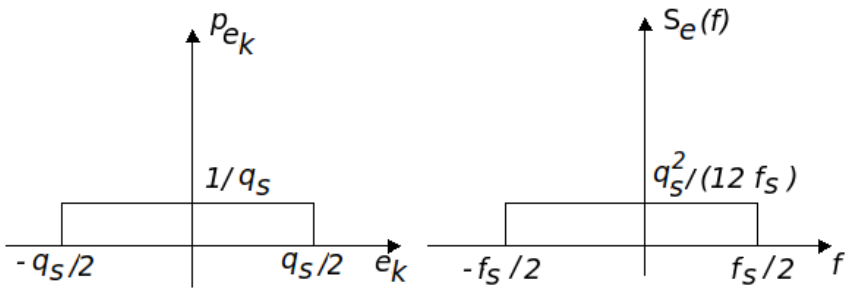


Figure 5.5: **Error probability density function (right) and error power density spectrum (left) of the quantization error under the hypothesis quantization error additive, uniformly distributed and uncorrelated with s_k.**

5.4 AD converter specifications

The purpose for using specifications is to quantify the ADC performance. In others words, designers must take into account specifications in order to select or to design the ADC architecture that achieves the desired application features.
The list of used ADC features and specifications is large and well described in literature, however the most frequently mentioned are: bit number for data representation, effective number of bits ($ENOB$), signal-to-quantization noise ratio (SNR), full voltage scale range (V_{FS}), sampling frequency (f_s), signal-to-noise ratio plus distortion (SDNR),differential non-linearity (DNL) and integral non-linearity (INL).
Nonetheless, available specifications' equation are deduced under particular cases, where simplifications are assumed in order to obtain closed expressions and formulations easy to handle.
It is due to such expressions are used to test the ADC under controlled situations and to compare its performance with others ADC regardless of the circuit architecture.

Several questions arise when we try to analysis the ADC performance under situations that do not match the hypothesis under which, the specifications' expressions have been deduced (e.g. in general, the signal behavior is not known beforehand in WSN scenarios and as we will see next, most used specifications reported in literature are deduced assuming a sinusoidal input signal). If our purpose is to dynamically adapt the ADC in order to obtain the best performance from the power consumption point of view and for different input signals, then, expressions reported in literature do not offer insight to develop methodologies to this end.
We offer the following example in order to get insight of the problem we face. In **Figure 5.6** (left) is depicted the $ENOB$ as function of the sampling frequency, voltage power supply and signal frequency (Figure 12 of **[AN693, App. Note]**). Under the assumption that the ADC is working at the point B (right), the associated $ENOB$ value can be obtained.
However, if the hardware platform experiment a progressive battery voltage decreases due to the depletion process (from $5V$ to $2.7V$), then, the ADC operating point B should be moved to the non-existent curve C and the effects on the $ENOB$ are not modeled in these graphs.
Instead, the ADC vendor provides the $ENOB$ for another sampling frequency (see curve A). It is clear then, that more complete expressions for the $ENOB$ are required. Generally speaking, it is required expressions that model the ADC behavior for general cases.

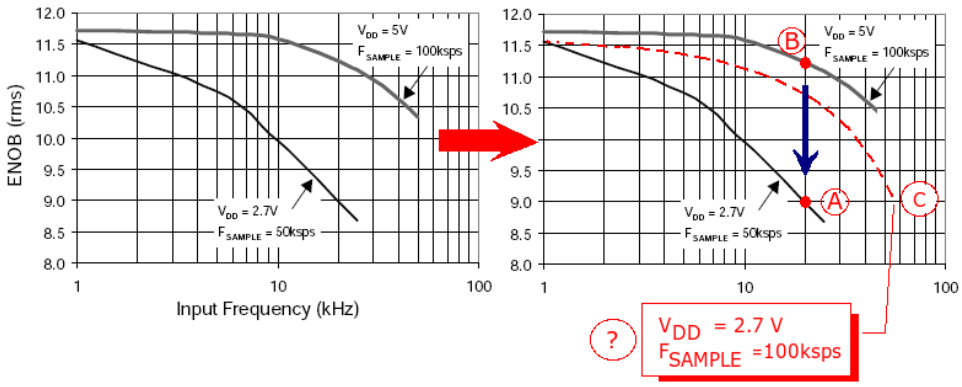


Figure 5.6: **Right: Figure 12 from [AN693, App. Note],** $ENOB$ **as function of the input signal frequency and parametric on the voltage power supply** V_{DD} **and sampling frequency** f_s **(in ksps). Left: we remark the non-existent curve C. This curve is required in order to evaluate the** $ENOB$ **when the** ADC **operating point changes due to the** V_{DD} **decrease.**

In order to implement power-aware data acquisition and signal conditioning systems, it is necessary to study the behavior of the whole data acquisition system under the tradeoffs which most of the hardware parameters are constrained.
For this we start the study with one of the most important specification: the SNR.

5.4.1 Signal to noise ratio (SNR)

This is the most used specification. It characterizes the ratio between the ADC input signal power to the noise power. The ADC has three main noise sources which are: quantization noise, jitter and electronic noise generated by the circuitry (for instance,thermal noise). Only in an ideal ADC the quantization noise is the only noise source.
In literature there are SNR expressions given by (5.18) **[Plassche, R. (1987)]**:

$$SNR(dB) = 6.02n - 1.25 + 10\log_{10}\frac{f_s}{f_w} \tag{5.18}$$

Where: f_w is the signal bandwidth, n is the number of bits and $f_s = 1/T_s$ is the sampling frequency. Equation (5.18) is used as a figure of merit to characterize or to compare the performance of different ADC architectures under controlled conditions. Such use is attained at the cost of the restrictive required assumptions for its deduction:

- the input signal is a deterministic sine wave with full-scale amplitude range
- the full-scale voltage reference (V_{FS}) is assumed to be fixed
- n must be at least 4
- the quantization noise is assumed to be white noise uncorrelated with the input signal and only it is considered noise spectral components within the signal bandwidth f_w.

When $f_s = 2f_w$, equation (5.18) becomes:

$$SNR(dB) = 6.02n + 1.76 \tag{5.19}$$

Equation (5.18) models the SNR of signals with stationary behavior because it is required for testing the ADC performance **[Plassche, R. (1987)]**. Therefore, it fails to model the effects produced by signals and data acquisition systems with time-varying attributes. Thus, for modeling the SNR behavior under such situations, a novel methodology is needed and the approach is faced in the following sections.

To conclude this section, few words about the $ENOB$ are required. In order to measure the SNR, an input sinusoidal signal is applied so that to obtain an output signal FFT as is drawn in **Figure 5.7**. Due to the presence of non-linearities in the ADC, the output signal suffers distortion and harmonics arise at the ADC output. Therefore, for ADC performance characterization, it is necessary to define the following two parameters which are more used than the SNR because they capture non-lineal behavior of the ADC, **but under the sine wave input signal**.

Spurious Free Dynamic Range ($SFDR$):

This specification is often named also as dynamic range because it gives insight of the minimum input signal that can be distinguished from others harmonics which have been created by distortion. It is defined as the ratio between the input signal amplitude to the level of the worst or largest harmonic present in the spectrum (more specifically, as is drawn in **Figure 5.7**, it is the largest distortion component).

Signal-to-Noise plus Distortion Ratio ($SNDR$):

It is defined as the ratio between the output signal average power to the total noise power at the ADC output. It is important to underline that the used power noise includes all noise sources and harmonics present in the ADC bandwidth, i.e. all effects that produce undesired spectrum components.

The $ENOB$ (Effective Number of Bits) is an specification used to compare the performance of different ADC (even if the ADCs have different circuit architectures). The expression for the $ENOB$ estimation comes from the SNR expression (5.19) and the $SNRD$ definition as follow:

$$ENOB = \frac{SNDR - 1.76}{6.02} \tag{5.20}$$

The $ENOB$ indicates that the ADC is equivalent to a perfect ADC of $ENOB$ number of bits.

It is important to remark that the previous expressions are always used when an input sine wave is applied to the ADC. As we have discussed previously, it is necessary to develop novel expression for more general cases. Hence, first of all we start to study one of the most important specification: the SNR.

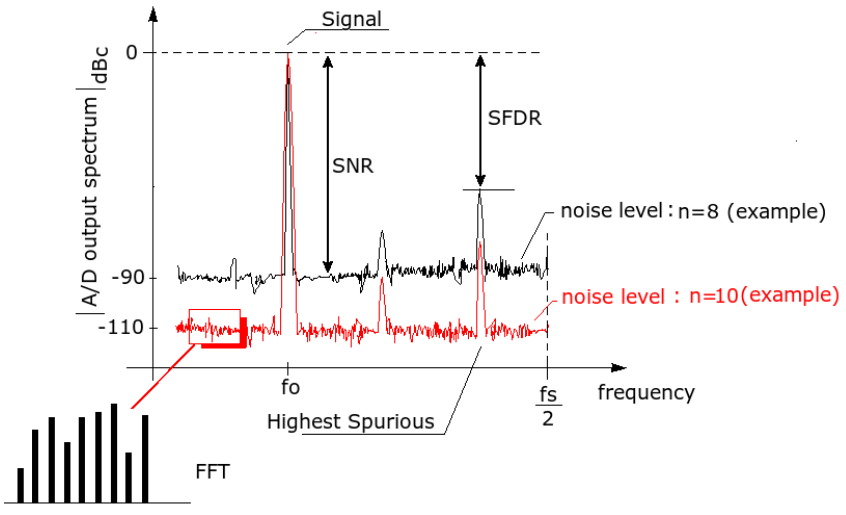


Figure 5.7: **Example of FFT computation for the ADC output characterization (spectral components).The input signal is a sine wave with amplitude value equals the full-scale voltage reference and frequency f_o.**

5.4.2 Power consumption

In addition to the previous described figures of merit, other two interesting figures of merit are presented below as they relate ADC features and power consumption (and for section conclusion), referring the interested reader to the reference **[Haddad, S.A.P. (2009)]**.
The first figure of merit quantify the power consumption P_{ADC}:

$$P_{ADC} = \frac{2^{ENOB} f_s}{F} \tag{5.21}$$

Where P_{ADC} is the power consumption and $F = 1.2\,10^{12}$ is considered a factor in the ADC present-day state-of-the-art (see **[Haddad, S.A.P. (2009)]**). The $ENOB$ is as (5.20).

The second figure of merit quantifies the quantization energy E_Q per conversion step:

$$E_Q = \frac{P_{ADC}}{2^n 2F_{BW}} \tag{5.22}$$

Where F_{BW} is the effective resolution bandwidth, (equals to f_s for a Nyquist ADC). The minimum published E_Q value is $E_Q = 2.8\,pJ$ (see **[Haddad, S.A.P. (2009)]** -Chapter 1).

5.5 Novel SNR evaluation methodology

In this section it is briefly explained the treatment for the methodology derivation. **Figure 5.8** introduces the considered signal and acquisition system architecture. The analogue input signal $s(t)$ is assumed to be a stochastic process within $[L_{min}, L_{min} + A_{pp}]$,i.e. $s(t) \in [L_{min}, L_{min} + A_{pp}]$.
We name A_{pp} and L_{min} as *signal attributes* or *signal features*, where L_{min} is the lower signal value and A_{pp} is the peak-to-peak signal amplitude.
They must satisfy: $0 \leq L_{min}$ and $L_{min} + A_{pp} \leq V_{FS}$.
Moreover, it must not be assumed any particular $s(t)$ wave shape beforehand, nor signal bandwidth. The probability density for $s(t)$ is assumed to be uniform $p(s) = A_{pp}^{-1}$. After sampling, the signal becomes a train of pulses (or random digital wave) with width d, $(d \ll Ts)$, as (5.23) expresses:

$$s_s(t) = \sum_k s_k p(t - kT_s) \tag{5.23}$$

Where s_k (or also $s(k)$) is the discrete-time representation of the signal $s(t)$ (or samples) and $f_s = 1/T_s$.
This formulation for signal sampling is well established and analyzed in literature **[Carlson, B. (1986)]**.

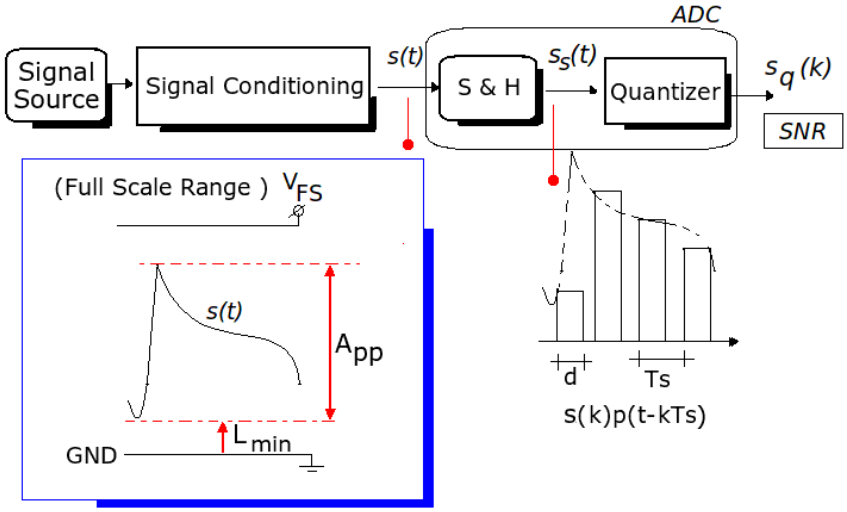


Figure 5.8: **Considered signal acquisition architecture and signal model.**

It is assumed that the samples s_k build a discrete-time stochastic process in the *mean-ergodic and correlation-ergodic mean square error sense* **[Haykin,S. (1996)]**.
We calculate the sampled signal average power:

$$R_s(0) = E[s_k, s_{k-\tau}]_{\tau=0} = \int_{L_{min}}^{A_{pp}+L_{min}} s_k^2 \frac{1}{A_{pp}} ds_k = \frac{(A_{pp}+L_{min})^3 - L_{min}^3}{3A_{pp}} \tag{5.24}$$

Where $s_k = s(kT_s)$ is the sampled signal. The samples s_k is an analog random variable that falls in the interval $[L_{min}, L_{min} + A_{pp}]$, but it is the discrete-time signal representation.

The DC power is $P_{DC} = m_s^2$ where m_s can be estimated as follows:

$$m_s = \int_{L_{min}}^{A_{pp}+L_{min}} s_k \frac{1}{A_{pp}} ds_k = \frac{(A_{pp}+L_{min})^2 - L_{min}^2}{2A_{pp}} = \frac{A_{pp}+2L_{min}}{2} \tag{5.25}$$

After sampling the quantization process takes place and the quantized signal results $s_{q,k} = s_k + e_k$. Where:

- e_k : quantization error, $e_k \in [-q_s/2, q_s/2]$
- q_s : quantization step, $q_s = LSB = V_{FS}/2^n$ (n is the number of bits for data representation and V_{FS} is the full-scale voltage reference.)

We define the SNR in terms of autocorrelation as follows:

$$SNR = \frac{R_s(0)}{\sigma_q^2} = \frac{(A_{pp}+L_{min})^3 - L_{min}^3}{3A_{pp}\sigma_q^2} \tag{5.26}$$

Where σ_q^2 is the mean-square quantization error or quantization error average power.
Two critical issues deserve attention:

- Please, note that for the used SNR expression it is not required hypothesis about the statistical relation between the sampled signal and the quantization error. It is due to the SNR is not defined in terms of the sampled and quantized signal. Contrarily, the SNR is defined in terms of the sampled signal. We believe that it is better than that of conventional approach. The prediction capability of (5.26) is demonstrated by a number of experiments.
  Detailed study of quantization error is addressed in [**Wadgy, M.F. (1989)**], [**Wong, P.W. (1990)**] and [**Gray, R. M. (1990)**].
- This approach does not take into account the relation between the quantization noise and the sampling frequency.
  It has been recently revealed through simulations [**Perez-Alcazar, P.R. (2002)**].

5.6 Mean-square quantization error estimation

According to literature **[Peebles, P. (1987)]**, the mean square quantization error for the signal sample s_k within the $i-th$ interval $[s_i, s_{i+1}]$ can be calculated as (5.27):

$$\sigma_{q,i}^2 = E[(s_k - s_{q,i})^2 /_{s_k \in [s_i, s_{i+1}]}] = \int_{s_i}^{s_{i+1}} (s_k - s_{q,i})^2 p(s_k/[s_i, s_{i+1}]) ds_k \tag{5.27}$$

Where s_k is a signal sample but it must be considered as a continuous random variable for the integration process and $p(s_k/[s_i, s_{i+1}])$ is the conditional probability density of s_k given that s_k falls onto $[s_i, s_{i+1}]$. Please, observe that the index i in changes according to the interval to which the sampled signal s_k belongs $s_k \in [s_i, s_{i+1}]$,then,the quantization level $s_{q,i}$ depends on the sample s_k. The conditional probability density can be estimated as follows **[Peebles, P. (1987)]** by using the Bayes' theorem:

$$p_s(s_k/[s_i, s_{i+1}]) = \frac{p(s_k)}{P_i} \quad \texttt{for} \quad s_k \in [s_i, s_{i+1}] \tag{5.28}$$

Where $p(s_k)$ is the previously presented probability signal density and P_i is the probability that the sampled signal s_k falls onto $[s_i, s_{i+1}]$.

$$P_i = \int_{s_i}^{s_{i+1}} p(s_k) ds_k \tag{5.29}$$

An important issue deserves attention.If the quantization step $q_s = [s_{i+1} - s_i]$ is small enough, P_i approaches to:

$$P_i = \int_{s_i}^{s_{i+1}} p(s_k)ds_k \approx p(s_{q,i})q_s \tag{5.30}$$

However, we observe that is not possible to reach some useful closed expression in absence of hypothesis about the signal probability density. As it has been previously discussed, we assume uniform distribution for the samples s_k, i.e. $p(s_k) = A_{pp}^{-1}$. Therefore:

$$P_i = \int_{s_i}^{s_{i+1}} p(s_k)ds_k = \frac{q_s}{A_{pp}} \tag{5.31}$$

The next task is to calculate the mean square quantization error, which is:

$$\sigma_q^2 = \sum_{i=1}^{N_L} \sigma_{q,i}^2 P_i = \int_{L_{min}}^{L_{min}+A_{pp}} (s_k - s_{q,i})^2 p(s_k)ds_k \tag{5.32}$$

(To have better insight of the written expressions in this section, readers should observe **Figure 5.9**). Where N_L is the number of quantization levels used to describe the signal,more specifically N_L is the number of levels between s_{m+1} and s_h, i.e. $s_h - s_{m+1} = N_L q_s$. Therefore it is easy to see that $N_L q_s \leq A_{pp} \leq (N_L + 2)q_s$ and the number of required bits n_R to represent the signal must satisfy the following expression:

$$2^{n_R - 1} \leq N_L \leq 2^{n_R} \tag{5.33}$$

Expression 5.32 can be restated as:

$$\sigma_q^2 = \int_{L_{min}}^{s_{m+1}} (s_k - s_{q,m})^2 p(s_k)ds_k + \sum_{i=m+1}^{h-1} \int_{s_i}^{s_{i+1}} (s_k - s_{q,i})^2 p(s_k)ds_k + \int_{s_h}^{L_{min}+A_{pp}} (s_k - s_{q,h})^2 p(s_k)ds_k \tag{5.34}$$

Where:

- m value such that: $s_m \leq L_{min} \leq s_{m+1}$ or $L_{min} \in [(m - 1/2)q_s, (m + 1/2)q_s]$
- h value such that: $s_h \leq L_{min} + A_{pp} \leq s_{h+1}$ or $L_{min} + A_{pp} \in [(h - 1/2)q_s, (h + 1/2)q_s]$

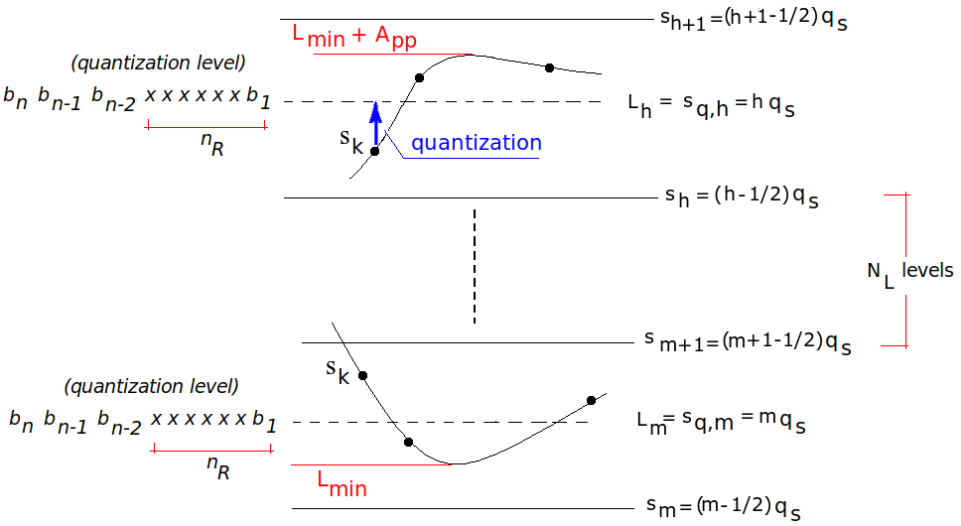


Figure 5.9: **Sampled signal s_k and its upper and lower boundaries for the mean square quantization error estimation. The quantization level $s_{q,h}$ is codified by using a n bits binary word.**

Equation 5.34 can be separated in three terms $\sigma_q^2 = \sigma_{q,A}^2 + \sigma_{q,B}^2 + \sigma_{q,C}^2$. We proceed to calculate each one, then for $\sigma_{q,A}^2$ we have that:

$$\sigma_{q,A}^2 = \int_{L_{min}}^{s_{m+1}} (s_k - s_{q,m})^2 p(s_k) ds_k = \frac{1}{3A_{pp}} \left[(s_k - s_{q,m})^3 \right]_{L_{min}}^{s_{m+1}} = \frac{1}{3A_{pp}} \left[(s_{m+1} - s_{q,m})^3 - (L_{min} - s_{q,m})^3 \right] \tag{5.35}$$

Thus:

$$\sigma_{q,A}^2 = \frac{1}{3A_{pp}} \left[\frac{q_s^3}{8} - (L_{min} - s_{q,m})^3 \right] \tag{5.36}$$

The most complex terms is:

$$\sigma_{q,B}^2 = \sum_{i=m+1}^{h-1} \int_{s_i}^{s_{i+1}} (s_k - s_{q,i})^2 p(s_k) ds_k = \sum_{i=m+1}^{h-1} \int_{(i-1/2)q_s}^{(i+1/2)q_s} (s_k - iq_s)^2 p(s_k) ds_k \tag{5.37}$$

Therefore:

$$\sigma_{q,B}^2 = \frac{1}{3A_{pp}} \sum_{iexpression()=m+1}^{h-1} \left[(s_k - iq_s)^3 \right]_{(i-1/2)q_s}^{(i+1/2)q_s} = \frac{1}{3A_{pp}} \sum_{i=m+1}^{h-1} \left[\frac{(q_s)^3}{8} - \frac{(-q_s)^3}{8} \right] \tag{5.38}$$

Finally, we obtain:

$$\sigma_{q,B}^2 = \frac{1}{3A_{pp}} \frac{N_L q_s^3}{4} \tag{5.39}$$

The last term is:

$$\sigma_{q,C}^2 = \int_{s_h}^{L_{min}+A_{pp}} (s_k - s_{q,h})^2 p(s_k) ds_k = \frac{1}{3A_{pp}} \left[(s_k - s_{q,h})^3 \right]_{s_h}^{L_{min}+A_{pp}} \tag{5.40}$$

$$\sigma_{q,C}^2 = \frac{1}{3A_{pp}} \left[(L_{min} + A_{pp} - s_{q,h})^3 - (s_h - s_{q,h})^3 \right] \tag{5.41}$$

Thus:

$$\sigma_{q,C}^2 = \frac{1}{3A_{pp}} \left[(L_{min} + A_{pp} - s_{q,h})^3 - \frac{(-q_s)^3}{8} \right] \tag{5.42}$$

Finally, we reach the complete expression and final result for $\sigma_q^2 = \sigma_{q,A}^2 + \sigma_{q,B}^2 + \sigma_{q,C}^2$:

$$\sigma_q^2 = \frac{1}{3A_{pp}} \frac{(N_L + 1)q_s^3}{4} + \frac{(L_{min} + A_{pp} - s_{q,h})^3 - (L_{min} - s_{q,m})^3}{3A_{pp}} \tag{5.43}$$

The previous equation (5.43) has the shortcoming that N_L, $s_{q,m}$ and $s_{q,h}$ have to be estimated as a function of the input signal. For this, we use the following approximation:

$$\frac{(L_{min} + A_{pp} - s_{q,h})^3 - (L_{min} - s_{q,m})^3}{3A_{pp}} \leq \frac{(q_s)^3 + (q_s)^3}{3A_{pp}} = \frac{2q_s^3}{3A_{pp}} \tag{5.44}$$

Finally, we can use:

$$\sigma_q^2 = \frac{1}{3A_{pp}} \frac{(N_L + 9)q_s^3}{4} \tag{5.45}$$

There is another possible approximation if q_s is small enough (or N_L large enough) such that $N_L \gg 9$ and $N_L q_s \approx A_{pp}$, equation (5.45) reaches the well-known estimation for the average quantization noise power:

$$\sigma_q^2 = \frac{q_s^2}{12} \tag{5.46}$$

At the end, the SNR evaluation results:

$$SNR = \frac{R_s(0)}{\sigma_q^2} = \frac{(A_{pp} + L_{min})^3 - L_{min}^3}{3A_{pp}\sigma_q^2} \tag{5.47}$$

With σ_q^2 as (5.45) or (5.46).

5.6.1 Sine wave case: comparison

The expression (5.19) evaluates the $SNR(dB)$ for an input sinusoidal waveform with zero DC component as it has been explained above. In this section we compare results provide by the equation (5.19) and by our methodology. Formally speaking, the comparison is not correct at all, because our approach is for random signals and the input sinusoidal signal used for (5.19) is deterministic.
Nevertheless, this comparison is useful to give an idea of the error caused by using the expression (5.47) to estimate the SNR of deterministic input signals. For this, we consider a sine wave with $A_{pp} = 2V_{FS}$, $L_{min} = -V_{FS}$ and σ_q^2 as (5.46) with $q_s = \frac{2V_{FS}}{2^n}$. Then, expression (5.47) becomes:

$$SNR = \frac{(2V_{FS} - V_{FS})^3 + V_{FS}^3}{3.2.V_{FS}\left(\frac{2V_{FS}}{2^n}\right)^2 \frac{1}{12}} = 4^n \frac{2V_{FS}^3}{2V_{FS}^3} \tag{5.48}$$

Hence:

$$SNR = 4^n \tag{5.49}$$

In dB:

$$SNR(dB) = 10\log_{10}(4^n) = 6.02n \tag{5.50}$$

Table 5.1: $SNR(dB)$ **as function of the number of bits. Table 1.1 from [Plassche, R. (1987)] where results provided by the expression (5.50) have been added.**

Bits Number	1	2	3	4	5	6	7	8	9	10
Estimate [1]	6.31	13.30	19.52	25.59	31.65	37.70	43.76	49.82	55.87	61.93
$6.02n + 1.76$	7.78	13.80	19.82	25.84	31.86	37.88	43.90	49.92	55.94	61.96
expression (5.50) [2]	6.02	12.04	18.06	24.08	30.1	36.16	42.14	48.16	54.18	60.2

[1] Interested readers can read the complete numerical estimation in **[Plassche, R. (1987)]**.
[2] In a real situation, the input waveform should be considered with non deterministic behavior because the sampling starts at unknown phase, i.e. the phase φ is a random variable with uniform distribution in $[0, 2\pi]$

Few words about ADC with reduced number of bits (less than 4) deserve attention. In general, the most used ADC features a number of bits higher than 4. Nevertheless, recently in literature have arisen novel applications where the use of reduced number of bits yields an advantage. In **[Chen, S.W.M. (2007)]** authors discuss the implementation of a subsampling radio architecture for ultrawideband communications. It is discussed how reduced number of bits can be used for input signal with low SNR because does not degrade the system performance in this kind of RF front end.
On the other hand, in **[O'Driscoll, S. (2005)]** authors present a $100kS/s$ SAR whose resolution varies from 3 to 8 bits. This resolution can be modified in order to enhance neural signal content, at the same time the adaptive resolution reduces power consumption.
Such novel results are in line with the goal of this thesis.

5.7 SNR contour lines

Equation (5.47) defines SNR contour lines as it is depicted in **Figure 5.10**.The SNR contour lines are the loci of the signal attributes and data acquisition system features which related by (5.47) achieve the same SNR value.
A numerical first-hand example is depicted in **Figure 5.10**. The curve A is the $SNR = 53\,dB$ contour line for $V_{FS} = 3\,V$ and $n = 10$ (the value $53\,dB$ was selected for convenience of presentation).Likewise, the curve B is the $SNR = 53\,dB$ contour line for $V_{FS} = 1.8\,V$ and $n = 8$. It is possible to visualize mechanisms to adapt the data acquisition system in order to maintain or to enhance the SNR value in adverse conditions.
Particularly, in **Figure 5.10**, if the sensor signal and the associated signal conditioning circuit and data acquisition system work at point Q_1 and for instance, if $V_{FS} = 1.8\,V$ is required to supply the rest of the

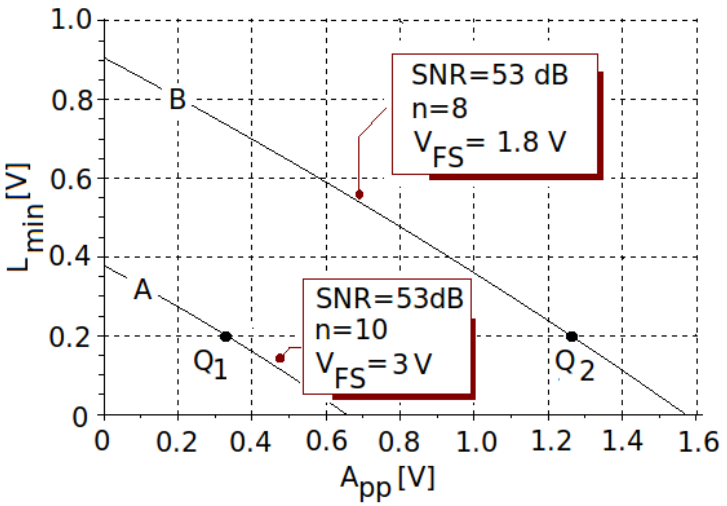


Figure 5.10: **Two examples of SNR contour lines obtained with different signals and data acquisition system features.Figure 5.11 below shows an indicative schema.**

circuitry (generally speaking the embedded wireless microsystem) which would have been built by using scaled CMOS technology, then the signal amplitude and the number of bits must be adapted to reach the operating point Q_2 with the same SNR value at the digital signal processing (DSP) input stage.

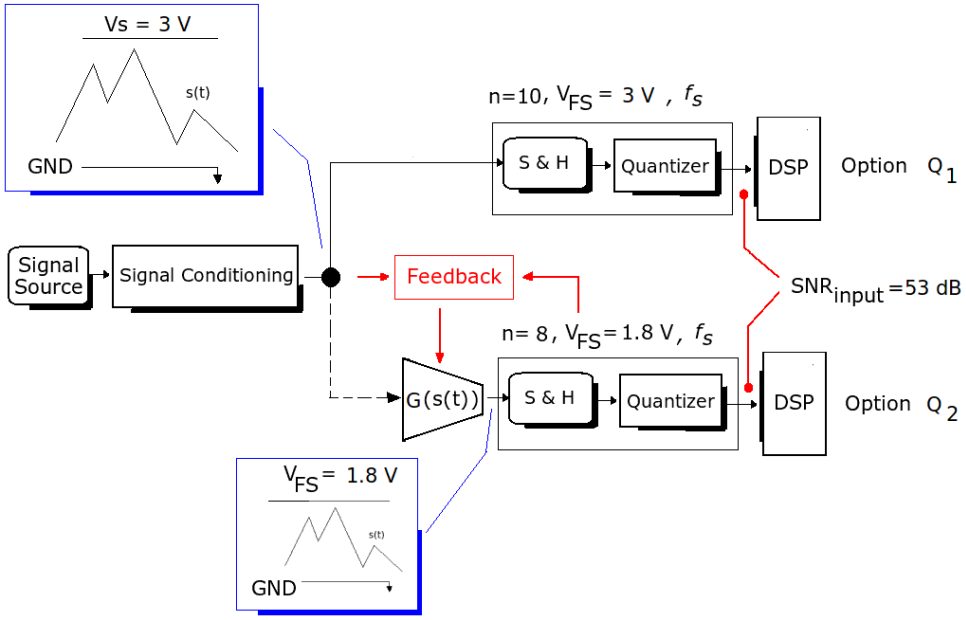


Figure 5.11: **To avoid the voltage matching problem, a large input signal and an accommodation in the number of bits are required (see also Figure 5.10).It should be added a circuit component capable of implementing the adjustment which in this case consist of an amplification.**

It is important to highlight that the block G could implement any possible transformation within the contour lines, i.e. linear gain or compression as well signal distortion and L_{min} adjustment. An important idea deserve attention because it constitutes the core of this work. It is clear that the maximum SNR can be easily achieved when the signal amplitude meets the full scale voltage reference.
However, at the conceptual level our goal is not to maximize the SNR, instead, the strategy is to achieve and to maintain the minimum required SNR value (because it can change during wireless sensor node operation) and to study the circuits implementations to enable this.

5.8 Experimental validation

To assess the modeling capabilities of the methodology to evaluate the SNR, an experimental evaluation has been conducted. Below, it is briefly described the conducted procedure for measurements, as well as the experimental setup and the signals which have been involved in the experimental process.

5.8.1 Methodology

Previously we have deduced that the SNR at the digital signal processing input can be evaluated as:

$$SNR = \frac{R_s(0)}{\sigma_q^2} = \frac{(A_{pp} + L_{min})^3 - L_{min}^3}{3A_{pp}\sigma_q^2} \tag{5.51}$$

Where we use SNR_E as notation which means *estimated SNR*.

$$SNR_E = \frac{(A_{pp} + L_{min})^3 - L_{min}^3}{3A_{pp}\sigma_q^2} \tag{5.52}$$

The measurement goal is to compare the SNR_E value with the measured SNR which we define as SNR_M. In next section it is explained how the measured SNR_M has been carried out.

$$SNR_M = \frac{R_{sq}(0)}{\sigma_q^2} \tag{5.53}$$

The methodology to evaluate the SNR can be validated when it is verified that:

$$SNR_E \approx SNR_M \tag{5.54}$$

From equation (5.53) we observe that it is necessary to calculate $R_{sq}(0)$. Since the sampled signal has been assumed to be a discrete-time stochastic process in the mean-ergodic and correlation-ergodic mean square error sense (wide-sense stationary) **[Haykin,S. (1996)]**, then it is reasonable to assume that the quantized sampled signal can also be considered wide-sense stationary.
Therefore it can be used the time average to estimate the average power (equation (5.55)) of the discrete-time signal if the number of used samples N is large enough (strictly approaches infinity) **[Haykin,S. (1996)]**:

$$R_{sq}(0) = \frac{1}{N}\sum_{k=1}^{N} s_{q,k}^2 \tag{5.55}$$

Moreover, signal samples are taken with an ADC with $n = 10$ bits, then in such situation the approximation $\sigma_q^2 = q_s^2/12$ holds (see also expressions (5.45) and (5.46)) and equation (5.52) becomes:

$$SNR_E = 4\frac{(A_{pp} + L_{min})^3 - L_{min}^3}{A_{pp}q_s^2} \tag{5.56}$$

The experimental validation is then performed by considering the following expressions:

$$SNR_M = \frac{\frac{1}{N}\sum_{k=1}^{N} s_{q,k}^2}{\frac{q_s^2}{12}} \tag{5.57}$$

$$SNR_E = 4\frac{(A_{pp} + L_{min})^3 - L_{min}^3}{A_{pp}q_s^2} \tag{5.58}$$

5.8.2 Experimental setup

The *Agilent 33250A 80MHz Function Arbitrary Waveform Generator* (see **Figure 5.12**) has been used as signal source. It can generate complex output waveforms because the modulated signals can be built with carriers and modulating signals of any wave shape.
Signals have been sampled by the MicaZ node which features a microcontroller Atmega128L with an integrated 10-bit successive approximation ADC. An embedded driver has been written in nesC to manage the sensor node. The node was plugged in to a PC in order to deliver data for analyses.

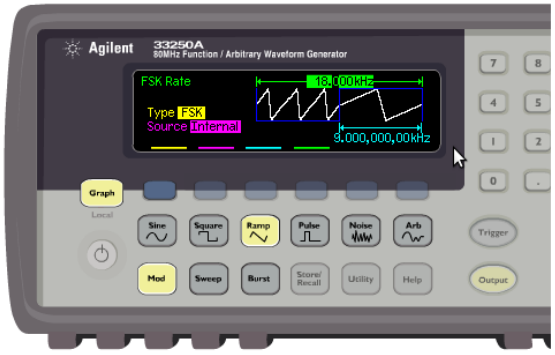


Figure 5.12: **Demo from http://wireless.agilent.com/flash/33250/index.html . The Agilent 33250A user interface.**

5.8.3 Signal selection

First step: we plot the SNR contour line for a given SNR value. Arbitrarily it has been selected $SNR = 65dB$ (see **Figure 5.13**).On the plotted contour line, we arbitrarily selected five points whose features L_{min} and A_{pp} are indicated in **Table 5.2**.
Second step: it consists in the waveforms generation for each marked point. We have used waveforms built with AM, FM and FSK modulations in order to obtain non trivial waveforms. To have better insight, we next specifically explain the procedure for the point A.

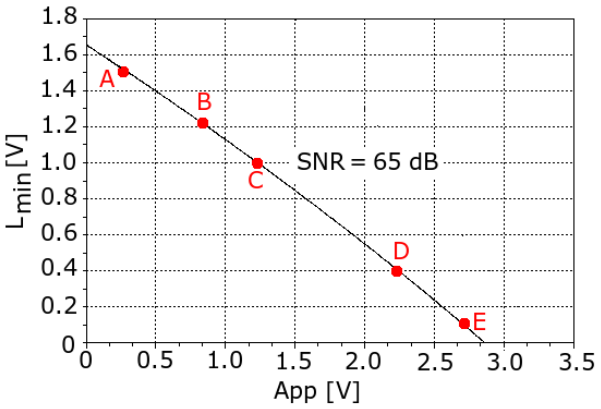


Figure 5.13: **Plotted SNR contour line for $SNR = 65dB$. It has been selected the 5 points with the used signal features for the experimental validation.**

Table 5.2: **The five selected points' attributes on the plotted contour lines (Figure 5.13)**

	Point A	Point B	Point C	Point D	Point E
$A_{pp}\,[V]$	0.27	0.84	1.23	2.25	2.70
$L_{min}\,[V]$	1.52	1.22	1.00	0.40	0.10

For point A, by using the *Agilent 33250A Waveform Generator*, we built 3 different modulated waveforms which have the same point A attributes, i.e A_{pp} and L_{min}. **Table 5.3** shows in the first row, the implemented AM waveform for the point A with the associated modulating parameters.
(The modulation parameters indicated in Tables are the nomenclature offered by the *Agilent 33250A* in the user interface screen) . Similarly, **Table 5.4** (first row) shows the implemented FM waveform which has the same point A attributes. Finally, **Table 5.5** in the first row, shows the implemented FSK modulated waveforms with L_{min} and A_{pp} as point A. We selected these modulated waveforms because they are the most complex waveforms capable of being built by the waveform generator, and they are

not easily described and mathematically tractable, then such waveforms could be interpreted as random signals.

Table 5.3: **Case 1, built and measured AM waveforms for the five points marked in the plotted SNR contour line (see Figure 5.13).**

	Carrier Sine	Shape	AM Freq.	AM Depth.
Point A	$600\,Hz$	down ramp	$350\,Hz$	110 %
Point B	$900\,Hz$	triangle	$500\,Hz$	60 %
Point C	$500\,Hz$	triangle	$160\,Hz$	100 %
Point D	$850\,Hz$	square	$400\,Hz$	40 %
Point E	$1\,kHz$	sine	$860\,Hz$	100 %

Table 5.4: **Case 2, built and measured FM waveforms for the five points marked in the plotted SNR contour line (see Figure 5.13).**

	Carrier Type	Shape	FM Freq.	Freq. Dev.
Point A	triangle, $5\,kHz$	sine	$500\,Hz$	$300\,Hz$
Point B	triangle, $1\,kHz$	sine	$500\,Hz$	$300\,Hz$
Point C	sine, $1\,kHz$	triangle	$200\,Hz$	$180\,Hz$
Point D	sine, $2\,kHz$	sine	$800\,Hz$	$800\,Hz$
Point E	triangle, $1\,kHz$	sine	$500\,Hz$	$300\,Hz$

Table 5.5: **Case 3, built and measured FSK waveforms for the five points marked in the plotted SNR contour line (see Figure 5.13).**

	Carrier Type	FSK rate	Hop. Freq.
Point A	sine, $750\,Hz$	$250\,Hz$	$10\,Hz$
Point B	Ramp, $1\,kHz$ (symmetry 40%)	$250\,Hz$	$10\,Hz$
Point C	sine, $1\,kHz$	$1\,kHz$	$100\,Hz$
Point D	sine, $1.5\,kHz$	$100\,Hz$	$150\,Hz$
Point E	ramp, $1\,kHz$	$100\,Hz$	$150\,Hz$

5.8.4 Results

The signal average power value has been estimated in the PC by using equation (5.57),where the number of used samples for our case is $N = 1000$.
Finally, measurement results are offered in **Table 5.6** and **5.7**. For instance, for the L_{min} and A_{pp} values indicated for the point A in **Table 5.2**, we observe that for the Case 1 (waveform obtained with AM modulation) the measured SNR results $64.88\,dB$ (see **Table 5.6**). As we can see, the methodology has accurately assessed the SNR value for all cases.

Table 5.6: **SNR_M for the five points marked on the plotted SNR contour line (see Figure 5.13).The SNR_E value for all cases is $65\,dB$. The sampling frequency is $15.93\,kHz$.**

Waveform	Point A	Point B	Point C	Point D	Point E
Case 1	64.88	64.84	64.99	64.60	64.23
Case 2	64.89	64.95	64.99	66.15	65.36
Case 3	64.90	64.89	64.99	64.86	65.32

For the Case 1 and Case 3 we also have performed measurements at the sampling frequency $f_s = 3.98\,kHz$ and results are offered in **Table 5.7**.

Table 5.7: SNR_M **for the five points marked on the plotted** SNR **contour line (see Figure 5.13). The** SNR_E **value for all cases is** $65\,dB$**.Sampling frequency** $f_s = 3.98\,kHz$**.**

Waveform	Point A	Point B	Point C	Point D	Point E
Case 1	64.88	64.83	64.78	64.62	64.26
Case 3	64.90	64.89	64.99	64.86	65.32

5.9 Real signals

In the previous section we have dealt with artificially built signals. In the following section we offer actual sensor signals with the associated SNR evaluation.

5.9.1 Example 1: sound signals

The Micaz node features the MTS300CB/MTS310CB sensor board with the miniature electret Microphone WM-62A (Panasonic).In **Figure 5.14** it is depicted an example of the microphone output voltage signal captured by the node.
This sound recording has been performed as part of the work presented in **[Lopez, M.A. (2008)]**, related to soft acoustic event detection for wireless sensor nodes applications. The sound is produced by an object that impacts on the floor.The signal starts in the stationary state while environmental sound is being recorded (marked as A). The spike or signal stream marked as B is the first object impact on the floor. The two next identified spikes (marked as C and D) indicate that the object bounced on the floor twice. For such recorded sound,the signal features and the used data acquisition parameters are: $n = 10$, $f_s = 7.95\,kHz$, number of samples $N = 3200$, $V_{FS} = 3.3\,V$,$L_{min} = 2.92\,V$ and $A_{pp} = 0.374\,V$. Since the number of bits is large enough, we use the approximation $\sigma_q^2 = q_s^2/12$. We evaluate the SNR in a time-windows of $400\,ms$ and the comparison between the measured and expected SNR values is:

- $SNR_E(dB) = 70.49dB$
- $SNR_M(dB) = 70.45dB$

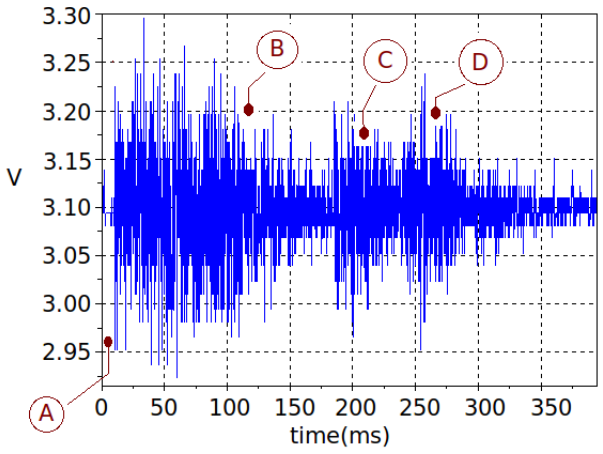


Figure 5.14: **Microphone output voltage profile recorded by the sensor node.**

5.9.2 Example 2: measure of acceleration signal (person walking and running)

The Micaz node features the MTS300CB/MTS310CB sensor board with the Low-Cost 2-g Dual-Axis Accelerometer ADXL202E **[ADXL202, Datasheet]**. The measurements consist in sampling the accelerometer output signal while the node is being carried by a person who walks, runs and takes a pause.The acceleration pattern has been captured without special accelerometer orientation, i.e. the node was carried with aleatory orientation in the shirt's pocket.

Two cases have been analyzed: case 1 (walk) and case 2 (walk and run) . The data acquisition features are: number of samples $N = 1000$, sampling frequency $f_s = 20\,Hz$,($T_s = 50\,ms$) and $V_{FS} = 3.3\,V$ ($q_s = 3.3\,V/1024 = 3.22\,mV$).

Case 1: walking man

In **Figure 5.15** it is depicted the accelerometer output signal captured by the node.The sections of the signal marked as A,C and E correspond to the normal walking; on the other hand the events B and D show two direction changes, i.e. the person change direction to take another path.
Signal features: $L_{min} = 1.27\,V$ and $A_{pp} = 0.27\,V$.
The measured and estimated SNR are as follows (see equation (5.57)):

- $SNR_E(dB) = 63.56dB$
- $SNR_M(dB) = 63.75dB$

Another important issue to verify is the number of required bits n_R. With reference to the **Figure 5.9** and expression (5.33), we have calculated for this example the representative number of bits n_R.
It results:

- In **Figure 5.15**, $L_{min} = 1.27\,V$ and its binary representation is 0110001001_b or 189_{hex} (remember that our AD provides 10 bits).
- In **Figure 5.15**, $L_{min} + A_{pp} = 1.54\,V$ and its binary representation is 0111011110_b or $1DE_{hex}$.

It is clear that only the 7-LSB change and therefore $n_R = 7$. We verify that the expression (5.33), re-written below holds:

$$2^{n_R-1} \leq N_L \leq 2^{n_R} \tag{5.59}$$

Then:

$$2^6 = 64 \leq N_L \simeq \frac{A_{pp}}{q_s} = 84 \leq 2^7 = 128 \tag{5.60}$$

This example shows that it is only required 7 bits to effectively describe the signal with $q_s = V_{FS}/1024 = 3.22\,mV$.

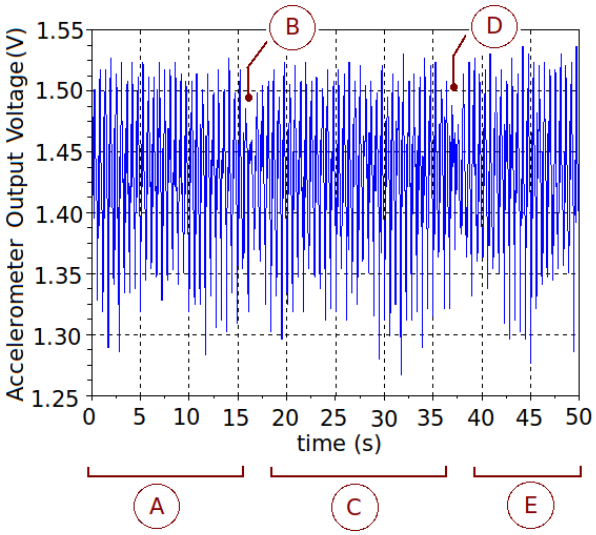


Figure 5.15: **Recorded person acceleration while walking.**

Case 2: walk and run

Description of **Figure 5.16**: A normal walking, B the person changes direction and immediately starts to run, C still running $10\,s$, D stop and long pause, E start to walk.
Data acquisition features: number of samples $N = 1000$,sampling frequency $f_s = 20\,Hz$, ($T_s = 50\,ms$)

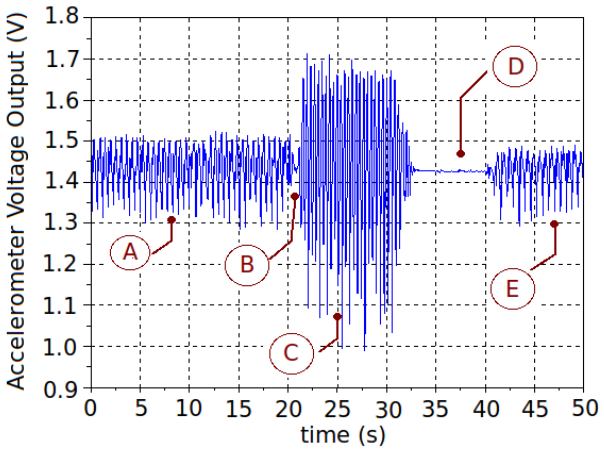


Figure 5.16: **Recorded Acceleration for Human Walk.**

and $V_{FS} = 3.3\,V$.Signal features: $L_{min} = 0.99\,V$ and $A_{pp} = 0.73\,V$.

The measured and estimated SNR are as follows:

- $SNR_E(dB) = 63.35dB$
- $SNR_M(dB) = 63.77dB$

Results shows that the methodology for SNR evaluation could well be used to approximate the SNR of complex sensor signals.

5.9.3 Example 3: ECG signals

In this section we present a case related to biological signals.The sampled signal has been taken from the free on-line available *PhysioBankATM* database (http://www.physionet.org/cgi-bin/ATM?database=aami-ec13), where a huge number of records containing sampled ECG signals are offered to the researchers community.

Case 1:

From such database, as first example we use the signal cataloged as: *Selected input record chfdb/chf07 (ECG), annotator ecg, from [12:12:00.000] to [12:12:10.000] - BIDMC Congestive Heart Failure Database (chfdb).* The file is provided in *.txt format, with the shown format in **Table 5.8** .

Table 5.8: **Record chfdb/chf07 (ECG),from *http://www.physionet.org***

Time (hh:mm:ss.mmm)	ECG (mV)
[12 : 12 : 00.000]	−0.515
[12 : 12 : 00.008]	−0.500
[12 : 12 : 00.012]	−0.480
[12 : 12 : 00.016]	−0.455
.....	.....
[12 : 12 : 09.996]	−0.435

We assume that the signal came from an electrode, and we amplify the signal with a gain $G = 2.23\,V/V$ and an offset is applied in order to move the signal into the band $[0, V_{FS}]$, after doing this, the signal is quantized. The results are depicted in **Figure 5.17**.
(The process is performed in Scilab by using the recorded signal in *.txt format, available from http://www.physionet.org)
This signal example is depicted in **Figure 5.17**. The number of samples is $N = 2500$, sampling frequency $f_s = 250\,Hz$ and the signal features are: $L_{min} = 0\,V$ and $A_{pp} = 3.3\,V$.
The measured and estimated SNR are as follows:

- $SNR_E(dB) = 66.22dB$
- $SNR_M(dB) = 67.22dB$

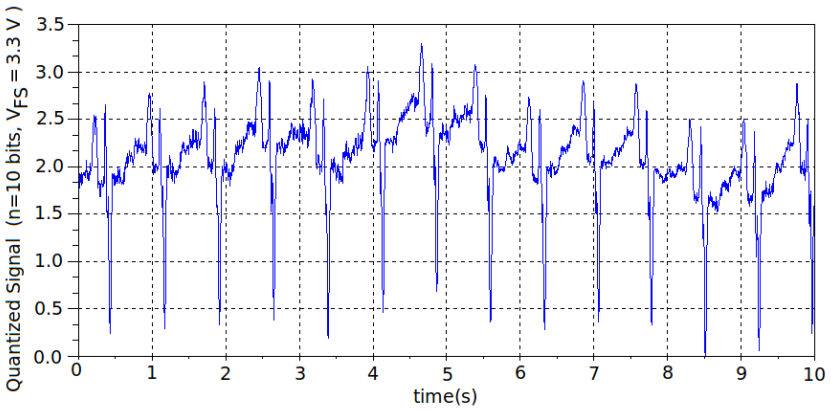


Figure 5.17: **Record chfdb/chf07 (ECG)**

Case 2: methodology limitation

From the database, as second example we use the signal cataloged as: *Selected input: record apnea-ecg/a01 (ECG), annotator apn, from 0:00.000 to 0:10.00 - Apnea-ECG Database (apnea-ecg).* The signal is depicted in **Figure 5.18**.
The number of samples is $N = 2500$, sampling frequency $f_s = 250\,Hz$ and the signal features are: $L_{min} = 0\,V$ and $A_{pp} = 3.3\,V$. The measured and estimated SNR are as follow:

- $SNR_E(dB) = 66.22dB$
- $SNR_M(dB) = 61.85dB$

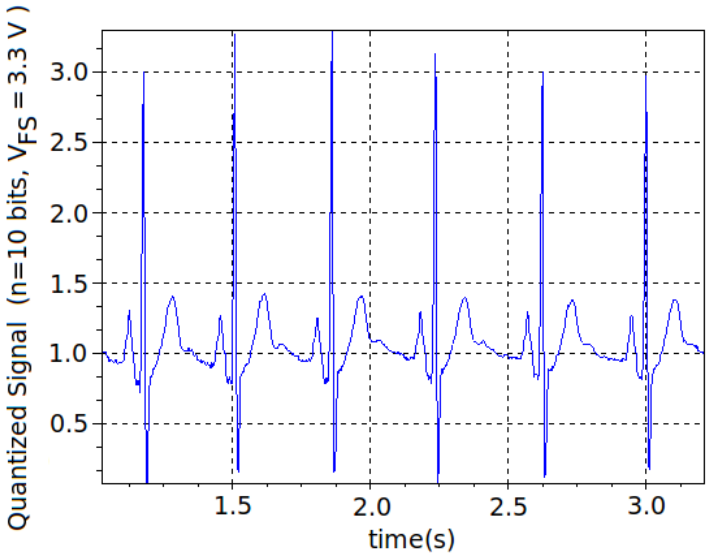


Figure 5.18: **Record apnea-ecg/a01 (ECG)**

We observe that in this case there is a difference of $4.36\,dB$ between SNR_M and SNR_E . The reason lies in the fact that the signal is clearly non-uniformly distributed in the interval $[L_{min}, L_{min} + A_{pp}]$. There are large spikes and the core of the signal (and the associated main power) is concentrated in a small range within $[L_{min}, L_{min} + A_{pp}]$.

For this kind of signal, it could be better the description with an asymmetric Gaussian probability density function $p(x)$ and further study is required.
Nevertheless, it is important to highlight that the methodology limitation has been identified.

5.10 Analysis example

In this section we highlight how the proposed methodology provides basis for solving problems during wireless sensor nodes design stage. We proceed to state an example with the associated solution.

5.10.1 Application example: selection of the number of bits

We suppose that the signal processing algorithm (implemented into the node) requires sensor signal samples with SNR above a given threshold, e.g. $SNR_{th} = 45\,dB$. The problem is to select the minimum number of bits for the AD converter in order to achieve such SNR_{th} in spite of battery voltage variations and signal amplitude variations within the range $0.4V_{FS} \leq A_{pp} \leq V_{FS}$. The solution is not provided in literature.
An approach for the solution can be derived from the proposed methodology because it can be obtained the SNR wort-case expression as a function of the normalized signal amplitude, i.e. the smallest achievable SNR (or lower bound) for all situations that holds:

$$SNR_{w.c.} = 4^{n+1}\left(\frac{A_{pp}}{V_{FS}}\right)^2 \leq SNR \tag{5.61}$$

Figure 5.19 shows the $SNR_{w.c.}$ behavior. Curve A satisfies the given constraint with $n = 8$ bits, as well as B with $n = 10$ bits. Then, $n = 8$ is the minimum number of bits that assures to met the SNR specification, i.e. the SNR value for 8 bits is above the curve A or $SNR_{w.c.}$ for $n = 8$ and consequently it satisfies the required SNR specification.

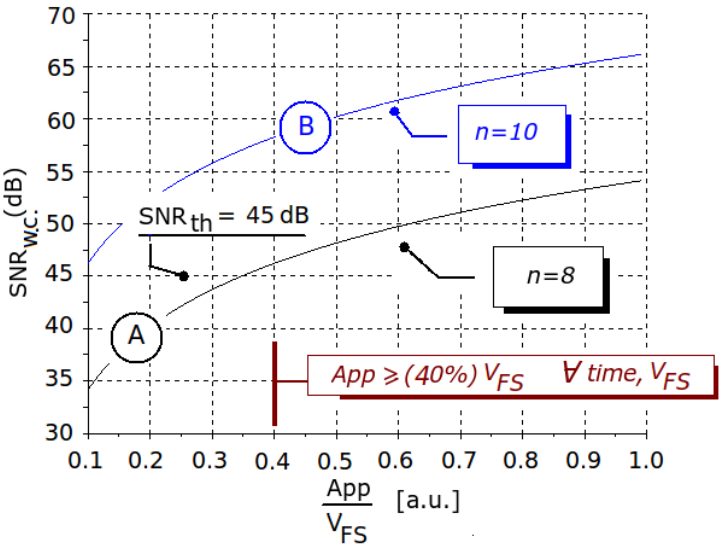


Figure 5.19: SNR **worst-case** ($SNR_{w.c.}$) **vs. normalized signal amplitude for different ADC number of bits** ($n = 8$ **and** $n = 10$)

5.10.2 Deduction of the expression (5.61)

We start the deduction of (5.61) by using the expression (5.56)

$$SNR_E = 4\frac{(A_{pp} + L_{min})^3 - L_{min}^3}{A_{pp}q_s^2} = 4^{n+1}\frac{(A_{pp} + L_{min})^3 - L_{min}^3}{A_{pp}V_{FS}^2} \tag{5.62}$$

The variables L_{min} and A_{pp} must fulfill the following two conditions:$L_{min} \in [0, V_{FS} - A_{pp}]$ and $0 \leq L_{min} + A_{pp} \leq V_{FS}$. Formally L_{min} and A_{pp} are not independent variables but since does not exist

the relation $A_{pp} = F(L_{min})$ it becomes impossible to use traditional minimization techniques such as Lagrange multipliers.
It is easy to see that SNR_E is a crescent function of L_{min} because its derivate yields:

$$\frac{\partial SNR_E}{\partial L_{min}} = \frac{4^{n+1}}{A_{pp}V_{FS}^2} 3\left\{(A_{pp} + L_{min})^2 - L_{min}^2\right\} \geq 0 \qquad \forall \quad L_{min} \tag{5.63}$$

Therefore, a local minimum occurs for $L_{min} = 0$ and the associated expression is the $SNR_{w.c.}$:

$$SNR_{w.c.} = 4^{n+1}\left(\frac{A_{pp}}{V_{FS}}\right)^2 \tag{5.64}$$

Then, we select the data acquisition features with the criterion:

$$SNR_{th} \leq SNR_{w.c.} = 4^{n+1}\left(\frac{A_{pp}}{V_{FS}}\right)^2 \tag{5.65}$$

5.10.3 Experimental verification

We perform an experimental validation. The goal of the measurements is to verify that, (with reference to the **Figure 5.19**) with $n = 10$ and under the condition $\frac{A_{pp}}{V_{FS}} \geq 0.4$ it is held that $SNR_M \geq SNR_{w.c.}$.
The number of bits $n = 10$ has been selected because our node can not features ADC with 8 bits.
Results are offered in **Table 5.9**.
To do this, the selected node input signal is a sine sweep waveform. The used sampling frequency is $f_s = 3.98\,kSamples/s$ and the methodology to calculate the SNR_E and SNR_M is the same that the used (and explained) for the previous empirical validation.
The experimental setup is built with the waveform generator and the node that takes signals samples and delivers them to the PC, where numerical calculations are performed. In addition to this, the node is powered by a variable voltage power supply, in order to be able to modify the V_{FS} value (see column 1 in **Table 5.9**).
The input signal features are arbitrarily modified in the waveform generator but being compliant with the condition $\frac{A_{pp}}{V_{FS}} \geq 0.4$ (see column 2, 3 and 4).
The shown results confirm that the SNR_M is above $SNR_{w.c.} = 45dB$ for every case with $n = 10$.

Table 5.9: **Measurements to verify the criterion (5.65).**

V_{FS} (V)	L_{min} (V)	A_{pp} (V)	$\frac{A_{pp}}{V_{FS}}$	SNR_E (dB)	SNR_M (dB)
3.3	1	2.0	0.61	67	67
3.0	1	1.6	0.53	66.84	66.77
2.7	1	1.8	0.67	67.96	68.41
2.4	1	1.0	0.42	67.07	67.3
3.3	0.8	2.5	0.76	67.37	67.61
3.0	0.8	2.0	0.67	66.97	67.07
2.7	0.8	1.8	0.67	67.37	67.46
2.4	0.8	1.0	0.42	65.88	65.93
3.3	0.2	3.1	0.94	66.5	67.01
3.0	0.2	1.2	0.4	60.26	60.61
2.7	0.2	1.35	0.5	61.99	62.22
2.4	0.2	1.9	0.8	65.50	65.99

5.11 Final considerations

This work deals with the problem of the SNR evaluation in wireless sensor nodes. The motivation arises because the use of data acquisition adaptive-circuits (with the purpose of achieving node energy consumption reduction) can produce SNR variations.
Such variations, regrettably can jeopardize the digital signal processing performance. Hence, for the purpose of analyzing the SNR dependence on signal and data acquisition system features for general cases, a novel methodology for estimating the SNR has been deduced and tested. The methodology uses

reduced hypothesis on the input signal and data acquisition features.
The proposed stochastic approach provides criteria to modify the signal features and data acquisition system parameters for enhancing the SNR or, to adapt the implemented signal processing algorithm accordingly to the estimation of when SNR adverse conditions will occur.
Moreover, it has been shown the existence of SNR contour lines that enable the study of tradeoffs between data acquisition features and input signal attributes.
Measurements show that the proposed methodology is simple to handle, but nonetheless useful to perform SNR evaluations for a large number of waveforms types.
An important issue deserves attention. It must be emphasized that the novel methodology for SNR evaluation is not a substitute nor a generalization of (5.18). Both expressions must be considered complementary because our SNR expression can not perform ADC characterization as others figures of merit effectively do.
On the other hand, the presented methodology presume the non-deterministic behavior of the signal. The SNR of a deterministic sine wave can not be effectively approximated by this methodology. Therefore, the limits of the methodology should be rigorously examined in terms of defining when a waveform can be considered as non-deterministic or stochastic random process.
Issues worthy of further examination are aimed at addressing more measurements taking into account variations of the signal features, the number of bits and V_{FS}.
Finally, the distortion effects that actual AD converters produce should be added to the methodology.

Chapter 6

Wavelets Assessment on Wireless Sensors Nodes: Case Study

Chapter Summary

This chapter presents a feasibility study for implementing wavelet transforms into wireless sensors nodes.
By using wavelet transforms it is possible to detect local features of measured signals, for instance, where the signal contains sharp spikes and their associated time duration.
In order to explore the signal frequency components and their time duration, wavelets with not widespread spectrum should be used. Unfortunately, these wavelets transforms are not hardware amenable and the digital implementation is very power demanding.
Nevertheless, there exist a very simple wavelet transform (Haar wavelet) whose digital implementation only requires few subtractions and additions, thus, it has been selected as potential candidate to be implemented into wireless sensor nodes.
Implementation issues are presented and discussed. We have explored the detection capability by thresholding of the digital Haar wavelet transform. Signal transients identification by thresholding with digital Haar wavelet could be very efficient from the current consumption point of view, however simulations have revealed the presence of enormous shortcomings.
Thus, to overcome such problems and to enhance the Haar wavelet detection capability, we proposed and analyzed a hardware system able to preprocess the input signal. It is defined as pulse shaping.
As final conclusion, the feasibility to establish wavelets transform into wireless sensor nodes is then discarded. Motivated by the previous discussion, we observed the arisen need to use other family of algorithms.

6.1 Introduction

In literature different approaches for getting feeling about wavelets transforms are offered and discussed and the number of published works and books is quite large. Therefore, this chapter illustrates the theory by using explanations from different literature sources (see e.g. [**Mallat, S.G. (1999)**], [**Vitterli, M. (1992)**] and [**Sarkar, T.K. (1992)**]). Nevertheless, we attempt to unify notation and we adopted the notation used in [**Vitterli, M. (1992)**] and [**Walnut, D.F. (2004)**].

Wavelets transforms (WT from herein) provide the methodology to make representations of signals in components that show time and frequency signal features. Contrarily to the Fourier Transform, WT provide better insight and characterization of signals with time-varying frequency content.
Wavelets transforms enable breaking up a signal into coefficients defined as *approximations* and *details*, where, such coefficients represent the signal projection into subspaces generated by the basis functions $\Phi_{j,k}(t)$ and $\Psi_{j,k}(t)$ (the scaling and wavelet functions respectively and which will be formally defined in the next section).
It is well known that a signal $x(t)$ which meets certain mathematical properties (i.e. signals with finite energy or L^2) could be expressed as a linear combination of other functions. The process for synthesizing the signal $x(t)$ by using functions $\Phi_k(t)$ can be also interpreted as the $x(t)$ projection onto the space V where $\{\Phi_k(t)\}$ are the orthogonal functions basis of the space V. It could be expressed as:

$$x(t) = \sum_k c(k)\Phi_k(t) \tag{6.1}$$

$$c(k) = \langle x, \Phi_k \rangle = \int_{-\infty}^{\infty} x(t)\Phi_k(t)dt \tag{6.2}$$

Where:

$$\langle \Phi_p, \Phi_k \rangle = \delta_{p,k} = \begin{cases} 0 & p \neq k \\ 1 & p = k \end{cases} \tag{6.3}$$

With a suitable selection of $\{\Phi_k(t)\}$, the coefficients $c(k)$ not only show very well where transitions, abrupt changes and discontinuities are time-located, but also the signal long time behavior.
The Fourier transform only gives us the information regarding frequency components but spreading signal time features among the frequency axis. Conversely, wavelets transforms can describe the signal into frequencies bands but keeping time information.

6.1.1 Usage examples

In [**Ho, K.C. (2000)**], [**Prakasam, P. (2008)**] and [**Barsanti, R.J. (2007)**] the use of wavelets for allowing signal modulation identification in real time are proposed .The wavelet transform can extract the transient signal characteristic yielding signal modulation identification. Examples are provided the classification of PSK,FSK, $M-ary\ PSK$ and $M-ary\ FSK$ modulated signals are discussed.

In [**Haddad, S.A.P (2005)-3**] a procedure to implement wavelet transform by means of analog circuits is offered. The design meets the required low-power and low-voltage specifications and is based on log-domain integrators as key building blocks. The reported analog wavelet filter design operates from $1.5\,V$ supply voltage and achieves a total bias current of $4.3\,\mu A$.
Authors mention that digital WT implementation is not suitable for ultra-low power applications (e.g. biomedical applications) but instead, low-power circuitry for analog WT provide reliable signal features identifications and also reduced power consumption.
In [**Tsea, P.W. (2004)**] continuous wavelet transforms are proposed as effective tools for vibration-based machine fault diagnosis because by using wavelet analysis, it is possible to detect anomalous stationary and transitory signals thus avoiding breakdown of machines.
Moreover, in [**Sawigun, C. (2009)**] it is shown that wavelets transform is a powerful tool for real time signal processing that can be implemented in circuits achieving reduced power consumption and low-voltage power supply (around $1\,V$).
Authors proposed a nano-power wavelet filter ($1-st$ derivative of the Gaussian) for biomedical applications employing weak inversion $0.13\mu m$ CMOS technology with operation from a $1\,V$ single supply. The technique is defined as *Switched Gain Cell (SGC)* and authors highlight some advantages of such technique: *i)* it works with weak inversion MOSFETs that can achieve reduced current consumption; *ii)* the need of high precision transistor sizing is eliminated and *iii)* the impulse response scale modification can be implemented by electronic adjusting of the clock frequency.
Finally, in [**Tian, Q. (2006)**] authors present a novel model of noise removal for the human body vibration signals in wireless sensor networks.
Thus, wavelets transforms are useful in problems where time and frequency features are both important:

- recognition of local signal features or morphologies
- detection of transients or frequency variations
- SNR enhancement and denoising
- signal compression

6.1.2 Continuous wavelet transform: preliminary overview

The continuous wavelet transform (CWT) on L^2-signals is defined as:

$$CWT_x(a,\tau) \triangleq \langle x, \Psi_{a,\tau} \rangle = \int_{-\infty}^{\infty} x(t) \frac{1}{\sqrt{a}} \Psi\left(\frac{t-\tau}{a}\right) \tag{6.4}$$

Where a is the scale ($a \in \mathbb{R}^+$), τ is the translation and $\Psi(t)$ is the mother wavelet. In some literature is used the notation $X_\Psi(a,\tau) \triangleq CWT_x(a,\tau)$.
The wavelet transform is also considered as the following convolution product:

$$CWT_x(a,\tau) = \left[x(t) * \frac{1}{\sqrt{a}} \Psi\left(\frac{-t}{a}\right) \right]_{t=\tau} = \left[\int_{-\infty}^{\infty} x(s) \frac{1}{\sqrt{a}} \Psi\left(\frac{-t+s}{a}\right) ds \right]_{t=\tau} \tag{6.5}$$

There are a large number of mother wavelets, the most used are:

- Mexican Hat: $\Psi(t) = K(1-t^2)e^{-t^2/2}$
- Morlet: $\Psi(t) = K(1-t^2)e^{-t^2/2}e^{-j5.336\,t}$
- Derivative of Gaussian: $\Psi(t) = K\frac{d}{dt}e^{-t^2/2}$
- Haar: $\Psi(t) = \begin{cases} K & 0 < t < T_s/2 \\ -K & T_s/2 < t < T_s \\ 0 & otherwise \end{cases}$

Where K is the constant used for normalization and T_s, for the Haar wavelet is the time interval. For the discrete-time wavelet implementations such time interval is clearly identified as the time between signal samples (or sampling period). For discrete-time implementations the summation replaces the integral in (6.4) as follows:

$$DWT_x(a,n) = \frac{1}{\sqrt{a}}\sum_k x(nT_s)\Psi\left(\frac{kT_s - nT_s}{a}\right) \tag{6.6}$$

6.1.3 Transients identification and wavelet selection

In general a large amount of simulations is required in order to choose the scale which best highlights or shows CWT variances when a signal transient (whose detection is desired) takes place.
In literature two approaches to select suitable wavelets are mentioned, however the criteria to select the best mother wavelet and scale has not arisen yet in the researcher community and efforts towards to achieve such goal are being conducted. These two approaches are: *by maximum CWT magnitude variance* and *by using the Holder exponent.*
Both approaches are described below.

6.1.3.1 By maximum CWT magnitude variance

In **[Jin, Q.(1994)]** it is conjectured and used the following statement: if a the signal $x(t)$ can be described as $x(t, p(t))$ where $p(t)$ models a parameter that changes, then the wavelet that best detects the signal variation must meet the following two conditions:

- the $CWT_x(a,\tau)$ should be constant if no signal change occur, i.e. $CWT_x(a,\tau) = C(a)$ when $p(t) = constant$ and where $C(a)$ is function of a but independent of τ .
- when a signal change takes place at time t_0 (i.e. $p(t_0)$ transient) it should be clearly different from $C(a)$ so as to be clearly detected. Then, the following expression V has to be maximized:

$$V = |CWT_x(a,t_0) - C(a)| \tag{6.7}$$

Authors mention that there is no proof if a particular CWT maximizes (6.7). It should be verified by performing simulations. Particularly, in this work we use the Haar wavelet and as authors have mentioned, there is no proof that the Haar wavelet maximizes V.

6.1.3.2 By using the Holder exponent

The Holder regularity can measure the local regularity of a signal and it is defined in literature, e.g. **[Jaffard, S. (2004)]**, **[Mallat, S.G. (1992)]** and **[Zbigniew, R. S. (Report)]**. We explain the concept.
There is a polynomial $P_n(t-t_0)$ of order n that locally approximates the signal $x(t)$ at t_0 with a residual error characterized by the constant C and the exponent α.

$$x(t) = a_0 + a_1(t-t_0) + a_2(t-t_0)^2 + ... + a_n(t-t_0)^n + C\,|t-t_0|^\alpha = P_n(t-t_0) + C\,|t-t_0|^\alpha \tag{6.8}$$

In the wavelet framework the exponent α is defined as *Holder exponent.* It indicates the signal singularity and smoothness. A wavelet transform can not distinguish the signal $x(t)$ smoothness because the polynomial part of the signal $x(t)$ is ignored. Nevertheless, it is possible to highlights the remain error

between the polynomial and the signal $x(t)$. In order to achieve this, the wavelet transform needs n vanishing moments as next expressed:

$$\int_{-\infty}^{\infty} t^p \Psi(t) dt = 0 \qquad p = 0, 1, 2, 3.....n \tag{6.9}$$

The calculus of the Holder exponent α and to determine the wavelet that fulfills (6.9) is not an easy task and the researchers' efforts are toward the definition of a clear methodology to achieve this goal.

As we can see, the criteria to design the best wavelet are rather ad-hoc, and they depend on the signal.

6.1.4 Chapter goal

In this chapter we concentrate effort in the development of methodologies for signal transient detection capable to be implemented into the node with reduced power consumption.
Digital Haar wavelets (which will be formally defined in the next section) are characterized by the fast computation with reduced power consumption. Such wavelet is then a potential candidate to be implemented into wireless sensor nodes, nevertheless, examples as well as performance study for real WSN applications are scantily reported in literature.
Traditionally, wireless sensor nodes are programmed for implementing continuous signal sampling and transmission, with overloaded memory, radio transceiver excessive usage and potential traffic congestion (i.e. nodes work as wireless data-loggers).
The problem of implementing local signal processing is relatively new in the WSN arena and recently it has started to be studied [**Greenstein, B. (2006)**].
The wavelets implementation is then motivated by the node's need of performing local signal processing techniques in order to reduce power consumption. Such power consumption reduction is achieved because only the signal features of interest would be transmitted, with reduced radio transceiver activity.
The problem clearly emerges for applications which require a huge quantity of data. For instance, applications that involve 3-axis accelerometers per node are normal in WSN for structural health monitoring [**Lynch, J.P. (2004)**] and [**Kim, S. (2007)**]. (As regards accelerometers for WSN applications, interested readers can see [**Chang, M. (2006)**] and [**MMA7360 Datasheet**])
A crucial issue deserves attention. Traditionally, the wavelet is built by means of (6.9) or (6.7) and the methodology to achieve the suitable mother wavelet is rather ad-hoc. It often results in heavy power demanding and input signal features have to be known in advance.
In this chapter another strategy is undertaken, i.e. we selected the Haar wavelet its detection capability by thresholding and for general cases has been explored. In fact, one should not forget that the node is a very constrained hardware and only a very simple wavelet implementation has sense because it is closely related with the power demanding.

Motivated by the previous issues, we avoid overloaded processing of the wavelet coefficients (or transforms), as it is usually done, for instance by means of differentiation of transformed signals. Instead we attempt to test the simple detection by thresholding.
Therefore, the chapter focuses attention on the digital Haar wavelets implementation into nodes and its reliability to perform signal transition classification (or detection) *by thresholding*. Once the event is detected, a warning can be then transmitted or the sampling frequency can be increased in order to capture the event.

6.2 Background

6.2.1 The Scaling Function

For developing the Wavelet Transforms theory, it must be considered the family of functions that can be obtained through integer translations and dyadic scale dilatation (or contraction) of another function as follows:

$$\Phi_{j,k}(t) = 2^{\frac{j}{2}} \Phi(2^j t - k) \tag{6.10}$$

To get insight about notation, for instance the subspace V_0 is said to be generated by the set $\Phi_{0,k}(t)$, i.e the subspace V_0 is generated by dilatations or contractions of th scaling function $\Phi_0(t)$.
The function $\Phi(t)$ is named mother scaling function and $\{\Phi_{j,k}(t)\}$ $j, k \in \mathbb{Z}$ conform the scaling functions

or basis functions of the space V_j (for $j > 0$, we have a contraction of $\Phi_{j,k}(t)$ and for $j < 0$ dilatations).

For a given index j, ($j \in [-\infty, \infty]$) and with $-\infty < k < \infty$, if a function $f(t) \in V_j$, it yields:

$$f(t) = \sum_k a(k)\Phi_{j,k}(t) \tag{6.11}$$

For which:

$$a(k) = \langle f(t), \Phi_{j,k}(t) \rangle = \int_{-\infty}^{\infty} f(t)\Phi_{j,k}(t)dt \tag{6.12}$$

The scale of $\Phi_{j,k}(t)$ is $2^{-j/2}$ and the position is $k2^{-j}$. By adequately selecting the scaling function it is possible to describe the whole space $L^2(\mathbb{R})$ (i.e. the space of real functions $p = 2$ power integrable).

$$f(t) \in \mathbb{R} \quad \texttt{is} \quad L^p \quad \texttt{if} \quad \left[\int_{-\infty}^{\infty} |f(t)|^p dt\right]^{1/p} < \infty \tag{6.13}$$

For a given j, the space V_j is a subspace of $L^2(\mathbb{R})$. As the j value becomes as large as we want, i.e $j \to \infty$, the V_j dimension increases, becoming possible to cover all the space $L^2(\mathbb{R})$.
This mathematical formalism can also be interpreted as follows: since the V_j dimension is increasing, then it is possible to capture smaller function details. The idea is better explained below, when it is introduced the concept of difference between spaces.
In **Figure 6.1** , it is depicted an example, i.e. two Haar scaling functions for the subspaces V_0 and V_1.

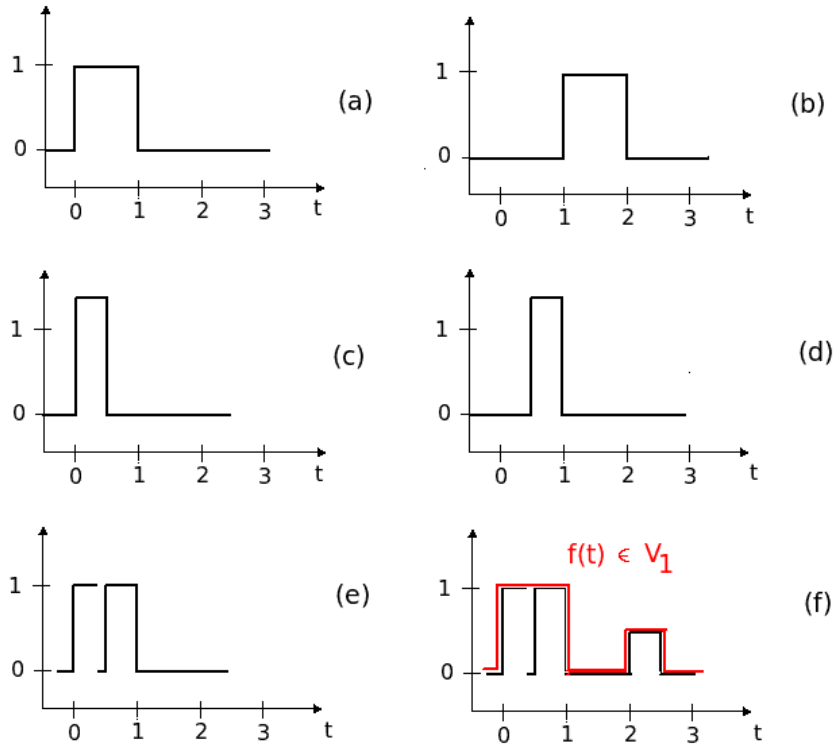


Figure 6.1: **Subfigure (a): Function $\Phi_{0,0}(t) = \Phi(t)$. Subfigure (b): Function $\Phi_{0,1}(t) = \Phi(t-1)$. Both basis functions are within the basis functions set of V_0. Similarly, subfigure (c) $\Phi_{1,0}(t) = \sqrt{2}\Phi(2t)$ and subfigure (d) $\Phi_{1,1}(t) = \sqrt{2}\Phi(2t-1)$. Both basis functions are within the basis functions set of V_1, please observe how the scale has been changed. Subfigure (e), $\Phi_{0,0}(t) = \frac{\Phi_{0,1}(t)}{\sqrt{2}} + \frac{\Phi_{1,1}(t)}{\sqrt{2}}$, therefore not only $\Phi_{0,0}(t) \in V_0$ but also $\Phi_{0,0}(t) \in V_1$. In Subfigure (f) we consider a simple function such that $f(t) \in V_1$. In V_1, by using the V_1 scale basis functions, a representation of the waveform could be: $f(t) = 0.5\Phi_{1,0}(t) + \Phi_{1,1}(t) + 0.3\Phi_{1,4}(t)$.**

6.2.2 Nested subspaces

The subspaces V_j are nested subspaces, because by construction we have that:

$$V_{-\infty} \subset \ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots \subset V_{\infty} \tag{6.14}$$

With: $V_{-\infty} = \{\emptyset\}$, and $V_{\infty} = L^2(\mathbb{R})$

The space difference (difference in the space sense) between spaces V_j and V_{j+1}, is denoted W_j. It is possible to show (see for instance [**Vitterli, M. (1992)**]) that the orthogonal complement of V_j in V_{j+1} is W_j. Therefore, if we define the wavelet function as the basis functions of the space W_j, then, it is possible to express that:

$$\langle \Phi_{j,p}(t), \Psi_{j,k}(t) \rangle = 0 \qquad \forall \quad j,p,k \in \mathbb{Z} \tag{6.15}$$

Similarly to the scale functions, the wavelet or W_j function basis can be obtained through integer translations and dyadic scale reduction from another mother wavelet function as follow:

$$\Psi_{j,k}(t) = 2^{\frac{j}{2}} \Psi(2^j t - k) \tag{6.16}$$

This leads to the possibility of infinity different decompositions for $L^2(\mathbb{R})$, for instance :

$$L^2(\mathbb{R}) = V_0 + W_0 + W_1 + W_2 + \cdots \tag{6.17}$$

$$L^2(\mathbb{R}) = V_1 + W_1 + W_2 + W_3 \cdots \tag{6.18}$$

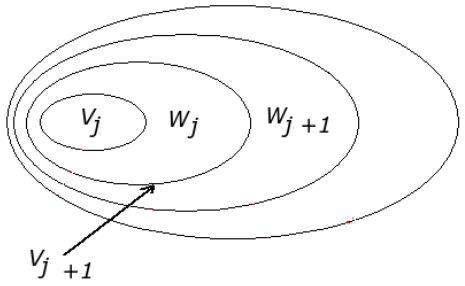


Figure 6.2: **Nested Subspaces $V_{j+1} = V_j + W_j$, $V_j \rightarrow V_{j+1}$: time-contraction (getting more signal details): $V_j \leftarrow V_{j+1}$: time-dilatation (worst signal description).**

Moreover, the space $L^2(\mathbb{R})$ could be expressed only as a function of the wavelets basis.

$$L^2(\mathbb{R}) = V_{-\infty} + \cdots + W_{-2} + W_{-1} + W_0 + W_1 + W_2 + \cdots \tag{6.19}$$

If $f(t) \in V_{j+1}$ but $f(t) \notin V_j$, thus $f(t) \in W_j$ and $f(t)$ can be expressed as :

$$f(t) = \sum_k d_j(k) \Psi_{j,k}(t) \tag{6.20}$$

Where: $-\infty < k < \infty$ and j correspond with the space index. It is clear that if $f(t) \in V_{j+1}$ then $f(t) \in V_j$ since $V_j \subset V_{j+1}$.

6.2.3 Scaling functions: main properties

Since $V_j \subset V_{j+1}$, there is an expression that relates the scale function and itself, by using the coefficients $h_{\Phi,k}(n)$ for which it is possible to write:

$$\Phi_{j,k}(t) = \sum_n h_{\Phi,k}(n) \Phi_{j+1,n}(t) = \sum_n h_{\Phi,k}(n) 2^{(j+1)/2} \Phi(2^{j+1} t - n) \tag{6.21}$$

Such equation is well-known in literature as *refinement or dilation equation* (the number of terms n depends on the scaling function type, but generally speaking it is a finite terms sum). Particularly,it is easy to see that equation (6.22) becomes:

$$\Phi(t) = \sum_n h_{\Phi}(n) \sqrt{2} \Phi(2t - n) \tag{6.22}$$

(We remark that in literature, readers can find the equation (6.22) expressed with coefficients defined as $c_n = \sqrt{2} h_{\Phi}(n)$ or $h_0(n)$ instead of $h_{\Phi}(n)$, but the difference lays only in notation).

Another important feature is that by definition, basis functions as well as wavelets functions are build with the desired feature of to be time compactly supported, so, there is a finite number L of coefficients $h_\Phi(n)$ non-zero. Thus, expression (6.22) could be written also as:

$$\Phi(t) = \sum_{n=0}^{L-1} h_\Phi(n)\sqrt{2}\Phi(2t-n) \tag{6.23}$$

To have better insight, in **Figure 6.3** it is reproduced the well-known example often offered in literature of the Haar scaling function, and for which:

$$\Phi(t) = \frac{1}{\sqrt{2}}\sqrt{2}\Phi(2t) + \frac{1}{\sqrt{2}}\sqrt{2}\Phi(2t-1) \quad \Rightarrow h_\Phi(0) = \frac{1}{\sqrt{2}},\ \ h_\Phi(1) = \frac{1}{\sqrt{2}} \tag{6.24}$$

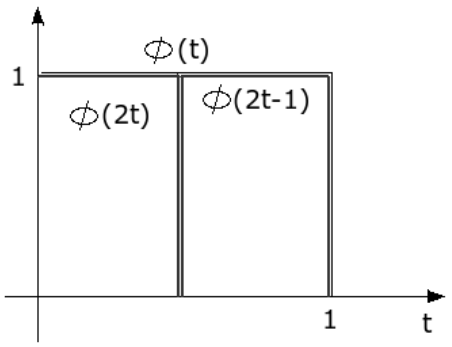


Figure 6.3: **Example of the scale equation for the Haar scale function case, see expression (6.24) and .**

Similarly, as $W_j \subset W_{j+1}$ and particularly $W_0 \subset V_1$ for the wavelet function it is also hold that:

$$\Psi(t) = \sum_n h_\Psi(n)\sqrt{2}\Phi(2t-n) \tag{6.25}$$

Finally, it is possible to show that:

$$h_\Psi(n) = (-1)^n h_\Phi(1-n) \tag{6.26}$$

(Interested readers can get the proof from in the above mentioned literature)
It is important to underline that the scaling functions are orthogonal among them with respect to the integer translation. Moreover, there is an important property named in literature as *2-factor compression propriety* or scale equation which is expressed as follows:

$$f(t) \in V_j \longleftrightarrow f(2t) \in V_{j+1} \tag{6.27}$$

In addition to this, nested subspaces fulfil other proprieties that can be enunciated as:

$$f(t) \in V_j \longrightarrow f\left(\frac{t}{2}\right) \in V_j \tag{6.28}$$

Finally, if we consider the space V_0, then:

$$f(t) \in V_0 \longrightarrow f(t-k) \in V_0 \quad \forall\, k \tag{6.29}$$

$$\langle \Phi(t-j), \Phi(t-k) \rangle = \delta_{j,k} \quad \forall\, k \tag{6.30}$$

6.2.4 Coefficients: main properties

The values for the coefficients $h_\Phi(n)$ and $h_\Psi(n)$ (see equation (6.26) and (6.25)) are determined by constraints of normalization and orthogonality among the selected basis functions.
First of all, the scaling functions requires to be normalized , i.e. $\int_S \Phi(t)dt = 1$, where S is the compact support. Then, we have that:

$$1 = \int_S \Phi(t)dt = \sum_0^{L-1} \sqrt{2} \int_S h_\Phi(n)\Phi(2t-n)dt = \sum_0^{L-1} \frac{1}{\sqrt{2}} h_\Phi(n) \int_{S_n} \Phi(u)du \tag{6.31}$$

Thus:

$$\sqrt{2} = \sum_{0}^{L-1} h_\Phi(n) \tag{6.32}$$

(We remark that in literature it is often used the notation $c_n = \sqrt{2}h_\Phi(n)$ for the coefficients). Another required condition is the orthogonality among basis functions and wavelets.

$$\langle \Phi(t-q), \Phi(t-k) \rangle = \sum_{l=0}^{L-1} h_\Phi(l)\sqrt{2} \left(\sum_{m=0}^{L-1} h_\Phi(m)\sqrt{2} \langle \Phi(2t-2q-l), \Phi(2t-2k-m) \rangle \right) \tag{6.33}$$

$$\delta_{q,k} = \sum_{l=0}^{L-1} h_\Phi(l)\sqrt{2} \left(\sum_{m=0}^{L-1} h_\Phi(m)\sqrt{2}\frac{1}{2} \langle \Phi(u), \Phi(u+2q+2l-2k-m) \rangle \right) \tag{6.34}$$

$$\delta_{q,k} = \sum_{l=0}^{L-1} h_\Phi(l) \left(\sum_{m=0}^{L-1} h_\Phi(m)\delta_{l+2q,2k+m} \right) \tag{6.35}$$

Then:

$$\delta_{q,k} = \sum_{l=0}^{L-1} h_\Phi(l) \sum_{m=0}^{L-1} h_\Phi(l+2q-2k) \tag{6.36}$$

Particularly:

$$\delta_{0,k} = \sum_{l=0}^{L-1} h_\Phi(l) \sum_{m=0}^{L-1} h_\Phi(l-2k) \tag{6.37}$$

The discrete filter is orthogonal to its own even translates. Similarly it is possible to show that the coefficient must fulfill the following:

$$\sum_{n=0}^{L-1} h_\Psi(n) = 0 \tag{6.38}$$

6.2.5 Multiresolution processing and wavelets

The decomposition of a function $f(t) \in L^2(\mathbb{R})$ at scale j_0 with respect to the wavelet $\Psi(t)$ and the scale function $\Phi(t)$ is:

$$f(t) = \sum_k a_{j_0}(k)\Phi_{j_0,k}(t) + \sum_{j=j_0}^{\infty} \sum_k d_j(k)\Psi_{j,k}(t) \tag{6.39}$$

Where: $\sum_k a_{j_0}(k)\Phi_{j_0,k}(t) \in V_{j_0}$ and $\sum_k d_j(k)\Psi_{j,k}(t) \in \Psi_j$

Is then says that the function $f(t) \in L^2(\mathbb{R})$ is expressed by its projection into a certain sum of spaces:

$$f(t) \in [V_{j_0}, W_{j_0}, W_{j_0+1}, W_{j_0+2} \cdots] \tag{6.40}$$

Please, notice how the function $f(t)$ is expressed at scale j_0 by using the scale functions.
The remaining details are added through the wavelet functions. The functions $\Phi(t)$ and $\Psi(t)$ are often named expansion functions.
Since wavelets functions and scale functions are orthogonal among them and for each respective subspace, thus, the coefficients can be calculated as:

$$a_{j_0} = \langle f(t), \Phi_{j_0,k}(t) \rangle \tag{6.41}$$

$$d_j = \langle f(t), \Psi_{j,k}(t) \rangle \tag{6.42}$$

Where: $a_{j_0}(k)$ is the scale approximation coefficient and $d_j(k)$ is the detail coefficient. There are other expression to perform the decomposition into subspaces.
If $f(t) \in V_j$, as $V_j = V_{j-1} + W_{j-1}$ it yields:

$$f(t) = \sum_{i=-\infty}^{j-1} \sum_k d_i(k)\Psi_{i,k}(t) = \sum_{i=-\infty}^{j-2} \sum_k d_i(k)\Psi_{i,k}(t) + \sum_k d_{j-1}(k)\Psi_{j-1,k}(t) \tag{6.43}$$

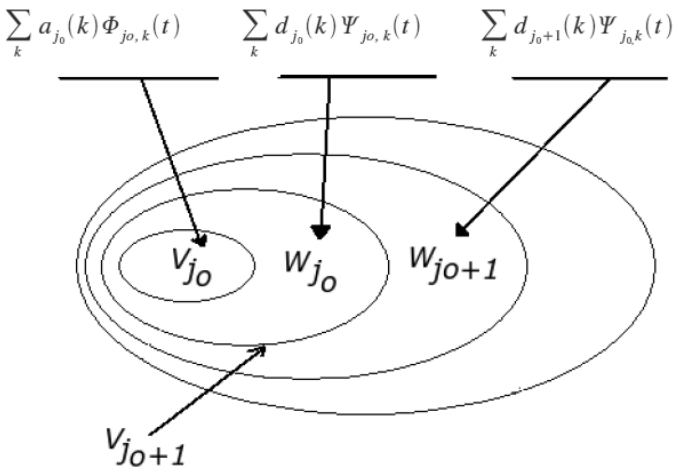


Figure 6.4: **Graphical Representation for the expression (6.39).**

In addition to this, we also have that:

$$f(t) = \sum_k a_j(k)\Phi_{j,k}(t) = \sum_k a_{j-1}(k)\Phi_{j-1,k}(t) + \sum_k d_{j-1}(k)\Psi_{j-1,k}(t) \tag{6.44}$$

At this point we arrive to the Multiresolution Analysis meaning. The signal decomposition is carried in relation to a nested sequence of subspaces. To illustrate, we take $f(t) \in V_j$ and as we showed previously:

$$V_j = V_{j-1} + W_{j-1} \tag{6.45}$$

$$V_j = V_{j-2} + W_{j-2} + W_{j-1} \tag{6.46}$$

$$V_j = V_{j-3} + W_{j-3} + W_{j-2} + W_{j-1} \tag{6.47}$$

$$V_j = V_{j-4} + W_{j-4} + W_{j-3} + W_{j-2} + W_{j-1} \tag{6.48}$$

Thus:

$$f(t) = a_{j-1}(t) + d_{j-1}(t) \tag{6.49}$$

$$f(t) = a_{j-2}(t) + d_{j-2}(t) + d_{j-1}(t) \tag{6.50}$$

$$f(t) = a_{j-3}(t) + d_{j-3}(t) + d_{j-2}(t) + d_{j-1}(t) \tag{6.51}$$

$$f(t) = a_{j-4}(t) + d_{j-4}(t) + d_{j-3}(t) + d_{j-2}(t) + d_{j-1}(t) \tag{6.52}$$

Where the approximation at level i is $a_i(t) \in V_i$, and $d_i(t) \in V_{i+1} - V_i = W_i$. More specifically in order to have better insight we point out as example and from the expression (6.49) for the case $V_j = V_{j-1} + W_{j-1}$, we have that:

$$a_{j-1}(t) = \sum_k a_{j-1}(k)\Phi_{j-1,k}(t) \tag{6.53}$$

$$d_{j-1}(t) = \sum_k d_{j-1}(k)\Psi_{j-1,k}(t) \tag{6.54}$$

6.2.6 Examples of decomposition expressed by equation (6.52)

In this section two examples are depicted in **Figure 6.5** and **Figure 6.6**, where it is shown the signal decomposition at approximation level and details levels.
Please, observe that the first level holds the main signal features and below level 4, details signals seems "noise".
Could be possible to transmit only samples of the *approximated signals* with reduced frequency sampling.
For Haar basis functions:

$$\Phi(t) = \begin{cases} 1 & 0 < t < T_s \\ 0 & otherwise \end{cases} \tag{6.55}$$

$$\Psi(t) = \begin{cases} 1 & 0 < t < \frac{T_s}{2} \\ -1 & \frac{T_s}{2} < t < T_s \\ 0 & otherwise \end{cases} \tag{6.56}$$

T_s is the sampling interval, or $f_s = \frac{1}{T_s}$.

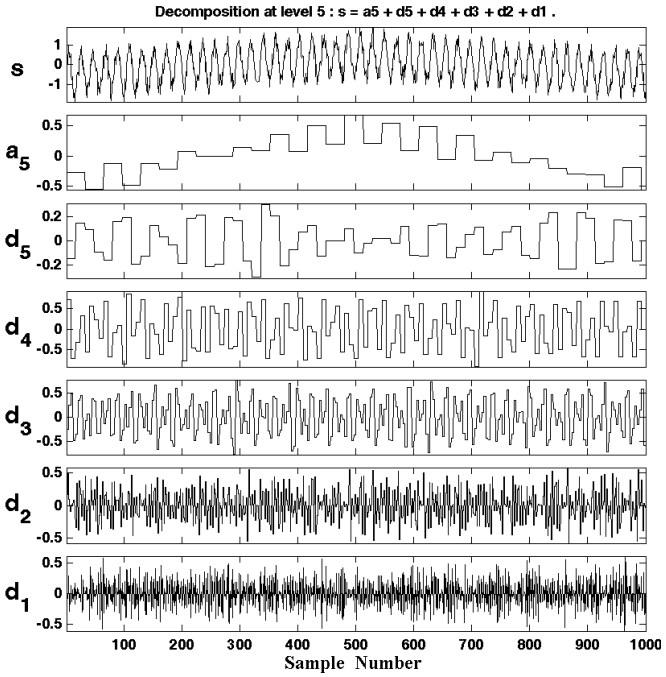


Figure 6.5: **Example of a decomposition carried out using the tool wavemenu in MATLAB®. The original signal is $s(t)$ in the top picture. The decomposition is performed at level 5 by using Haar Wavelets. This signal example is provided by the MATLAB®wavelet demo tool.**

6.3 Filters banks: from detail to coarse signal description

In this section, it is shown how starting from the coefficients at scale index j, it is possible to produce two new coefficients at scale index $j-1$. More specifically, if we assume without loss of generality that $f(t) \in V_j$, then, a possible decomposition previously mentioned is:

$$f(t) = \sum_k a_j(k)\Phi_{j,k}(t) = \sum_k a_{j-1}(k)\Phi_{j-1,k}(t) + \sum_k d_{j-1}(k)\Psi_{j-1,k}(t) \tag{6.57}$$

Therefore, it is possible to produce the set $[a_{j-1}(k), d_{j-1}(k)]$ from the coefficient $a_j(k)$ at scale j as it is described below.

The coefficients can be computed with the inner product because wavelets and scale functions are orthogonal with respect to different index level. Then:

$$a_{j-1}(k) = \langle f(t), \Phi_{j-1,k}(t) \rangle = \sum_n \langle a_j(n)\Phi_{j,n}(t), \Phi_{j-1,k}(t) \rangle \tag{6.58}$$

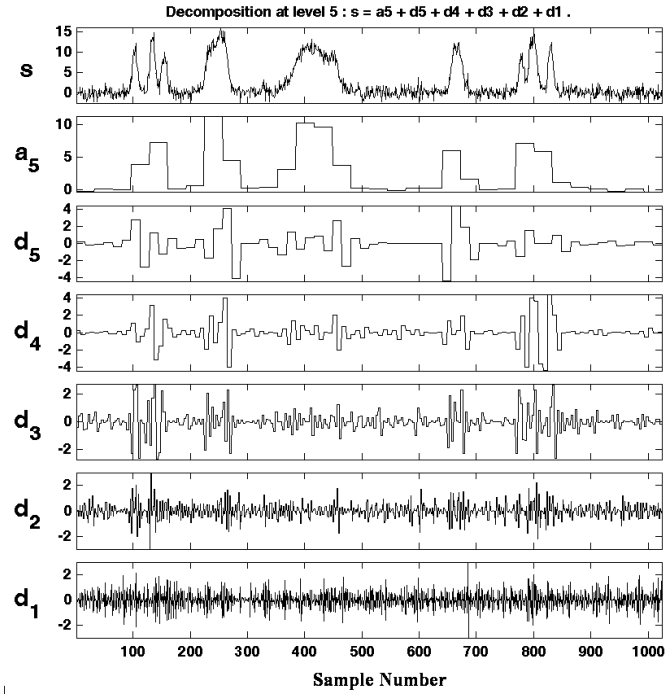


Figure 6.6: **Example of a decomposition carried out using the tool wavemenu in MATLAB®. The original signal is $s(t)$ in the top picture. The decomposition is performed at level 5 by using Haar Wavelets. This signal example is provided by the MATLAB®wavelet demo tool.**

After calculating, it yields to:

$$a_{j-1}(k) = \sum_n h_o(n-2k)a_j(n) \tag{6.59}$$

(Interested readers should take the referenced literature to read the proof). Where, in order to unify notation with the literature, the coefficients $h_\Phi(n)$ have been represented as $h_o(n)$ in expression (6.59). Readers can find in literature the coefficients named also as $h_G(n)$.
Such coefficients are the core of the scale refinement equation:

$$\Phi_j(t) = \sum_n h_o(n)\Phi_{j+1,n}(t) = \sum_n h_o(n)2^{(j+1)/2}\Phi\left(2^{j+1}t-n\right) \tag{6.60}$$

Similarly:

$$d_{j-1}(k) = \langle f(t), \Psi_{j-1,k}(t)\rangle = \sum_n \langle a_j(n)\Phi_{j,n}(t), \Psi_{j-1,k}(t)\rangle \tag{6.61}$$

After calculating, it yields to:

$$d_{j-1}(k) = \sum_n h_1(n-2k)a_j(n) \tag{6.62}$$

Where, in order to unify notation with the literature, we use the notation $h_1(n)$ for the coefficients which actually are $h_\Psi(n)$ (see for instance 6.25) and that hold the scale refinement equation for the wavelets basis functions as follows:

$$\Psi_j(t) = \sum_n h_1(n)\Phi_{j+1,n}(t) = \sum_n h_1(n)2^{(j+1)/2}\Phi\left(2^{j+1}t-n\right) \tag{6.63}$$

Then, the procedure to obtain the coefficients at scale $j-1$ starts at scale j with the coefficients $a_j(k)$. The mechanism for calculating the approximation and detail coefficients ($a_{j-1}(k)$ and $d_{j-1}(k)$) is a convolution with a reversed and decimated filter.

In fact, it is clear that:

$$a_{j-1}(k) = \left[\sum_n h_o\left(-\left(m-n\right)\right)a_j(n)\right]_{m=2k,\ k\geq 0} = \left[h_o(-m) * a_j(m)\right]_{m=2k,\ k\geq 0} \tag{6.64}$$

Figure 6.7: **The procedure to obtain the coefficients at scale $j-1$ starts at scale j with the coefficients $a_j(k)$. By using the filter banks it is possible to calculate $a_{j-1}(k)$ and $d_{j-1}(k)$.**

That can be expressed also as:

$$a_{j-1}(k) = [h_o(-m) * a_j(m)]_{m=2k,\ k\geq 0} = \sum_n h_o(n-2k)\, a_j(n) \tag{6.65}$$

Or, it is also expressed in literature as follows:

$$a_{j-1}(k) = \left[h_o^{\downarrow}(m) * a_j(m)\right]_{m=2k,\ k\geq 0} = \sum_n h_o^{\downarrow}(n-2k)\, a_j(n) \tag{6.66}$$

Or, alternatively that is:

$$a_{j-1}(k) = \left[h_o^{\downarrow}(k) * a_j(k)\right] \downarrow 2 \tag{6.67}$$

One point that deserve attention is the time reversed filter definition which is:

$$h_o(-n) = h_o^{\downarrow}(n) = [h_o(L-1), h_o(L-2), \cdots, h_o(1), h_o(0)] \tag{6.68}$$

Similarly:

$$d_{j-1}(k) = \left[\sum_n h_1(-(m-n))\, a_j(n)\right]_{m=2k,\ k\geq 0} = [h_1(-m) * a_j(m)]_{m=2k,\ k\geq 0} \tag{6.69}$$

That can be expressed also as:

$$d_{j-1}(k) = [h_1(-m) * a_j(m)]_{m=2k,\ k\geq 0} = \sum_n h_1(n-2k)\, a_j(n) \tag{6.70}$$

$$d_{j-1}(k) = \left[h_1^{\downarrow}(k) * a_j(k)\right] \downarrow 2 \tag{6.71}$$

Please, remember that the coefficients are related by the following expression [**Vitterli, M. (1992)**]:

$$h_1(n) = (-1)^n h_o(L-1-N) \tag{6.72}$$

Due to convolution properties and the definition of the time-reversed filter it is possible to show that the following expressions are equivalent:

$$a_{j-1}(k) = \sum_n h_o(-n) a_j(n-2k) = \sum_{n=0}^{L-1} h_o(L-1-n) a_j(n+2k) \tag{6.73}$$

$$d_{j-1}(k) = \sum_n h_1(-n) a_j(n-2k) = \sum_{n=0}^{L-1} h_1(L-1-n) a_j(n+2k) \tag{6.74}$$

(The digital convolution is defined as: $z(k) = x(k) * y(k) = \sum_n x(n)y(k-n) = \sum_n x(k-n)y(n)$)

Please, remember that L must be even. In the next **Figure 6.8** is depicted the final representation for the filters bank implementation. The process is implemented with time reversed filters with downsampling by 2.

The sequence $[h_o(-n)]$ is a digital half band low-pass filter and similarly, $[h_1(-n)]$ is a half high-pass digital filter. Both are FIR filters with size L.
According to [**Vitterli, M. (1992)**] the following properties are hold:

$$\langle h_o(n-2l), h_o(n-2k)\rangle = \delta_{l,k} \qquad l,k \in \mathbb{Z} \tag{6.75}$$

$$\langle h_1(n-2l), h_1(n-2k)\rangle = \delta_{l,k} \qquad l,k \in \mathbb{Z} \tag{6.76}$$

$$\langle h_1(n-2l), h_o(n-2k)\rangle = \delta_{l,k} \qquad l,k \in \mathbb{Z} \tag{6.77}$$

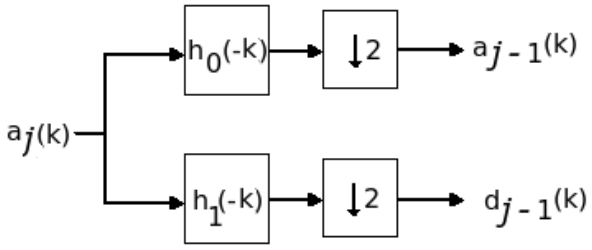


Figure 6.8: **One filter stage to obtain the coefficients by means of filter banks.**

The sequence $[a_j(k)]$ is considered to be the discrete time signal that has to be processed. By filtering process the frequencies above $fx/2$ are removed (f_x is the signal bandwidth). Due to this, half of the samples can be eliminated without violating the Nyquist-Shannon sampling theorem (the highest normalized signal frequency is $\pi\ rad/s$ which is the half of the sampling frequency). It is performed by discharging every sample between odd indexes samples.
After applying the filter bank the resulting size is then a half of the original $a_j(k)$ vector length.

6.3.1 Examples of coefficients $h(n)$

In order to have better insight on the above discussed concepts, this section offers two examples often presented in literature:

- **Haar Wavelets:**

$$h_o(n) = [h_o(0), h_o(1)] = \frac{1}{\sqrt{2}}[1,1] \Rightarrow h_o^{\prime}(n) = \frac{1}{\sqrt{2}}[1,1] \tag{6.78}$$

$$h_1(n) = [h_1(0), h_1(1)] = \frac{1}{\sqrt{2}}[1,-1] \Rightarrow h_1^{\prime}(n) = \frac{1}{\sqrt{2}}[-1,1] \tag{6.79}$$

- **Daubechies-4 Wavelets:**

$$h_o(n) = [h_o(0), h_o(1), h_o(2), h_o(3)] = \frac{1}{4\sqrt{2}}\left[1+\sqrt{3}, 3+\sqrt{3}, 3-\sqrt{3}, 1-\sqrt{3}\right] \tag{6.80}$$

$$h_1(n) = [h_1(0), h_1(1), h_1(2), h_1(3)] = [h_o(3), -h_o(2), h_o(1), -h_o(0)] \tag{6.81}$$

Then:

$$h_o^{\prime}(n) = [h_o(3), h_o(2), h_o(1), h_o(0)] \tag{6.82}$$

$$h_1^{\prime}(n) = [h_1(3), h_1(2), h_1(1), h_1(0)] = [-h_o(0), h_o(1), -h_o(2), h_o(3)] \tag{6.83}$$

6.4 Cascade filtering

If we assume without loss of generality that $x(t) \in V_j$, then a possible signal decomposition which has been previously mentioned is:

$$x(t) = \sum_k a_j(k)\Phi_{j,k}(t) = \sum_k a_{j-1}(k)\Phi_{j-1,k}(t) + \sum_k d_{j-1}(k)\Psi_{j-1,k}(t) \tag{6.84}$$

Written more compactly;

$$x(t) = a_{j-1}(t) + d_{j-1}(t) \tag{6.85}$$

The function $a_{j-1}(t)$ is the approximation of the function $x(t)$ and $d_{j-1}(t)$ contains the details omitted by $a_{j-1}(t)$.
A we have been seen previously, the coefficients $a_{j-1}(t)$ and $d_{j-1}(t)$ are calculated from $a_j(t)$, however we have also that:

$$a_{j-1}(t) = \sum_k a_{j-1}(k)\Phi_{j-1,k}(t) = \sum_k a_{j-2}(k)\Phi_{j-2,k}(t) + \sum_k d_{j-2}(k)\Psi_{j-2,k}(t) \tag{6.86}$$

Which can be expressed also as:

$$x(t) = a_{j-2}(t) + d_{j-2}(t) + d_{j-1}(t) \tag{6.87}$$

The nesting process can continue as **Figure 6.9** describes. Multiple filter banks built with a single stage (or basic filter bank) are used so that to obtain the signal transformation.

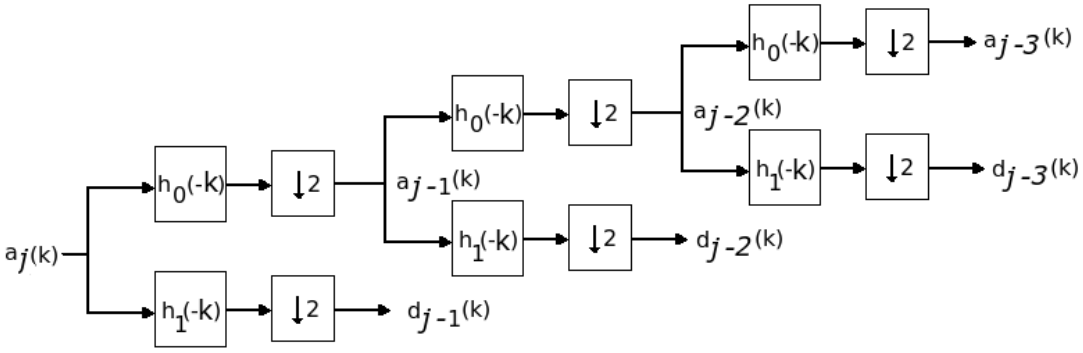


Figure 6.9: **Using filter bank in cascade to obtain the coefficients for the wavelet transform.**

6.5 Haar: suitable wavelet for implementation into wireless nodes

The digital wavelet implementation selection is conditioned not only by the reduced wireless sensor node computation capabilities but also for the node energy consumption constraints. Therefore, among the large number of possible wavelets and after performing a deep study of literature we selected Haar wavelets transforms due to the following reasons:

1. ***Easy Implementation:*** The digital implementation to decompose the signal into two sub-signals (**Figure 6.9**) can be easily performed within wireless sensor nodes (mainly, it consists in running averages and differences. In contrast with most of the digital wavelet transforms, the reduced computational effort and low energy budget seems to differ of many orders of magnitude in favor of the Haar DWT.
2. ***Algorithm Startup*** . The first coefficients ($a_j(n)$ in **Figure 6.9**) as we will see below, are directly related with the sampled signal.

Easy Implementation: The coefficients for each sub-bands filter is reported in literature and clearly illustrates the algorithm simplicity. With reference to the **Figure 6.9** we have that:

$$a_{j-1}(k) = \frac{1}{\sqrt{2}}\left[a_j(2n-1) + a_j(2n)\right] \tag{6.88}$$

$$d_{j-1}(k) = \frac{1}{\sqrt{2}}\left[a_j(2n-1) - a_j(2n)\right] \tag{6.89}$$

Where $n = 1, 2, 3.....\frac{N}{2}$ and $N = 2^P$ is the number of samples with P integer.
Please, observe the change in the index (n to k).
Algorithm Startup: The discussion previously addressed shows how to estimate the coefficients $a_{j-1}(n)$ and $d_{j-1}(n)$ from $a_j(n)$, however it is necessary to calculate the startup coefficient $a_J(n)$ (or often defined in literature as $a_0(n)$ because it is used at the beginning of the filtering chain). This section faces such issue.
In order to apply the digital wavelet transform it is required a signal within the space V_J ($x(t) \in V_J$).
If the signal $x(t) \in L^2(R)$,then, it is necessary to select a scale index J large enough so that the signal $x(t)$ can be reasonably approximated by $x_J(t)$ where $x_J(t)$ is the $x(t)$ projection into the space V_J.
That is:

$$x(t) \approx x_J(t) = \sum_n a_J(n)\Phi_{J,n}(t) \tag{6.90}$$

Therefore, the general expression is:

$$a_J(n) =< x(t), \Phi_{J,n}(t) >= \int_{-\infty}^{\infty} x(t)2^{J/2}\Phi(2^J t - n))dt \tag{6.91}$$

The signal $x(t)$ is in fact a sampled signal (from the AD converter) and it can be expressed as: $x(t) = \sum_k x(kT_s)p(t - kT_s)$, where $f_s = 1/T_s$ is the sampling frequency and $p(t)$ is a pulse with width T_s (flat top sampling or PAM obtained by employing sample-and-hold techniques).
Moreover, if we define the first level as $J = 0$, and as the scaling function $\Phi(t)$ is the square wave (Haar) we obtain that:

$$a_0(n) = \int_{-\infty}^{\infty} \sum_k x(kT_s)p(t - kT_s)\Phi(t - nT_s)dt = \int_{nT_s}^{(n+1)T_s} \sum_k x(kT_s)p(t - kT_s)dt = x(nT_s) \tag{6.92}$$

6.6 Transient signal detection by thresholding

In this work we concentrate effort in the development of methodologies for signal transient detection capable to be implemented into the node. For this, we avoid overloaded processing of the wavelet coefficients (or transforms) as we mentioned at the chapter beginning. Instead we attempt to test a simple detection by thresholding.
The following case studies are the found key numerical examples (among other) which show the powerful tool that wavelets can be, but at the same time it is revealed enormous shortcomings.

6.6.1 Numerical case study N° 1

We analyze the following example; suppose that we face the signal:

$$x(t) = \begin{cases} C + A_1 sin\left(2\pi f_1 t\right) + n(t) & t \in [0, t_1], t \in [t_2, t_3], t \geq t_4 \\ C + A_2 sin\left(2\pi f_2 t\right) + n(t) & t \in [t_2, t_3] \\ C + A_3 sin\left(2\pi f_3 t\right) + n(t) & t \in [t_3, t_4] \end{cases} \tag{6.93}$$

Where the parameters are:

- $A_1 = 1\,V$, $f_1 = 25\,Hz$, $t_1 = 6.5\,s$
- $A_2 = 1\,V$, $f_2 = 26\,Hz$, $t_2 = 11.2\,s$
- $A_3 = 1\,V$, $f_3 = 27\,Hz$, $t_3 = 14\,s$, $t_4 = 26\,s$
- sampling frequency $f_s = 100\,Hz$
- number of samples $N = 2848$
- $C = 1\,V$ and $n(t)$ is considered to be white noise with $\sigma_n^2 = 4 \times 10^{-4}\,V^2$

Clearly, it is a sine wave that varies its frequency $1\,Hz$ with respect to itself within certain time intervals. The signal decomposition is performed in 3 decedent steps or levels, starting from level 0 or the sampled version of $x(t)$ (at the sampling frequency f_s). As we can see, the wavelet decomposition permits us to determine clearly when the frequency transitions (or transient signals) take place.
For instance, at level -2 the coarse signal approximation $a_{j-2}(t)$ and the detail signal $d_{j-2}(t)$ show the transition edges.
Please observe that in case of level -2 ($d_{j-2}(t)$) it is possible to define a threshold at value zero that identify portions of the signal. The threshold acts as a decision surface.

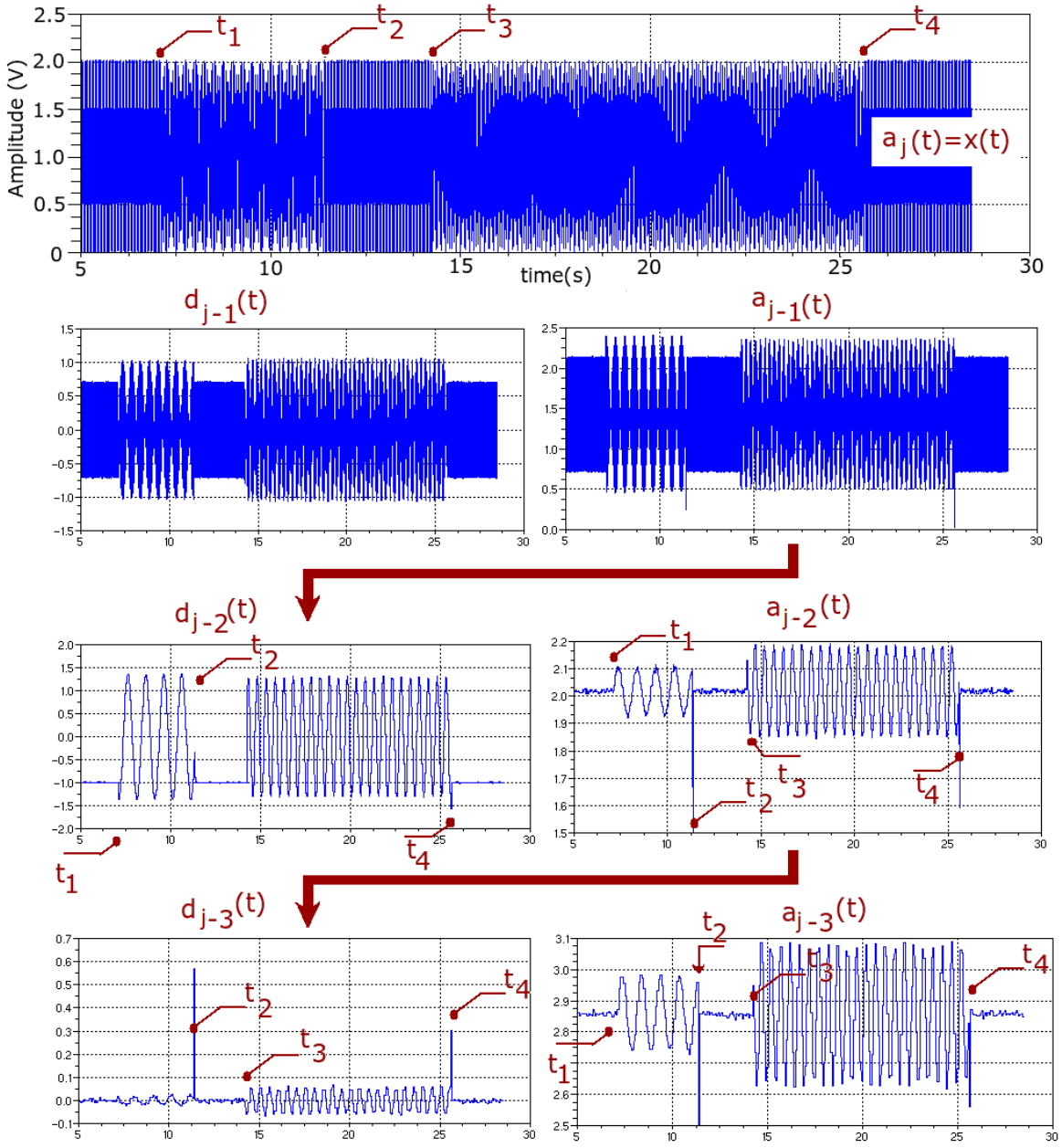


Figure 6.10: **Haar wavelet transform example for the signal (6.93)**

6.6.2 Numerical case study N° 2

We analyze the following example that intends to simulate vibration signals. Suppose that the following signal $x(t)$ is sampled:

$$x(t) = x_{dc} + \sum_{n=1}^{N} A_n sin(2\pi f_n t) \qquad \forall t \tag{6.94}$$

During $t \in [t_1, t_2]$ it arises another harmonic signal $y_o(t)$. We are interested on the detection of such signal, i.e. the goal is to detect when the signal $y_o(t)$ starts.

$$x(t) = x_{dc} + \sum_{n=1}^{N} A_n sin(2\pi f_n t) + y_o(t) \qquad t \in [t_1, t_2] \tag{6.95}$$

Where: $y_o(t) = A_o sin(2\pi f_o) \qquad t \in [t_1, t_2]$
In the **Figure 6.11**, the sampled signal es named $x(k)$ (or also $a_o(j)$) where the parameters are as follows:

1. $N = 150$
2. A_n is randomly selected with a PDF uniform between $[0, 50]\ V$, $(A_n \in [0, 50]\ V)$
3. $f_n \in [0.5, 50]\ Hz$ in steps of $\Delta f = 0.32Hz$, i.e. $f_{n+1} = f_n + \Delta f$
4. for the signal of interest $A_o = 1V$, $f_o = 23Hz$, $t_1 = 4.3s$ and $t_2 = 7.8s$
5. sampling frequency $f_s = 368Hz$
6. samples number 3200 ($k = 1, 2, \cdots, 3200$ for $x(k)$)

Figure 6.11 shows the signal $x(k)$ after ADC sampling, its FFT and different coefficients of the wavelets decomposition (up to level 4).
Please, observe that the problem can not be resolved by means of the FFT. Moreover, we see that at level 4 the presence of the signal of interest is revealed by using a well defined threshold, that, for this case study is $Level_{th} = 0$.
The threshold acts as surface decision which highlights the moment when the signal $y_o(t)$ appears.

6.7 Unreliable detection by thresholding: analysis

Previous examples have shown the wavelets' utility. However, the detection by thresholding has drawbacks related to the reliability that we reveal through the following study.
For this study, the wavelet filtering is implemented in the PC (with Scilab) but with real sampled signals. The reference signal is depicted in **Figure 6.12** and also it is shown in the successive figures (always placed at left side) for confronting case studies.
The reference signal consists in the simulation of an input sine waveform with anomalies. The waveform is built with 7 sine-cycles bursts with pauses between them by means of the Agilent 33250A 80MHz Function Arbitrary Waveform. generator by using the mode *burst built-in waveforms*, and by setting the following parameters:

1. *Amplitude:* 2 V_{pp} ,it is assumed to be fixed for each measurement.
2. *Burst Period (b_p):* it is shown in **Figure 6.12** and it relates the wave period.
3. *Sine Frequency (f_o):* it is the frequency of the sine wave.
4. *Start Phase (φ):* it is the phase with the the sine wave starts (it controls the position of the simulated anomaly).

The signal is sampled by the MicaZ sensor node which features a microcontroller Atmega128L with an integrated 10-bit successive approximation ADC.
The used sampling frequency is $f_s = 3.98kHz$ (see Chapter 4). The node is plugged in to the PC in order to deliver data for analyzing. A specific node program has been written in embedded C (nesC) to manage the sensor node.
The goal of the wavelet implementation is to detect when the sine is discontinued, i.e. it represents an anomalous behavior. We use the case study N° 1 as training process or tuned detection from which we decided thresholds levels and by means of which coefficients the detection can be performed.
The following case studies consist in the use of the input signal with few modifications (for this we modify the parameters b_p, f_o, φ) and we try to understand when and why the detection fails.

6.7.1 Unreliable detection: case study N° 1

This example is depicted in **Figure 6.13**. For this case, input signal parameters are as follows: **sine frequency** $f_o = 277Hz$, **burst period** $b_p = 29.33ms$ and **start phase** $\varphi = 0$.
We can see that the anomalous feature presence can be pointed out by the coefficients $d_{-1}(n)$ and $d_{-2}(n)$. Hence, it is possible to define a clear threshold level L_{th} around the value 10 for $d_{-1}(n)$ and 50 for $d_{-2}(n)$ in order to have instantaneous detection of the feature and its associated time-position.
Each threshold value acts as decision surface.

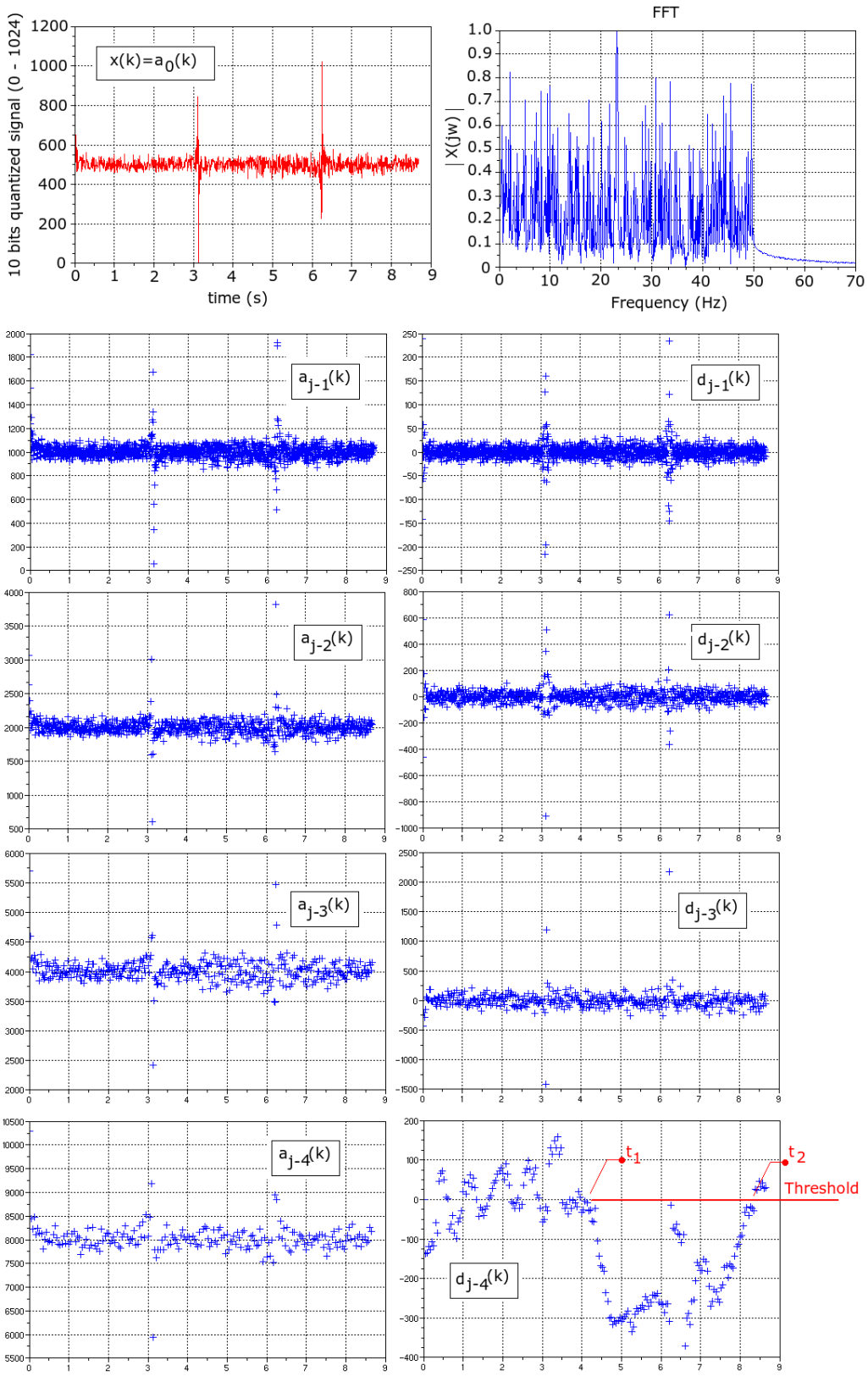


Figure 6.11: **Different decomposition levels or Haar wavelet coefficients for the signal defined by (6.95) and (6.94). At level -4 or by means of d_{j-4} it could be implemented the detection by thresholding.**

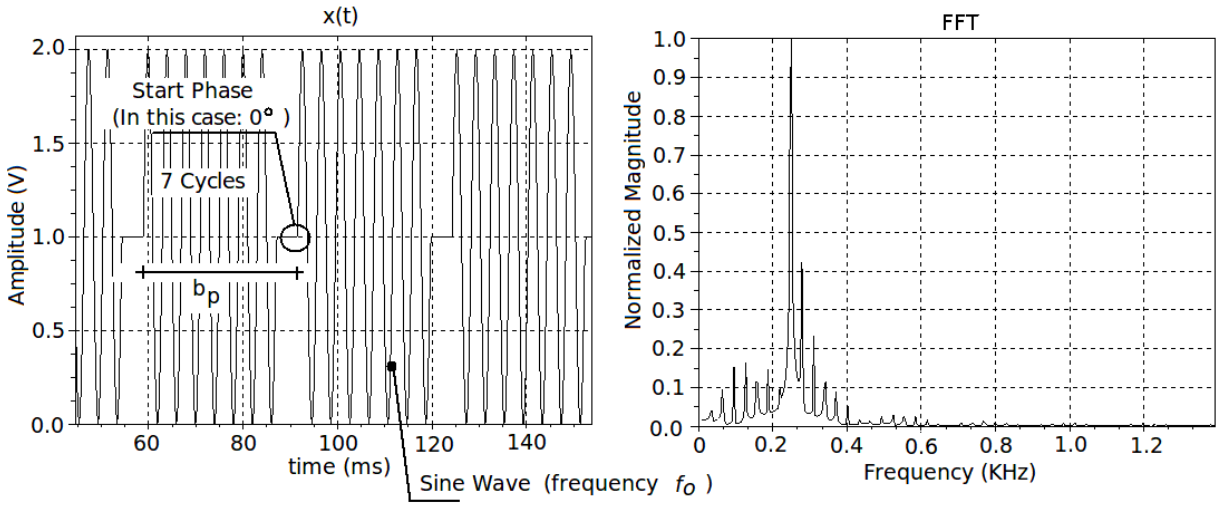


Figure 6.12: **Used reference signal to study problems in the transient signal detection by thresholding**

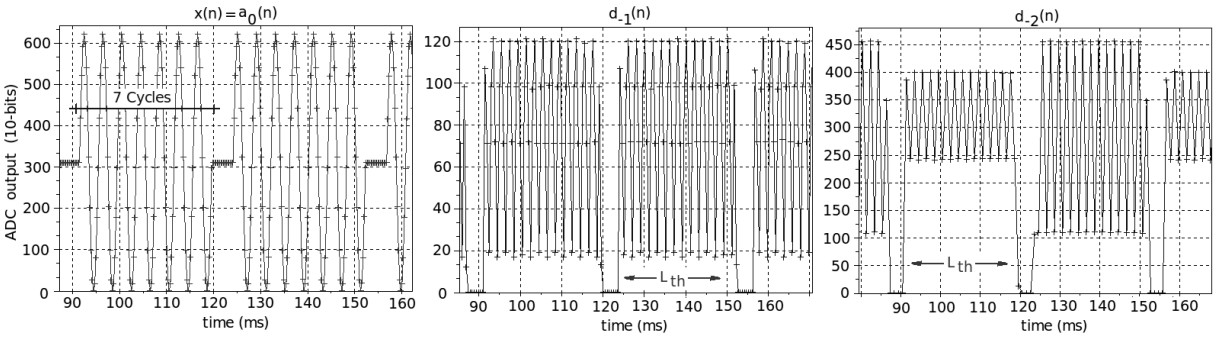


Figure 6.13: **Case study N° 1:** $f_o = 277Hz$**,** $b_p = 29.33ms$ **and** $\varphi = 0$**.**

6.7.2 Unreliable detection: case study N° 2

The example is depicted in **Figure 6.14**. It shows how signal features affects the detection behavior. For this case, input signal parameters are as follows: **sine frequency** $f_o = 275Hz$, **burst period** $b_p = 29.33ms$ and **start phase** $\varphi = 0$.
Please, note that the difference with respect to the case study N° 1 is only the sine frequency, which decreases only $2Hz$..
It is possible to define a clear threshold level L_{th} in order to have instantaneous detection of the feature and its associated time-position only for coefficients $d_{-1}(n)$, in this case $d_{-2}(n)$ fails (lower levels have to be studied in order to see if the detection is possible). Thus, we evidence the thresholding detection sensibility with respect to the signal frequency content. I has been lost one useful threshold at level -2.

6.7.3 Unreliable detection: case study N° 3

The example is depicted in **Figure 6.15**. For this case, input signal parameters are as follows: **sine frequency** $f_o = 272Hz$, **burst period** $b_p = 29.33ms$ and **start phase** $\varphi = 0$.
Instead of case study N° 2, the detection can be performed by the coefficients $d_{-2}(n)$ but not for $d_{-1}(n)$. The level where the detection can be performed changed with respect to the case study N° 2.

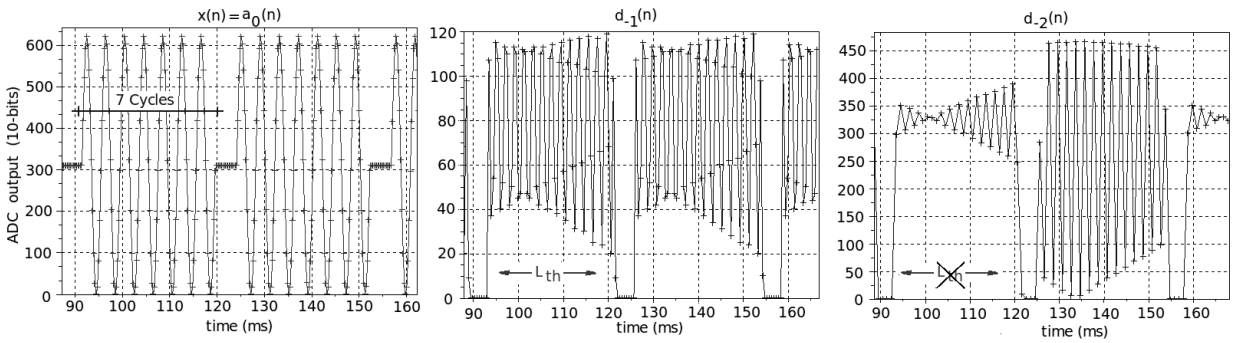


Figure 6.14: **Case study N° 2:** $f_o = 275Hz$, $b_p = 29.33ms$ **and** $\varphi = 0$.

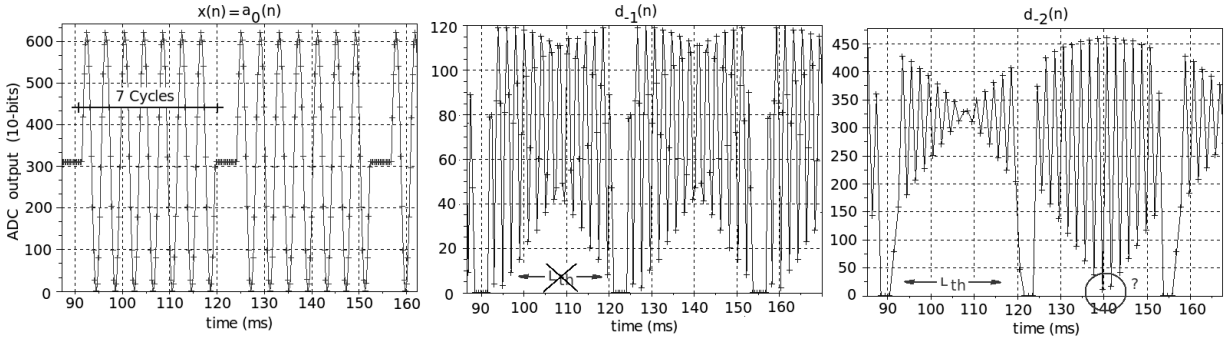


Figure 6.15: **Case study N° 3:** $f_o = 272Hz$, $b_p = 29.33ms$ **and** $\varphi = 0$.

6.7.4 Unreliable detection: case study N° 4

The example is depicted in **Figure 6.16**. For this case, input signal parameters are as follows: **sine frequency** $f_o = 270Hz$, **burst period** $b_p = 29.33ms$ and **start phase** $\varphi = 0$.
The detection cannot be performed in coefficients $d_{-1}(n)$ or $d_{-2}(n)$, both fail (lower levels have to be studied in order to see if the detection could be possible).

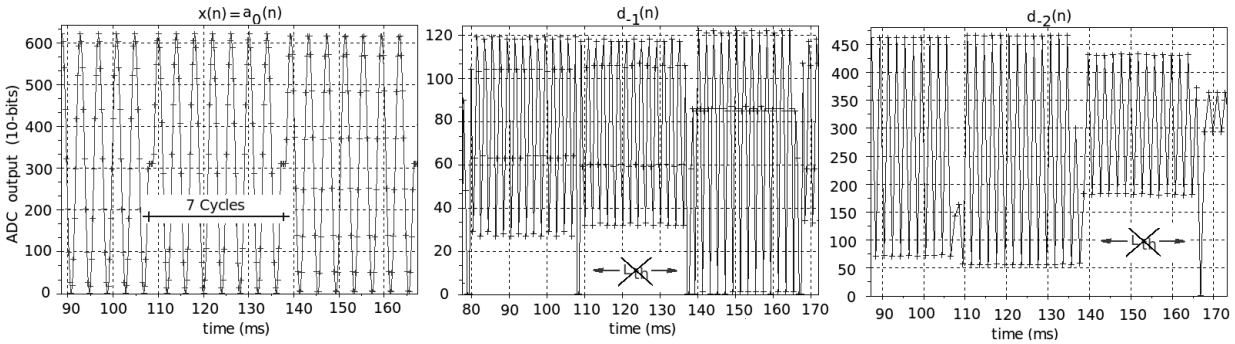


Figure 6.16: **Case study N° 4 :** $f_o = 270Hz$, $b_p = 29.33ms$ **and** $\varphi = 0$.

6.7.5 Unreliable detection: case study N° 5

The example is depicted in **Figure 6.17**. The input signal parameters are as follows: **sine frequency** $f_o = 277Hz$, **burst period** $b_p = 29.334ms$ and **start phase** $\varphi = +66$.
It is possible to define a clear threshold level L_{th} for coefficients $d_{-1}(n)$ in order to have instantaneous detection of the feature and its associated time-position.

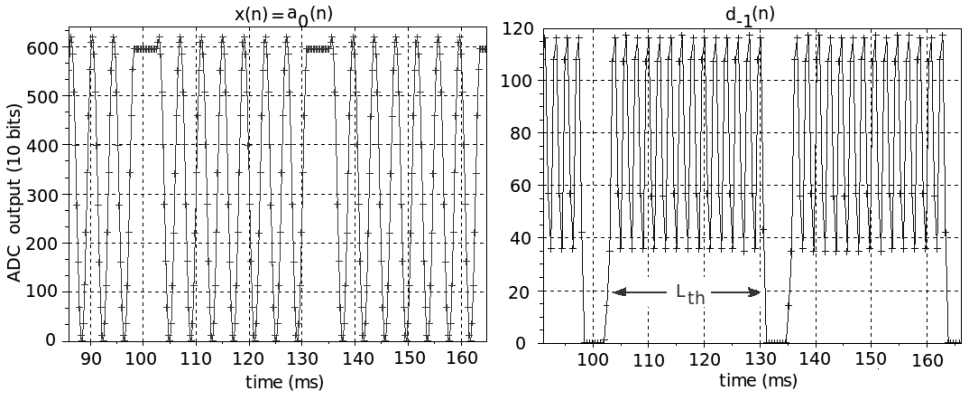


Figure 6.17: **Case study N° 5:** $f_o = 277Hz$, $b_p = 29.334ms$ **and** $\varphi = +66$.

6.7.6 Unreliable detection: case study N° 6

This example is depicted in **Figure 6.18**. The input signal parameters are as follows: **sine frequency** $f_o = 277Hz$, **burst period** $b_p = 26.334ms$ and **start phase** $\varphi = +66$.
The detection could be performed by means of the coefficients $d_{-1}(n)$ but it is not possible in $d_{-2}(n)$.

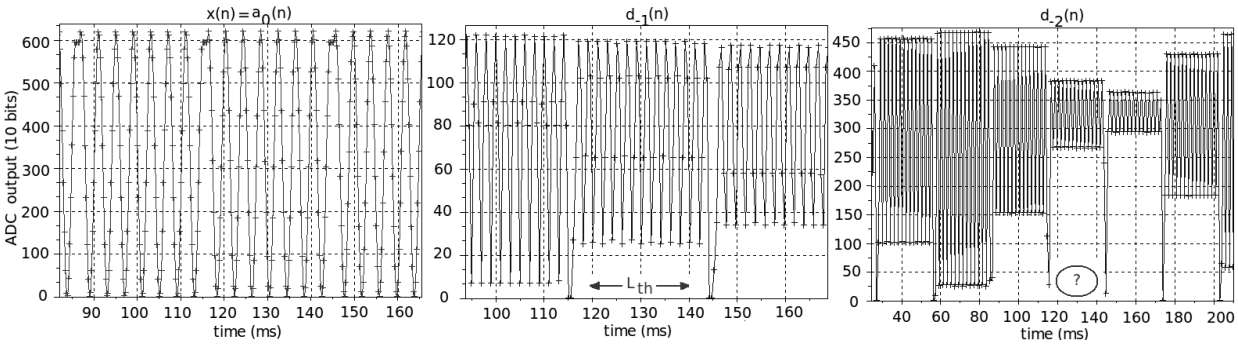


Figure 6.18: **Case study N° 6:** $f_o = 277Hz$,$b_p = 26.334ms$ **and** $\varphi = +66$.

6.7.7 Unreliable detection: case study N° 7

This last example is depicted in **Figure 6.19**. Input signal parameters are as follows: **sine frequency** $f_o = 266Hz$, **burst period** $b_p = 29.334ms$ and **start phase** $\varphi = +66$.
It is not possible the detection with $d_{-1}(n)$ or $d_{-2}(n)$ (lower levels have to be studied in order to see if the detection could be possible).

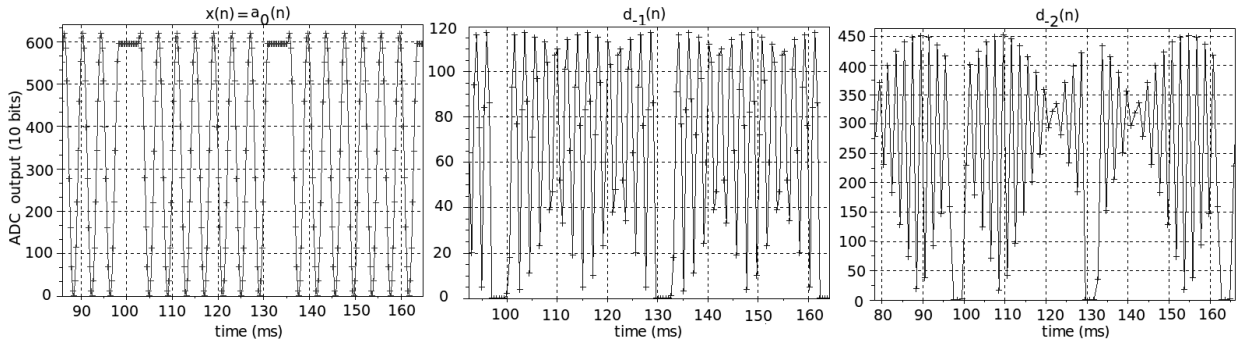


Figure 6.19: **Case study N° 7:** $f_o = 266Hz$, $b_p = 29.334ms$ **and** $\varphi = +66$.

6.7.8 Unreliable detection: detected open issues

Different factors deteriorate the performance of the Haar wavelet, i.e. the detection by means of thresholding. The previous case studies illustrate the problems that we summarize below:

1. *sampling frequency selection:* we observed (after several testing) that the detection capability depends on the selected sampling frequency because it is related to the algorithm startup, i.e. the level of granularity that the coefficients $a_0(n) = x(n)$ can approximate the signal $x(t)$.
   This dependence should be object of study and it has not been found in literature a methodology ensuring the best sampling frequency. This sampling frequency determines the detection capability in the sense that it not only affects where it takes place (i.e. at which decomposition level) but also the threshold value.
   This is an open issue.
2. *threshold and decomposition level selection* . It is not clear how to know in advance the decomposition level where the feature detection can be performed, and for which threshold value. It depends on the signal features and it must be carefully selected for each case by using training signals. The detection by thresholding has to be designed by means of ad-hoc methodologies.
   This is another open issue.

Thus, at this moment does not exist a methodology to select the best sampling frequency and neither methodology to determine the decomposition level where the detection would take place, or to estimate the threshold value that assures the successful detection.

6.7.9 Important advantage: the implementation

Finally, regarding implementation, another important consideration is the execution time. In fact the digital Haar wavelet is well suited for constrained hardware as nodes are, but there is a limitation in the execution time related to the input signal bandwidth.
In **Figure 6.20** it is defined the execution time T and in the **Listening 6.1** is depicted the piece of code used to measure such execution time. For the MicaZ node it results $T = 15.8\mu s$.

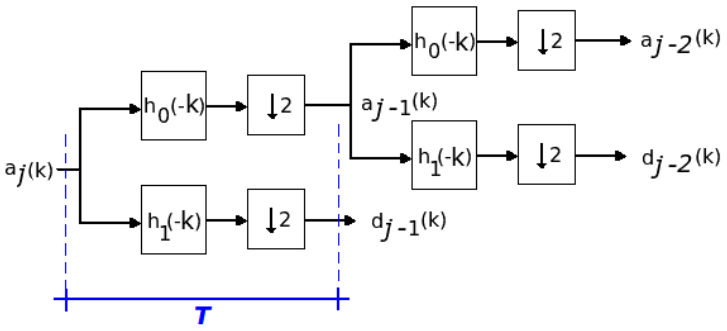


Figure 6.20: **Execution time measurement $T = 15.8\mu s$. It is defined as the required time to completely execute the bank filter.**

We define: f_w the signal bandwidth and Q the number of decomposition levels. Then, the following condition must be held in order to achieve the decomposition level Q in real-time:

$$2f_w < f_s < \frac{Q}{T} \tag{6.96}$$

Numerical examples: for $f_w = 10kHz$ it results $Q \simeq 3$ levels and for $f_w = 1kHz$ it results $Q \simeq 31$ levels. In fact, after level 6 or 7 the detail signal can be considered as noise and levels below 7 in general has no sense for the Haar wavelet case.
Since signals from accelerometers rarely are higher that $1kHz$, the implementation is well suited for application where vibrating signals have to be analyzed.

Finally, to conclude this section it is important to highlight the following ideas.
The detection by thresholding seems to work when it is correctly tuned with training signals. However, as we have seen, the robustness can not be guaranteed, i.e. the detection would fail if signal features are

minimally modified.
Nevertheless, drawbacks are counterbalanced by other factors such as speed and a lot of potential applications where this signal processing technique would be applied. It has great potential with extraordinary computational capability and flexibility.
Rather than discouraging the Haar wavelet, in the following section we attempt to provide robustness and reliability to the detection by thresholding and with the objective of covering a large set of application. The effort is perfectly justified.

Listing 6.1: **Section of the NesC code that implement the first filter bank**

```
1
2  //parameters
3  int32_t    aj1[10];
4  int32_t    dj1[10];
5  int32_t*   paj1=&aj1[0];
6  int32_t*   pdj1=&dj1[0];
7  int32_t*   paj=&aj[0];
8  uint8_t    p1=0;
9  uint8_t    p2=0;
10
11 //piece of program
12
13 SET_BIT(PORTB,PORTB4);                    // set pin  - MARKER (A)
14
15 //start level 1                           //(A)
16 *(paj1+p1)=*(paj + p1)+ *(paj + p2);
17 *(pdj1+p1)=*(paj + p1)- *(paj + p2);
18
19 p2=p2<<1;                                 // *2
20 p1=p2-1;
21 //end level 1
22
23 CLR_BIT(PORTB,PORTB4);                    // set pin  - MARKER (B)
```

6.8 Pulse shaping

The pulse shaping is the process of changing the input waveform. We guess that by implementing a priori known and convenient signal deformation, the Haar DWT detection capability could be enhanced by using thresholding.
The pulse shaping could be implemented in discrete-time (B) (see **Figure 6.21**) or continuous time (C) (analog preprocessing).
Moreover, the pulse shaping can be implemented by the convolution with a mother wavelet or with another impulse response. In the first case, the time-reversed mother wavelet becomes the impulse response of a linear filter, i.e. the continuous wavelet transform (WT).

In the next section we propose a motivational example with real sampled signals. For such example we use the approach (B) of **Figure 6.21**, i.e. the pulse shaping is implemented in discrete-time because at this moment it is not available a circuit capable of implementing the analog preprocessing.
Moreover, having in mind the objective of covering a large set of applications, it is desired to implement the pulse shaping implementation in discrete-time, into the microcontroller in order to feature an acceptable flexibility.

6.8.1 Motivational example: acceleration signal

The Micaz node features the MTS300CB/MTS310CB sensor board with the low-cost 2-g dual-axis accelerometer ADXL202E. The accelerometer output signal is sampled while the node is being carried by a person who walks, runs and takes pauses. Sampled data are analyzed by the DWT implemented in the PC. The objective is to detect when the person has taken short pauses.

The acceleration pattern has been captured without special accelerometer orientation, i.e. the node was carried with aleatory orientation in the shirt's pocket. We measured the output signal from the axis X of the accelerometer, however, we don't know the relative position of such axis X with respect to the movement.

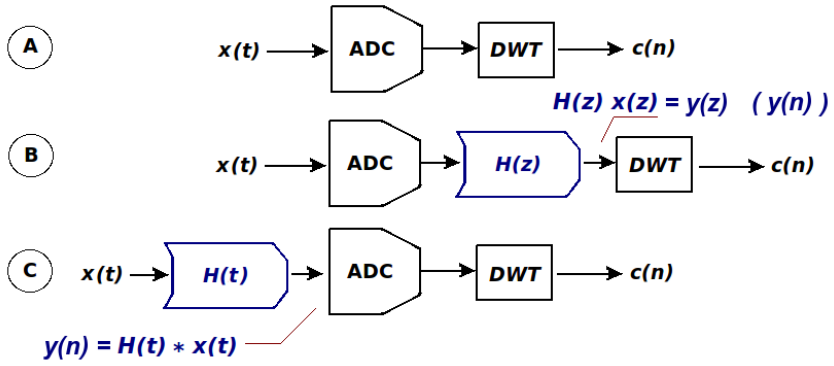


Figure 6.21: **Analog or time-discrete pulse shaping before implementing** DWT. $x(t)$ **is the input signal,** $c(n)$ **the coefficients after performing the** DWT. **Case** (A)**: typical case, Case** (B) **discrete-time pulse shaping and Case** (C) **analog pulse shaping.**

It was done in order to get insight about the detection capability of the Haar DWT, i.e. the perceptual quality of the detection should not drop significantly due to spatial variances of the accelerometer.
Two case studies have been analyzed: Case 1 (walk) and Case 2 (walk and run). Case 1 is used for training the impulse response (or training stage) i.e. for computing the required pulse shaping filter implementation $h(n)$ or $H(z)$, see system B in **Figure 6.21**.
This filter $h(n)$ is then applied to the Case 2 in order to enhance the Haar DWT detection capability. The data acquisition features were: number of samples $N = 1000$, sampling frequency signal $f_s = 20\,Hz$, $(T_s = 0.05\,s)$.

Case 1: walk

In **Figure 6.22, graphic 1** it is depicted the accelerometer output signal captured by the node. The sections marked as (A), (C) and (E) correspond to the normal walking; on the other hand, the transient signals marked as (B) and (D) show that the person takes short pauses for direction changing, i.e. the person suddenly stops and changes direction to take another path to avoid collision with the wall (in (B) and (D)).

We observed that it has been **impossible** to detect the transients (B) and (D) by using only the DWT. More specifically, system (A) in **Figure 6.21** has failed.

For this, we calculated a discrete-time impulse response $h(n)$ in order to be applied to the input signal as is shown in **Figure 6.21, system (B)** (the calculus is omitted but the methodological approach is addressed below and the impulse response is shown in **Figure 6.23**).
After performing the discrete-time convolution $x(n) * h(n)$, we applied the DWT and the resulting coefficients $d_{j-2}(n)$ are shown in **Figure 6.22, graphic 2**. Now, both signal transients **can be easily identified** and the events are marked with B and D. Excellent results.

A point to remark is related to the graphic representation. In **Figure 6.22, graphic 3** it is used the time $4T_s$ between two consecutive coefficients $(d_{j-2}(n), d_{j-2}(n+1))$ in order to plot and to compare by using the same input signal time interval. It is due to the number of obtained coefficients $d_{j-2}(n)$ is $N/4$, i.e. each level discharges the half of the previous coefficients.

Finally, the used discrete-time impulse response $h(n)$ and the associated DFT is depicted in **Figure 6.23**.
In this case we use a real sampled signal and the discrete pulse shaping was designed by calculating the coefficients $h(n)$. In the next section, the same coefficients $h(n)$ (i.e. same pulse shaping) are applied to another different input signal where the same event of *stop-person* is present.
Results show that the event can be detected.

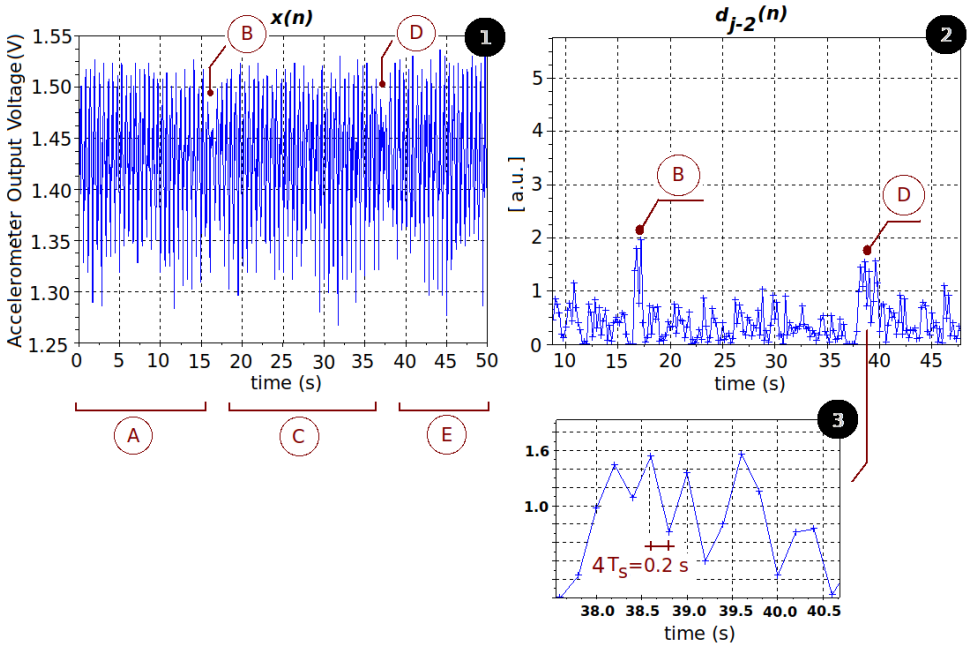


Figure 6.22: **Case 1: walk. Graphic 1 is the recorded person acceleration while walking. Graphic 2: DWT coefficient $d_{j-2}(n)$ after performing the pulse shaping. Graphic 3: example of the graphic representation, $d_{j-2}(n)$ is composed by $N/4$ coefficients and time between coefficients is $4T_s$**

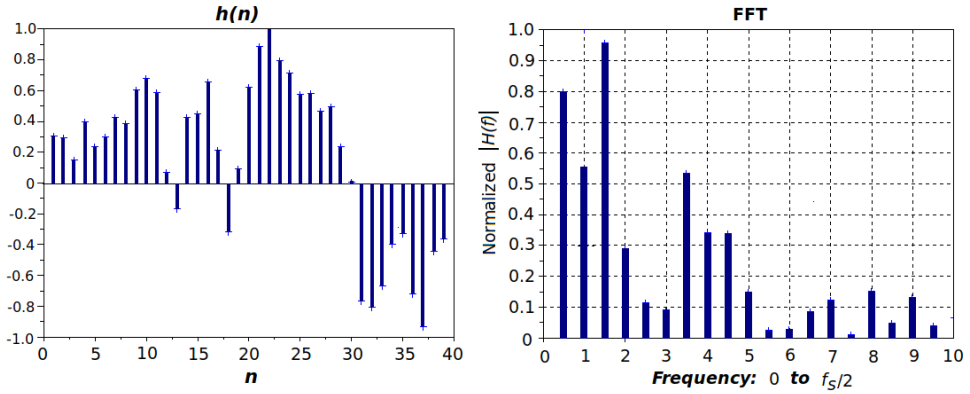


Figure 6.23: **Case 1: walk. Impulse response $h(n)$ and its associated FFT. This impulse response has been used to enhance the detection of the features as it is depicted in Figure 6.22, sufigure 2**

Case 2: walk and run

In **Figure 6.24** it is possible identify the following patterns: (A) normal walking, (B) the person stops for changing direction and immediately starts to run, (C) still running $10\,s$, D stop and long pause, (E) the person starts to walk again.
Data acquisition features: number of samples $N = 1000$, sampling frequency $f_s = 20\,Hz$,$(T_s = 50\,ms)$.
It is important to underline that at time $10\,s$, the person (i.e. me) change direction. Please, observe that the event marked (T) (zoom at right side) is not clearly identified as instead the human eyes can simply detect the transients in **Figure 6.22**, subfigure 1 (input signal).
After applying the DWT the transition (T) can not be revealed. However, by using the pulse shaping, all signal transients seem able to be detected in **(Figure 6.25)** by thresholding.

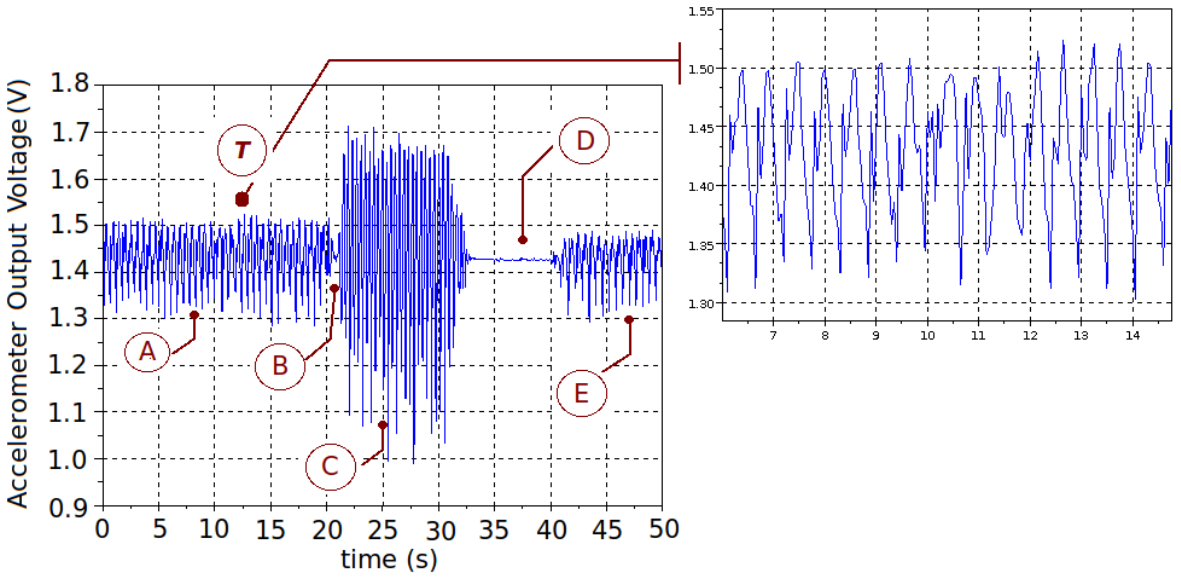


Figure 6.24: **Case 2: walk and run. Recorded acceleration for human walk and run. At the right side a signal zoom shows that there is not visible signal transient due to the direction change.**

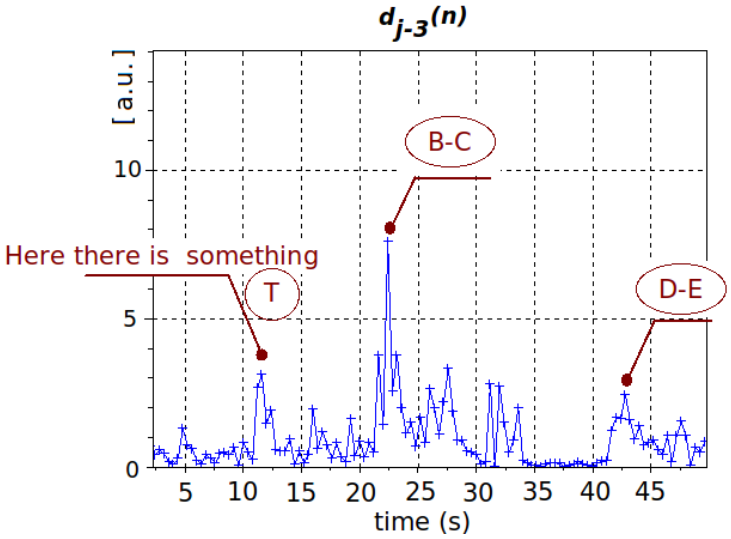


Figure 6.25: **Case 2: walk and run. Haar DWT coefficients after using pulse shaping. All transient signals can be detected by means of the DWT at level -3 or $d_{j-3}(n)$ after the pulse shaping implementation.**

6.9 Methodology to calculate $h(n)$ for time-discrete pulse shaping

6.9.1 Methodology description

In this section we describe the methodology to calculate the desired impulse response $h(n)$. The function of the pulse shaping consists in to enhance significantly the contrast between the transient signals and the rest of the signal. This strategy improves the detection capability for the used wavelet transforms (i.e. Haar).
We have addressed a heuristic approach and further comparisons as well as studies should be addressed in order to validate the methodology.
To have better insight, without loss of generality, we illustrate the used methodology for the previous described example and which essentially has the following concepts:

1. The acceleration signal $x(n)$ of **Case 1** was used as input reference signal for training purposes. The signal consists of N samples, i.e. the size of the vector $x(n)$ is N.
2. We identified (in fact it is assigned by our subjective observation) where the signal exhibits transitions. The first one is detected in $n = N_{x1}$ with length d_{x1} and the second one is observed in $n = N_{x2}$ with length d_{x2} as it is depicted in **Figure 6.26 graph 1**.
   Accordingly, the parameters for this case are: $N_{x1} = 318$, $d_{x1} = 15$, $N_{x2} = 734$, $d_{x2} = 25$.
3. An appropriate impulse response $h(n)$ is calculated in order to obtain a desired output signal $y(n) = h(n) * x(n)$ **only** for the sections where the transient signals have been detected. Just for this case, we need constant output and the result is shown in **Figure 6.26, graph 2**. The impulse response has size L, with $L = d_{x1} + d_{x2}$. The resulting signal $y(n)$ has size $N + L - 1$.
   The impulse response estimation is not an insurmountable problem. It is performed by writing and resolving the convolution equation in the matrix form as is described in the next section.
   (Please observe that it is not guaranteed the stability of the impulse response. We mention that the study about the discrete impulse response stability must be further carried out.)

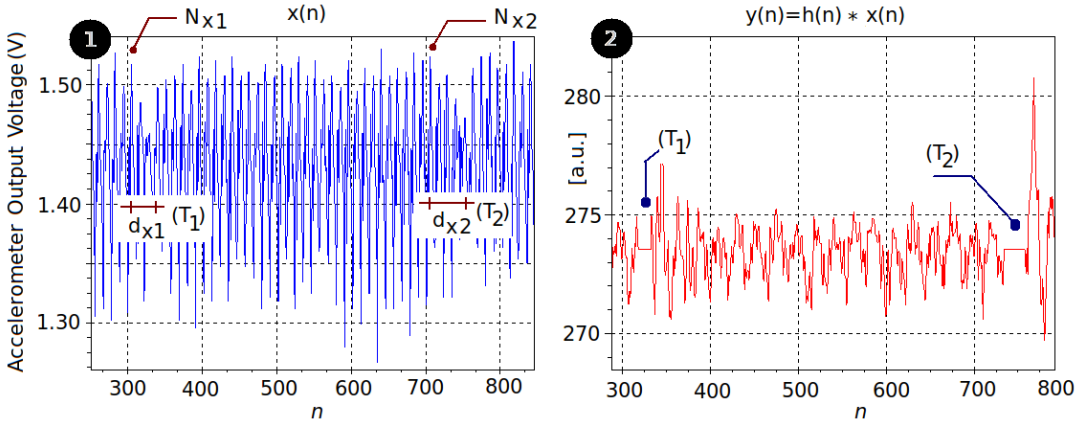


Figure 6.26: **The signal $x(n)$ used for calculating $h(n)$ (graph 1, left). Graph 2, right is the signal $y(n)$ where it is shown the sections with the transients after the convolution $y(n) = h(n) * x(n)$. (The notation [a.u] means adimensioned units, i.e. it is the resulting integers after applying the convolution in a fixed-point arithmetic)**

6.9.2 Impulse response $h(n)$ calculation

The canonical matrix representation for the convolution product between a finite set of time-discrete signal and impulse response is (6.97):

$$\begin{bmatrix} h(1) & 0 & 0 & \cdots & 0 & 0 \\ h(2) & h(1) & 0 & \ddots & \vdots & \vdots \\ h(3) & h(2) & h(1) & \ddots & \ddots & \\ \vdots & h(3) & h(2) & \ddots & \ddots & \vdots \\ \vdots & \vdots & h(3) & & \ddots & 0 \\ h(L) & \vdots & \vdots & & h(1) & 0 \\ 0 & h(L) & \vdots & & h(2) & h(1) \\ 0 & 0 & h(L) & & & \\ \vdots & \vdots & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & h(L-1) & h(L-2) \\ & & & & h(L) & h(L-1) \\ 0 & 0 & 0 & \cdots & 0 & h(L) \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \vdots \\ \vdots \\ x(N) \end{bmatrix} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ \vdots \\ y(N+L-1) \end{bmatrix} \tag{6.97}$$

Or, more compactly written:

$$\mathbb{H}.x = y \tag{6.98}$$

Where $\mathbb{H}$ is the previous described matrix of size $(N+L-1)-by-(N)$.
However, for our purpose is more convenient to express the convolution as (6.99), because the vector $h(n)$ is our vector of the variables, which can be resolved by using the Cramer rule for solving systems of equations.

$$\begin{bmatrix} x(1) & 0 & \cdots & & & 0 & 0 \\ x(2) & x(1) & 0 & \cdots & & 0 & 0 \\ x(3) & x(2) & x(1) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & & \vdots & \vdots \\ x(L-1) & x(L-2) & x(L-3) & \cdots & \cdots & x(1) & 0 \\ x(L) & x(L-1) & x(L-2) & \cdots & \cdots & x(2) & x(1) \\ x(L+1) & x(L) & x(L-1) & \cdots & \cdots & x(3) & x(2) \\ \vdots & \vdots & & & & \vdots & \vdots \\ 0 & x(N) & x(N-1) & \cdots & \cdots & x(N-L+2) & x(N-L+1) \\ \vdots & \vdots & \vdots & & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & x(N) & x(N-1) \\ 0 & 0 & 0 & & \cdots & 0 & x(N) \end{bmatrix} \cdot \begin{bmatrix} h(1) \\ h(2) \\ \vdots \\ \\ h(L) \end{bmatrix} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ \vdots \\ \\ y(N+L-1) \end{bmatrix} \tag{6.99}$$

And so write:

$$\mathbb{X}.h = y \tag{6.100}$$

Where, $\mathbb{X}$ is the sampled signal matrix, h is the vector of the sampled impulse response (our variable in the equation) and y is the imposed output signal that will be processed by the Haar DWT. This is the waveform we need at the input of the Haar DWT.
One last comment is in order from the implementation point of view; it is defined the vector V as:

$$V = [zeros(1:L-1), \hat{x}(1:N), zeros(1:L-1)] \tag{6.101}$$

where $\hat{x}$ is the swap vector x, i.e. $\hat{x}(n) = x(N+1-n)$. The size of V is $1-by-(N+2L-2)$ (The notation $zeros(1:L-1)$ means a row vector of size $L-1$ where each element is assigned to be 0).

Therefore, the $j-th$ row of $\mathbb{X}$ is $\mathbb{X}[j,1:L] = V[N+L-j:2L-2+N+1-j]$, where $j \in [1, N+L-1]$, i.e. every row of $\mathbb{X}$.
This property is exploited to build the algorithm that computes the impulse response, and which it is below described in the **Algorithm 2**. We have implemented the algorithm in $SCILAB$ in order to have an automatized procedure for computing the impulse response $h(n)$. The notation is the required for programing in $SCILAB$.

In this section, we explore the pulse shaping in discrete-time, system (B) in **Figure 6.21**. Results are promising and motivated by that, we suggest to the developers more research towards this direction. Nevertheless the methodology is still rather ad-hoc and results could depend on the specific application. In the next section we study the feasibility for implementing pulse shaping in analog domain, i.e. case C of **Figure 6.21**.

6.10 Approaches for implementing analog pulse shaping

6.10.1 Problem statement

Adding the analog pulse shaping capability to wireless sensors node (see **Figure 6.21**, system C), requires the design of analog filters able to implement impulse responses that best highlight the signal feature desired to detect.
Moreover, it is important to remark that such filter for analog pulse shaping must meet low-voltage and low-power consumption requirements.

In this stage we face two problems which are about:

Algorithm 2 Calculate $h(n)$ for the case of **Figure 6.26**

$N = length(x)$;
$N_{x1} = 318$;
$d_{x1} = 15$;
$N_{x2} = 734$;
$d_{x2} = 25$;
$L = d_{x1} + d_{x2}$;
$Nc = N + L - 1$;
$vlarge = 2 * L - 2 + N$;
for $p = 1$ to N **do**
  $xswap(p) = x(N + 1 - p)$;
$vector = [0 * ones(1 : L - 1), xswap', 0 * ones(1 : L - 1)]$;
for $j = 1$ to $N + L - 1$ **do**
  $start = N + L - j$;
  $end = 2 * L - 2 + N + 1 - j$;
  $Arow(j, 1 : L) = vector(start : end)$;
for $q = 1$ to d_{x1} **do**
  $M(q, 1 : L) = Arow(N_{x1} - 1 + q, 1 : L)$;
for $q = d_{x1} + 1$ to $d_{x1} + d_{x2}$ **do**
  $M(q, 1 : L) = Arow(N_{x2} - d_{x1} - 1 + q, 1 : L)$;
for $q = 1$ to L **do**
  $y_{out}(q) = y_d(n)$;
$delta = det(M)$; {Cramer to resolve the system}
for $q = 1$ to L **do**
  $H = M$;
  $H(1 : L, q) = y_{out}$;
  $h(q) = (1/delta) * det(H)$;
$h = h'$; {Algorithm ended} =0

1. the impulse response identification
2. the design of the circuit able to implement such impulse response.

The problem *1* can be resolved in discrete time as it has been previously presented in equations (6.99) and (6.100). In fact, the identification of the impulse response to be applied to the input signal in order to enhance the capability detection has been successfully tackled in the previous section.
Once the required impulse response is calculated, a plausible circuit implementation can be addressed. In the following section we explore how to face the problem *ii)*.

In literature methodologies and techniques to design filters with a desired impulse response are offered and discussed. Such techniques can achieve the best approximation which at the same time yield stable and physically realizable filters. In this section we offer an insightful guide with the key ideas offered in [**Baker, G.A. (1975)**], [**Agostinho, P.R. (2008)**], [**Haddad, S.A.P (2005)-1**] and [**Qingxiu, H. (2003)**].

6.10.2 Laplace domain: Pade approximation

The concept of the Pade approximation is similar to the Taylor approximation. It consists in the approximation of a function around one point but by means of a rational expression. The theory is well explained in **[Baker, G.A. (1975)]** and below, we offer a brief concept summary.
The Pade approximation can be stated as follows:

For a function $h(t)$ ($h(t) \in \mathbb{Z}$), the Taylor expansion of the Laplace transform $H(s)$ around $s = 0$ up to order k is:

$$\widehat{H}(s) = c_0 + c_1 s + ... + c_k s^k + O(s^{k+1}) \tag{6.102}$$

$$c_k = \frac{\widehat{H}^{(k)}(0)}{k!} \tag{6.103}$$

Then, the Pade approximation is the rational approximation of the truncated Taylor series (6.102) as follow:

$$\widehat{H}(s) = c_0 + c_1 s + ... + c_k s^k = \frac{P(s)}{Q(s)} = \frac{p_0 + p_1 s + ... + p_m s^m}{q_0 + q_1 s + ... + q_n s^n} \tag{6.104}$$

Where, the numerator is of order m, the denominator of order n and $k = m + n$. The parameters m, n, k would be selected by the designer, however k should be large enough to assure an accurate approximation for the required application.
Moreover, the most important restriction is that, in order to obtain a causal filter it must be $m \leq n$.
The coefficients for the polynomials $P(s)$ and $Q(s)$ are calculated by resolving the convolution equation in its matrix form as follows **[Haddad, S.A.P (2005)-1]**:

$$\begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & \vdots \\ \vdots & c_1 & \ddots & 0 \\ \vdots & \vdots & \ddots & c_0 \\ \vdots & \vdots & & c_1 \\ \vdots & \ddots & \ddots & \vdots \\ c_k & c_{k-1} & \cdots & c_{k-n} \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ \vdots \\ \vdots \\ p_k \end{bmatrix} \tag{6.105}$$

The solution is:

For the coefficients of $P(s)$:

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ \vdots \\ \vdots \\ p_m \end{bmatrix} = \begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_1 & c_0 & \cdots & \vdots \\ \vdots & c_1 & \ddots & 0 \\ \vdots & \vdots & \ddots & c_0 \\ \vdots & \vdots & & c_1 \\ \vdots & \ddots & \ddots & \vdots \\ c_m & c_{m-1} & \cdots & c_{m-n} \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_n \end{bmatrix} \tag{6.106}$$

Where $c_k = 0$ for $k < 0$ (because $m \leq n$ for causal filters) and with:

$$\begin{bmatrix} p_{m+1} \\ p_{m+2} \\ \vdots \\ p_k \end{bmatrix} = 0 \tag{6.107}$$

For the coefficients of $Q(s)$:

$$\begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_n \end{bmatrix} \in Nullspace \begin{bmatrix} c_{m+1} & \cdots & c_0 & 0 \\ c_{m+2} & \cdots & c_1 & c_0 \\ \vdots & & & \vdots \\ c_{m+n} & \cdots & & c_m \end{bmatrix} \tag{6.108}$$

Or, written also as:

$$\begin{bmatrix} c_{m+1} & \cdots & c_0 & 0 \\ c_{m+2} & \cdots & c_1 & c_0 \\ \vdots & & & \vdots \\ c_{m+n} & \cdots & & c_m \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_n \end{bmatrix} = 0 \tag{6.109}$$

The Pade approximation has advantages and drawbacks as it is described in **[Agostinho, P.R. (2008)]** and **[Haddad, S.A.P (2005)-2]**.
We summarize the most relevant aspect and considerations to be taken into account, among others:

- The Pade approximation is mathematically simple and general.
- To apply the rational Pade approximation to an impulse response $h(t)$, it is necessary to calculate the Laplace transform $H(s)$. A problem which frequently arises is the difficulty for the $H(s)$ calculus.
- The Pade approximation of an impulse response $h(t)$ yields immediately to the circuit implementation because it represents the transfer function (in Laplace), however, to conclude if the rational approximation is physically realizable, it must be verified the stability. **The Pade approximation does not guarantee stability itself**.
- The approximation is concentrated around a particular point s_0, in accordance to the Taylor approximation. In general, such particular point is assumed to be $s_0 \to \infty$. (The Taylor expansion can not be used to approximate impulse responses because it has only zeros).
- The Pade approximation is better than others methods such as the *CFOS - Complex First Order Systems*. Nonetheless, the approximation can encounter convergence problems to approximate impulse responses which involve many oscillations.

Interested readers can read noteworthy examples in **[Haddad, S.A.P (2005)-1]**, where authors have calculated the Pade coefficients of the first and second derivative Gaussian with: Taylor expansion of order $k = 16$, $Q(s)$ coefficients $n = 10$ and $P(s)$ coefficients $n = 6$.

6.10.3 Time domain: L_2 approximation

The L_2 approximation is another technique to estimate impulse response approximations. It could be done in time domain or Laplace domain. In **[Agostinho, P.R. (2008)]**, the methodology is described and used for implementing wavelet filters. In **[Karel, J.M.H. (2005)]**, key issues related to the methodology are presented and discussed.
The L_2 methodology consists in minimizing the least-mean-square-error. Such error is expressed in integral form, taking into account the difference between the desired function $h_d(t)$ and the associated approximation $h(t)$.

$$\|h_d(t) - h(t)\|^2 = \int_0^\infty \left(h_d(t) - h(t)\right)^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} \left(H_d(j\omega) - H(j\omega)\right)^2 d\omega \tag{6.110}$$

In literature, it is customary to assume that the function $h_d(t)$ is the filter impulse response that implements a wavelet filtering and $h(t)$ is intended to be the approximate impulse response.
Below, we offer a summary of the issue which commonly is faced and discussed in literature (for additional notes, please see **[Agostinho, P.R. (2008)]**), however, as persuaded in the next section, **we restate the methodology and formulate another approach.**

It is well known that the impulse response of a stable causal linear filter can be expressed in time domain as:

$$h(t) = \sum_{k=1}^{n} A_k e^{\lambda_k t} \qquad \lambda_k,\ A_k \in \mathbb{Z}, \quad Re(\lambda_k) < 0 \tag{6.111}$$

The minimization of (6.110) is carried out with respect to the poles and zeros of (6.111). It can be solved by choosing suitable numerical optimization techniques, nevertheless, in literature is observed that there is risk of obtaining local minimums as solution, instead of the global optimum.
For this, and since in general the use of a low order filter is enough to obtain accurate approximations, it is often used the following parametrized form for expression (6.111) **[Karel, J.M.H. (2005)]** :

$$h(t) = \sum_{k=1}^{n_1} c_k e^{-q_k t} + \sum_{k=n_1+1}^{n} e^{-p_k t}[a_k \sin(\omega_k t) + b_k \cos(\omega_k t)] \tag{6.112}$$

Where, all parameters $\in \mathbb{R}$, and for stability reasons it must be $q_k,\ p_k > 0$. As an example, one approximation expressed as (6.112) could be:

$$h(t) = c_1 e^{-q_1 t} + e^{-p_1 t}[a_1 \sin(\omega_1 t) + b_1 \cos(\omega_1 t)] + e^{-p_2 t}[a_2 \sin(\omega_2 t) + b_2 \cos(\omega_2 t)] \tag{6.113}$$

The Laplace transform of the building functions of (6.113) are listed below for the reader convenience:

- $\mathfrak{L}\left\{e^{-q_1 t}u(t)\right\}= \dfrac{1}{s+q_1}$, convergence for $Re\{s\} > -q_1$
- $\mathfrak{L}\left\{e^{-p_1 t}\sin(\omega_1 t)u(t)\right\}= \dfrac{\omega_1}{(s+p_1)+\omega_1^2}$, convergence for $Re\{s\} > -p_1$
- $\mathfrak{L}\left\{b_1 e^{-p_1 t}\cos(\omega_1 t)u(t)\right\}= \dfrac{s+\omega_1}{(s+p_1)+\omega_1^2}$, convergence for $Re\{s\} > -p_1$

Finally, it should be remarked two important requirements **[Karel, J.M.H. (2005)]**: *i)* all impulse responses have to be truncated at $t = 0$ for implementing causal filters, and *ii)* it must be satisfied that $H(0) = 0$ because the impulse response must tend to zero for large t in order to avoid unwanted bias in the approximation. It is similar to express that $\lim_{t\to\infty}\int_0^t h(\tau)d\tau = \lim_{s\to 0} H(s)$.
It adds conditions on the parameters of the approximate impulse response $h(t)$ in (6.112).

6.10.4 Pulse shaping: our proposed approach

We employ a different approach with respect to that it was presented in the previous section. From the parametrized form (6.112) we only consider pure exponential terms, with the sampled form $c_k e^{-q_k n T_s}$. Different exponential terms combinations can be used to obtain different performances in terms of the impulse response approximation accuracy. Moreover, the calculus of the parameters c_k and q_k can be best tackled in absence of exponentially damped harmonics.

The objective for using this modified approach are:

- we attempt to use physically realizable filters with the order as small as possible
- we attempt to take advantage of log-domain integrators as building blocks because such circuits are able to work with reduced voltage supply. It is perfectly in line with the issues discussed in **Chapter 5**, i.e the tendency of implementing the entire node on a single chip with the use of deep-submicron CMOS technologies results in the design of embedded mixed-signal systems powered by reduced voltages.
  Thus we take into account possible reduced voltage supply scenarios for our analog pulse shaping implementations.
- instead of expression (6.112) we use the following parametrized form for the approximate impulse response:

$$h(t) = \sum_{k=1}^{M} c_k e^{-q_k t} \qquad k = 0, 1, 2, 3, \cdots, M \qquad q_k > 0 \qquad \forall\ t > 0 \tag{6.114}$$

In **Figure 6.27**, it is depicted three examples that introduce insight about the classes of waveforms capable to be obtained by using the parametrized form (6.114). Where:

- *i)* for $h_1(t)$ the parameters are: $\{c_k\}= [0.1, -0.1, 1, -1, 0.5, -0.5, -5, 5]$ (without units) and $\{q_k\}= [-2, -4, -6, -8, -10, -38, -6, -40] \times 10^3\ seg^{-1}$.
- *ii)* for $h_2(t)$ the parameters are: $\{c_k\}= [0.1, -0.1, 1, -1, 0.5, -1, 2, -1.5]$ (without units) and $\{q_k\}= [-15, -6, -9, -12, -15, -18, -15, -15] \times 10^3\ seg^{-1}$;
- *iii)* and $h_3(t) = h_1(t) + h_2(t)$.

Then, the least-mean-square-error to be minimized can be expressed as:

$$\varepsilon(t) = [h_d(t) - h(t)]^2 = \left[h_d(t) - \sum_{k=1}^{M} c_k e^{-q_k t}\right]^2 \tag{6.115}$$

Where $h_d(t)$ is the desired impulse response and $h(t)$ its associated approximation. Keep in mind that the designer can arbitrarily choose the M value, in accordance to the accuracy required for the approximation. It could be possible to implement specific algorithms for minimization, such as hill-climbing or gradient based searching methodologies, however for this is needed the knowledge of the expression $h(t)$ and its derivative $\forall\, t$.

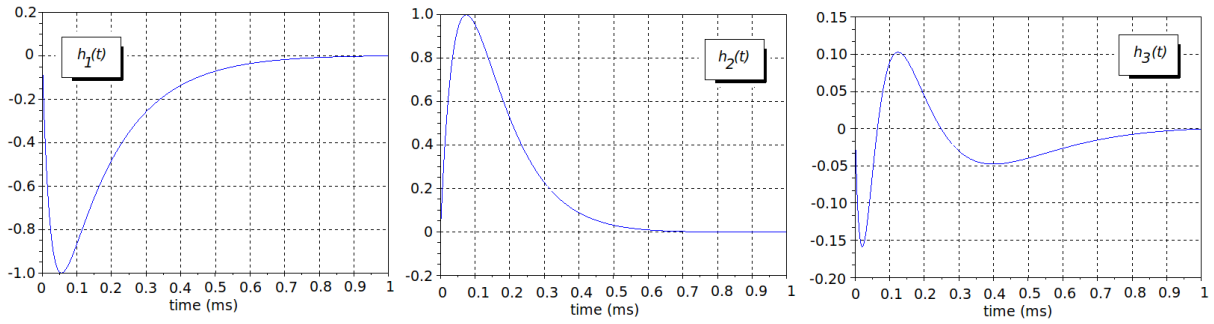


Figure 6.27: **Examples of waveforms obtained with the parametrized form (6.114)**

Needless to say, the minimization problem has to be tackled in discrete-time, because we work under the assumption that we only have $h(t)$ samples at time $t = nT_s$, i.e. $h(nT_s)$. In fact we explained above how to calculate $h(nT_s)$.

Then, the parameters c_k, q_k and M must be estimated in order to achieve the minimum least-mean-square-error, i.e since ε is a vector of size N, it must be performed the minimization of $\|\varepsilon\|^2$:

$$\varepsilon(n) = \left[h(n) - \sum_{k=1}^{M} c_k e^{-q_k n T_s} \right]^2 \tag{6.116}$$

In matrix notation, the expression 6.116 can be written as:

$$\begin{bmatrix} \left[h_d(1) - \sum_{k=1}^{M} c_k e^{-q_k T_s}\right]^2 \\ \left[h_d(2) - \sum_{k=1}^{M} c_k e^{-q_k 2T_s}\right]^2 \\ \vdots \\ \vdots \\ \left[h_d(N) - \sum_{k=1}^{M} c_k e^{-q_k N T_s}\right]^2 \end{bmatrix} = \epsilon = \begin{bmatrix} \epsilon(1) \\ \epsilon(2) \\ \vdots \\ \vdots \\ \epsilon(N) \end{bmatrix} \tag{6.117}$$

The goal is to estimate the parameters $\{c_k\}$, $\{q_k\}$ and M to achieve the minimum $\|\epsilon\|$. In other words it must be resolved the following problem:

$$\begin{array}{c} \arg\min \|\epsilon\| \\ c_k, q_k, M \in \mathtt{C_R} \end{array} \tag{6.118}$$

Where $\mathtt{C_R}$ means the set of parameters that lead physically realizable circuits structures (please see in advance expressions (6.141) and (6.142)).
Study of implementation issues related to the algorithm for the minimization has not been conducted. Instead, we have studied the circuit implementation feasibility because this is the real bottleneck for the analog pulse shaping methodology.
The reason is clear, under the hypothesis that equation (6.117) has feasible solution, one should emphasize that the ultimate goal is to implement circuits with reduced power consumption.
Thus, the undertaken strategy was to perform a preliminary study about possible circuit implementations by using log-domain integrators.

6.11 Building blocks: the log-domain integrator

In this section we present the circuit that implements the building term $c_k e^{-q_k t}$ (see expression (6.114)). Its Laplace transform is (first-order low-pass):

$$H(s) = \frac{c_k}{s + q_k} \tag{6.119}$$

We use log-domain circuits because they implement linear input-output filtering by taking advantage of the devices' non-linearities, i.e. the MOS transistors operate with their exponential characteristics (i.e.

nonlinear large signals). The main advantages lead in: *i)* enhancement of the signal dynamic range; *ii)* linearization schemes are not required and; *iii)* operating capabilities with low power supply levels.
The theory is analyzed and discussed in literature, e.g. in [**Toumazou, C. (2002)**],[**Vittoz, E.A. (2003)**] and [**Serra-Graells, F. (2003)**].

First of all, we offer a summary of the well-known DC drain current expressions for n-channel MOS transistors in weak inversion provided by **[Serra-Graells, F. (2003)] - Table 2.1**.

For p-channel MOS transistors, the following changes must be done: *i)* it has to be change the notation for the voltages because the bulk (B) becomes in the positive reference, more specifically, e.g. the voltage $V_{GB} = V_G - V_B$ used for n-channel MOS transistors ought to be modified as $V_{BG} = V_B - V_G$ and so on; *ii)* an output current drain is considered positive (in the n-channel MOS transistors an input current drain is considered positive); *iii)* it must be used $V_{th,p}$, which is negative.

Weak inversion; the condition to reach weak inversion is expressed as: V_{SB} and $V_{DB} \gg \frac{V_{GB} - V_{th,n}}{n}$
(In **[Vittoz, E.A. (2003)]** it is also expressed as $\frac{V_{GB} - V_{th,n}}{n} - V_{(SB),(DB)} \ll U_T$)

- Conduction:
  $|V_{DB} - V_{SB}| \ll U_T, \quad (I_F \sim I_R) \quad \Rightarrow \quad \mathrm{I}_D = I_{Do} \exp\left(\frac{V_{GB}}{nU_T}\right)\left[\exp\left(-\frac{V_{SB}}{U_T}\right) - \exp\left(-\frac{V_{DB}}{U_T}\right)\right]$
- Forward Sat.
  $(V_{DB} - V_{SB}) \gg U_T, \quad (I_F \gg I_R) \quad \Rightarrow \quad \mathrm{I}_D = I_{Do} \exp\left(\frac{V_{GB}}{nU_T}\right) \exp\left(-\frac{V_{SB}}{U_T}\right)$
- Reverse Sat.
  $(V_{SB} - V_{DB}) \gg U_T, \quad (I_F \ll I_R) \quad \Rightarrow \quad \mathrm{I}_D = -I_{Do} \exp\left(\frac{V_{GB}}{nU_T}\right) \exp\left(-\frac{V_{DB}}{U_T}\right)$

(For the above equations, it has been assumed to be the channel length modulation (CLM) negligible.)

Where, the list of parameters is:

- W, L : width and length of the channel.
- C_{ox} : gate capacitance per unit area.
- $U_T = \frac{kT}{q} \qquad (26mV \,@\, 300K)$
- V_{To} : gate threshold voltage.
- n : slope factor (usually equals to $1.2 - 1.6$).
- μ : mobility.
- $V_p \cong \frac{V_{GB} - V_{To}}{n}$: pinch-off voltage.
- $\beta = \mu C_{ox} \frac{W}{L}$
- $I_s = 2n\beta U_T^2$: specific transistor current.
- $I_{Do} = I_s \exp\left(-\frac{V_{th,n}}{nU_T}\right)$

6.11.1 Proposed building blocks: nMOS log-domain integrator

In **[Toumazou, C. (2002)]** it is proposed a log-domain lossy integrator by using BJT. We use the same circuit architecture but with n-channel MOS transistors in weak inversion region as is depicted in **Figure 6.28** and we show that it is also possible to implement log-domain integrators.
To achieve the input-output transfer function it is assumed the following hypothesis: *i)* all transistor in weak inversion with the same value for the parameters U_T (i.e. same temperature) and n; *ii)* all MOS transistor in forward saturation regimen.

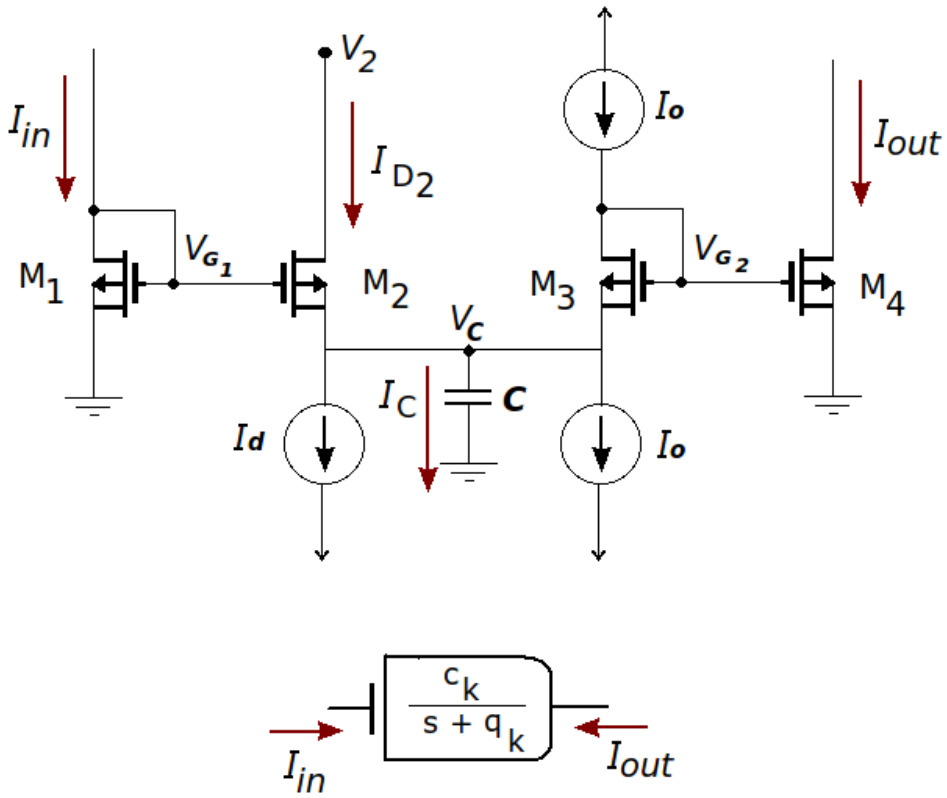


Figure 6.28: **Log-domain integrator implemented with n-channel MOS transistors and the associated symbol with sign convention. The current transfer implements (6.119).**

For M_1 the drain current is:

$$I_{in} = I_{Do1} \exp\left(\frac{V_{G1}}{nU_T}\right) \Rightarrow \exp\left(\frac{V_{G1}}{nU_T}\right) = \frac{I_{in}}{I_{Do1}} \tag{6.120}$$

For M_2 the drain current is:

$$I_d + C\dot{V}_c = I_{Do2} \exp\left(\frac{V_{G1} - V_c}{nU_T}\right) = I_{Do2} \exp\left(\frac{V_{G1}}{nU_T}\right) \exp\left(-\frac{V_c}{nU_T}\right) \tag{6.121}$$

Therefore, by using (6.120) we have that:

$$I_d + C\dot{V}_c = \frac{I_{Do2}}{I_{Do1}} I_{in} \exp\left(-\frac{V_c}{nU_T}\right) \tag{6.122}$$

Which can be stated as follow:

$$I_d \frac{I_{Do1}}{I_{Do2}} \exp\left(\frac{V_c}{nU_T}\right) + C\frac{I_{Do1}}{I_{Do2}} \dot{V}_c \exp\left(\frac{V_c}{nU_T}\right) = I_{in} \tag{6.123}$$

Which, once again it can be written as:

$$I_d \frac{I_{Do1}}{I_{Do2}} \exp\left(\frac{V_c}{nU_T}\right) + C\frac{I_{Do1}}{I_{Do2}} nU_T \frac{d}{dt}\left[\exp\left(\frac{V_c}{nU_T}\right)\right] = I_{in} \tag{6.124}$$

For M_4 the drain current is:

$$I_{out} = I_{Do4} \exp\left(\frac{V_{G2}}{nU_T}\right) \Rightarrow \exp\left(\frac{V_{G2}}{nU_T}\right) = \frac{I_{out}}{I_{Do4}} \tag{6.125}$$

For M_3 the drain current is:

$$I_o = I_{Do3} \exp\left(\frac{V_{G2} - V_c}{nU_T}\right) \tag{6.126}$$

Thus, by using (6.125)

$$I_o = \frac{I_{Do3}}{I_{Do4}} I_{out} \exp\left(-\frac{V_c}{nU_T}\right) \;\Rightarrow\; \exp\left(\frac{V_c}{nU_T}\right) = \frac{I_{Do3}}{I_{Do4} I_o} I_{out} \tag{6.127}$$

Thus, by substituting in (6.124), the following expression is reached:

$$\frac{I_d}{I_o} \frac{I_{Do1} I_{Do3}}{I_{Do2} I_{Do4}} I_{out} + \frac{I_{Do1} I_{Do3}}{I_{Do2} I_{Do4}} \frac{nCU_T}{I_o} \dot{I}_{out} = I_{in} \tag{6.128}$$

Or, better presented as:

$$\frac{I_d}{nCU_T} I_{out} + \dot{I}_{out} = \frac{I_o}{nCU_T} \frac{I_{Do2} I_{Do4}}{I_{Do1} I_{Do3}} I_{in} \tag{6.129}$$

Which implements:

$$H(s) = \frac{I_{out}(s)}{I_{in}(s)} = \frac{c_k}{s + q_k} \tag{6.130}$$

With:

- $q_k = \dfrac{I_d}{nCU_T}$
- $c_k = \dfrac{I_o}{nCU_T} \dfrac{I_{Do2} I_{Do4}}{I_{Do1} I_{Do3}}$

The parameters of the filter response can be tunned by varying the MOS transistor sizes and bias current.

In the next section we present another circuit that implements the required Laplace transference function (first-order low-pass), but, instead of using n-channel MOS transistors, we use p-channel MOS. The purpose of this will be well explained in the next subsection, but in advance we mention that coefficients $c_k < 0$ are required.

6.11.2 Proposed building blocks: pMOS log-domain integrator

The log-domain integrator is depicted in **Figure 6.29**
For M_1 the drain current is:

$$I_{in} = I_{Do1} \exp\left(\frac{V_{DD} - V_{G1}}{nU_T}\right) \;\Rightarrow\; \exp\left(-\frac{V_{G1}}{nU_T}\right) = \frac{I_{in}}{I_{Do1}} \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.131}$$

For M_2 the drain current is:

$$I_d - C\dot{V}_c = I_{Do2} \exp\left(\frac{V_c - V_{G1}}{nU_T}\right) = I_{Do2} \exp\left(-\frac{V_{G1}}{nU_T}\right) \exp\left(\frac{V_c}{nU_T}\right) \tag{6.132}$$

Therefore, by using (6.131) we have that:

$$I_d - C\dot{V}_c = \frac{I_{Do2}}{I_{Do1}} I_{in} \exp\left(\frac{V_c}{nU_T}\right) \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.133}$$

Which can be stated as follow:

$$I_d \exp\left(-\frac{V_c}{nU_T}\right) - C\dot{V}_c \exp\left(-\frac{V_c}{nU_T}\right) = I_{in} \frac{I_{Do2}}{I_{Do1}} \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.134}$$

Which, once again it can be written as:

$$I_d \exp\left(-\frac{V_c}{nU_T}\right) + CnU_T \frac{d}{dt}\left[\exp\left(-\frac{V_c}{nU_T}\right)\right] = I_{in} \frac{I_{Do2}}{I_{Do1}} \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.135}$$

For M_4 the drain current is:

$$I_{out} = I_{Do4} \exp\left(\frac{V_{DD} - V_{G2}}{nU_T}\right) \;\Rightarrow\; \exp\left(-\frac{V_{G2}}{nU_T}\right) = \frac{I_{out}}{I_{Do4}} \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.136}$$

For M_3 the drain current is:

$$I_o = I_{Do3} \exp\left(\frac{V_c - V_{G2}}{nU_T}\right) \tag{6.137}$$

Thus, by using (6.136)

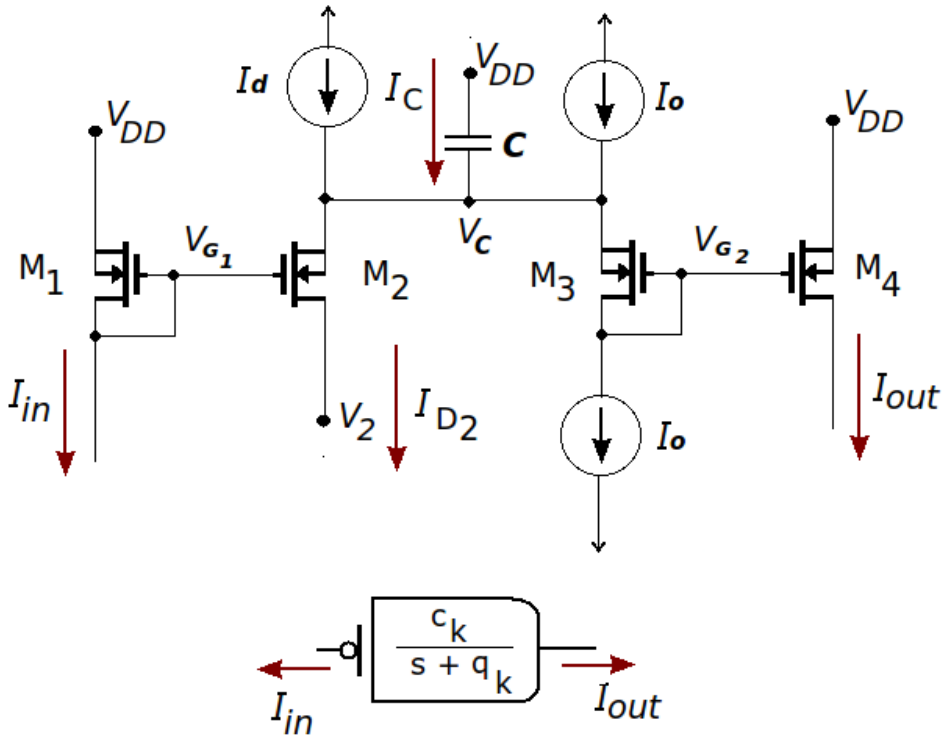


Figure 6.29: **Log-domain integrator implemented with p-channel MOS transistors and the associated symbol with sign convention. The current transfer implements (6.119).**

$$I_o = \frac{I_{Do3}}{I_{Do4}} I_{out} \exp\left(\frac{V_c}{nU_T}\right) \exp\left(-\frac{V_{DD}}{nU_T}\right) \Rightarrow \exp\left(-\frac{V_c}{nU_T}\right) = \frac{I_{Do3}}{I_{Do4} I_o} I_{out} \exp\left(-\frac{V_{DD}}{nU_T}\right) \tag{6.138}$$

Thus, by substituting in (6.135), the following expression is reached:

$$\frac{I_d}{nCU_T} I_{out} + \dot{I}_{out} = \frac{I_{Do2} I_{Do4}}{I_{Do1} I_{Do3}} \frac{I_o}{nCU_T} I_{in} \tag{6.139}$$

Which implements:

$$H(s) = \frac{I_{out}(s)}{I_{in}(s)} = \frac{c_k}{s + q_k} \tag{6.140}$$

With:

$$q_k = \frac{I_d}{nCU_T} \tag{6.141}$$

$$c_k = \frac{I_o}{nCU_T} \frac{I_{Do2} I_{Do4}}{I_{Do1} I_{Do3}} \tag{6.142}$$

The parameters' values for the filter response can be obtained with the correct design of the MOS transistor sizes and bias currents.

6.11.3 Basic configurations

Once building blocks are identified, it is possible to easily build circuits capable to implement the transfer function (6.114).
For instance, in **Figure 6.30** the following transfer is implemented, (which is a case of a case of (6.114) with $M = 2$):

$$H(s) = \frac{I_{out}(s)}{I_{in}(s)} = \frac{c_1}{s + q_1} + \frac{-c_2}{s + q_2} \tag{6.143}$$

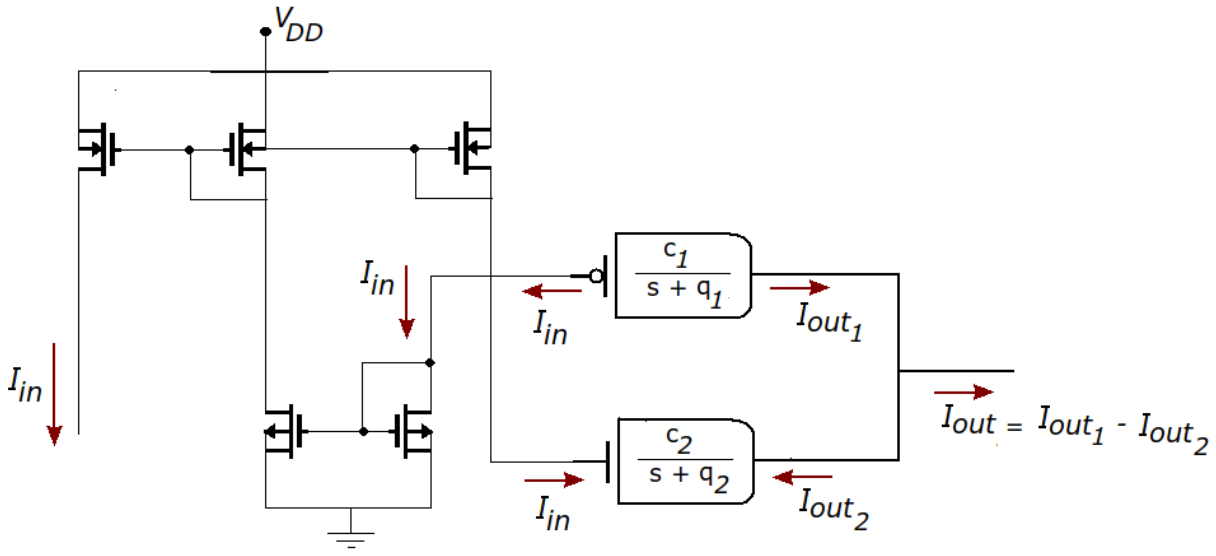


Figure 6.30: **Implementation of the impulse response expressed as (6.143)**

6.11.4 Drawbacks for the analog pulse shaping implementation

Despite the fact that the previous analysis is theoretically correct, the circuits suffer several disadvantages that compromises their real implementations.
We observe that:

- Considering the implementation, it is necessary to design an application specific integrated circuit (ASIC) for the given wireless sensor network application. This seriously compromise the system scalability and the philosophy of building wireless sensors nodes with general purpose hardware and low-cost components should be abandoned.
- All transistor must work in weak inversion with the same value for the parameters U_T (i.e. same temperature) and n, all MOS transistor must be in forward saturation regimen. Moreover, the quality of the manufacturing process (i.e. transistor mismatch) and the influence of the device parameter spread could be exponentially amplified producing large errors. It may lead to real implementations that behave differently with respect to the expected.
- In log-domain circuits the noise modeling and its effect on the output signal is an open issue in literature. It is not possible to establish a noise-figure for the log-domain circuits and this could be a problem for reduced input signals.
- Last but not least, equation (6.118) should be resolved encompassing the constraints (6.141) and (6.142) which lead to physically realizable circuits.

Then, as final remark, we don't recommend the use of analog pulse shaping. The Haar DWT detection capability by thresholding should be enhanced by implementing discret-time pulse shaping into the microntroller.

6.12 Final considerations

The main objective of this chapter was to investigate and to propose improvements for signal processing algorithms implemented into the wireless sensor node.
The problem of implementing local signal processing is relatively new in the WSN arena and recently it has started to be studied [**Greenstein, B. (2006)**].
The wavelets implementation is then motivated by the node's need of performing local signal processing techniques in order to reduce power consumption. Such power consumption reduction is achieved because only signal features of interest would be transmitted, with reduced radio transceiver activity.

The node is a very constrained hardware and only a very simple wavelet transform has sense because the implementation into the wireless sensors node is closely related to the power demanding.
Motivated by the previous, we selected the Haar wavelet and its reliability to perform signal transition classification (or detection) *by thresholding* has been studied.
The selection of such digital wavelet transform (DWT from herein) has been conditioned by; i) the reduced wireless sensor node computation capabilities and ii) the reduced node energy budget.
It has been observed through different simulations and analyses that there is an important lack in the detection capability and the *learning or training stage* is rather ad-hoc, i.e the level of decomposition selection, threshold value setting, and sampling frequency.
One possible proposed solution is to implement **pulse shaping** to compensate the poor Haar wavelet performance. However, it is not possible to assure correct detection by using analog pulse shaping due to several problems in the circuit implementation.
On the other hand, digital pulse shaping seems to be the suitable solution, but this technique is in its infancy and more intensive research should be addressed to develop the methodological aspects , i.e. level of decomposition selection, threshold value setting, sampling frequency and suitable impulse response.

Chapter 7

Conclusion and Final Remarks

This dissertation explores novel methodologies and systems to enable embedded wireless sensors nodes applications capable of achieving low power consumption with reduced voltage supply.
In particular, the performed research activity has focused attention on: *i*) study of the hardware and embedded software interaction and its effects on the battery current consumption and lifetime; *ii*) study and development of power-aware signal conditioning circuitry and signal processing techniques.
The proposed goal has been searched through a systematic approach which goes from network level to node level because the power consumption must be reduced locally as well as globally.

At the end of each chapter it has been offered critical evaluation about the faced research issue and presented inside the chapter.
Nevertheless, in this section we attempt to provide perspective to the research done, with final remarks and recommendations for addressing future work and improvements on this work.

As regards **Chapter 2**, the main objective of this chapter was to present an electronic system based on a dedicated PCB to visualize node current consumption usage and charge extracted from the battery during node operating states.
The second objective was to make reliable battery lifetime estimation for wireless sensors network applications. Both objectives have been successfully met.

Key open issues to address future research could be:

- Measurements on other embedded wireless sensors nodes in order to perform comparative analysis, specifically nodes which features different operating systems and (or) programming languages.
- More empirical validation about the battery lifetime estimation
- More research to validate the used battery model, specifically it is required better battery voltage profile characterization for different battery current loads.

Moreover, the obtained measurements have been used to propose a heuristic algorithm for searching wireless sensors deployments in **Chapter 3**. The algorithm predicts number of nodes, nodes distances and nodes output power level in order to give the deployment that reaches maximum battery lifetime compliant with user specifications.

Key open issues to address future research could be:

- Since the algorithm has no theoretical expression to estimate the convergence rate, its convergence capability has to be studied by performing statistical study of the different algorithm trials results. More simulations are required to give a converge rate estimation for large deployments.
- Moreover, more trials with different parameters values have to be implemented in order to test the convergence properties and robustness.
- An important issue which deserve special attention is the empirical validation because it must be implemented by using large number of nodes. It yields to problems of costs and testbed implementation under controlled environment to perform measurements.
  The empirical validation is very important because we revealed that the deployment behavior is

very different with respect to the expected and reported in literature when more real hypothesis are used.

As regards **Chapter 5**, we deal with the problem of the SNR evaluation in wireless sensor nodes. The motivation arises because the use of data acquisition adaptive-circuits (with the purpose of achieving node energy consumption reduction) can produce SNR variation.

Key open issues to address future research could be:

- The distortion effects as well as noise that actual AD converters produce should be added into the methodology
- It must be performed measurements with reduced numbers of bits and adaptive AD converters in order to better verify the SNR contour lines.

Finally, as a result and based on the discussion presented in **Chapter 2** and **Chapter 3** , it has been observed that to effectively reduce power consumption, embedded wireless sensors networks can not work as simple set of wireless data loggers.
Instead, the cooperative and distributed signal processing capabilities must be taken as key advantage and nodes should be able to locally process information.
The wavelet transform implementation into wireless sensors is then motivated by the node's need of performing local signal processing techniques in order to reduce power consumption. Such power consumption reduction is achieved because only signal features of interest would be transmitted, with reduced radio transceiver activity.
The discussion is presented in **Chapter 6**

The author recommendation is that, the approach related to wavelet implementation into embedded wireless sensors nodes, should be rejected.
As it has been explained in **Chapter 6** there are several reasons supporting this opinion. However, the idea of performing power-aware signal processing for signal features extraction is still opened thank to the new approach presented as very preliminary case study in **Chapter 8**.
The issue importance justify its discussion in another chapter.

Finally, we invite to interested readers to take more insight of the performed research activity by reading the appendixes.

Chapter 8

Future trend: Bio-inspired Computation on Wireless Sensors Nodes

Chapter Summary

In this final chapter we present a case study. The motivation arises due to the observed need of using different signal processing algorithms with respect to the discussion offered in Chapter 6.
The type of performed algorithm simulates the function of a spiking neural networks (SNNs). It is well-known that SNNs encode information and are capable of performing transients signal detection. In this framework, this test is aimed to prove the SNN signal identification capability by using reduced number of bits, i.e. 5 or less. The implementation of this biological inspired algorithm into the wireless sensor node has been investigated and interesting results arise from the power consumption reduction point of view.

8.1 Introduction

Consistent with the approach presented in the Chapter 6, this case study is aimed to implement algorithms for transient signal detection into the embedded wireless sensor, with reduced power consumption.
Motivated by the observation that digital Haar wavelet transform suffers lack in the transient signal detection capability, we attempt to test a biological inspired algorithm: SNN (spiking neural network) [**Valle, M. (Ed.) (2007)**], [**Bohte, S.M (2002)**]
In **Figure 8.1** it is briefly presented an intuitive idea (no mathematically formal) about how the SNN works. When the input signal accumulates enough energy and the potential is above a defined threshold, spikes appear at the SNN output.
With this briefly described behavior the SNN is capable of encoding information.

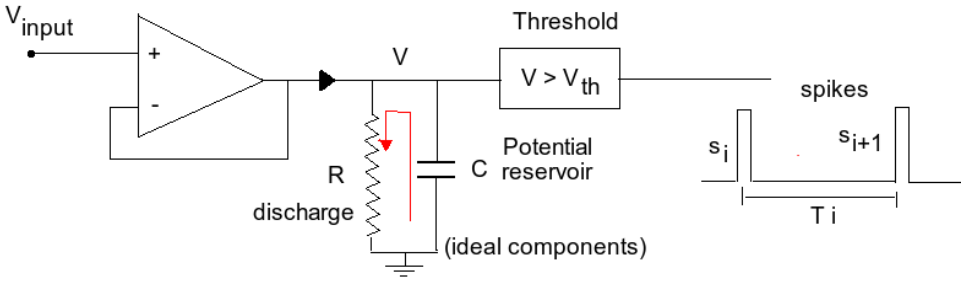


Figure 8.1: **Electrical model for a SSN basic unit.**

As case study, we use this algorithm in order to detect when an input signal has frequency components higher than $300Hz$. It is noteworthy to say that we obtained good results where the digital Haar wavelet failed. Moreover, in this preliminary study it is important to underline that:

- The algorithm uses only the 4 MSB (4 most significant bits). It means that for further application could be possible to use reduced number of bits for data representation, which is a desirable property to achieve reduced power consumption as we mentioned in Chapter 5.

- The number of 4 bits has been arbitrarily selected with the purpose of testing the algorithm robustness with reduced numbers of bits because by using ADC with this characteristic it is possible to build power-efficient wireless sensors nodes.
- Furthermore, only $1 - SNN$ has been implemented into the wireless node, with the purpose of characterizing this algorithm building block.
- Nowadays, embedded wireless sensors nodes don't feature AD converters with reduced number of bits, due to this, we approximate this scenario by using the 4 MSB of the sampled signal (it is shown in the implemented algorithm as and logical function AND with $F0_{hex}$.)

8.1.1 Learning process

We consider the following signal to be used in order to set the parameters:

$$y(t) = \begin{cases} C + C\,\mathtt{sin}\,(2\pi f_1 t) & t \in [0, t_1] \\ C + C\,\mathtt{sin}\,(2\pi f_2 t) & t \in [t_1, t_f] \end{cases} \tag{8.1}$$

Signal features: $f_1 = 100\,Hz$, $f_2 = 400\,Hz$, $V_{FS} = 3.3\,V$, $C = \dfrac{V_{FS}}{2}$, $t_1 = 0.5\,s$, $t_f = 1\,s$ and the sampling frequency $f_s = 1\,kHz$.
A program with the same structure that **Listening 8.1** has been written to perform the searching of the best values for the threshold V_{th} and the discharge rate R. It results $R = 200$ and $V_{th} = 70$. Whit this values the SSN identifies with persistent spikes when the input signal modifies its frequency component from $100Hz$ to $400Hz$.
We set the parameter iteratively by *trial and error*, rather ad-hoc after long time of testing (i.e. hours)

8.1.2 Experimental Validation

For the experimental validation it is used as input signal a linear sweep sine waveform which is expressed as follows:

$$s(t) = A\,\mathtt{sin}\,(\omega(t)\,t) \tag{8.2}$$

Where:

- A: amplitude
- $\omega(t)$: pulsation. For this case we use a lineal function of time $\omega(t) = 2\pi\,(f_i + R\,t)$
- f_i : initial frequency
- f_e : end frequency
- R : is the sweep rate with $R = \dfrac{f_e - f_i}{T_d}$
- T_d : time duration of the sine excitation, to which the end frequency is achieved.

For this study the following values are used:

- $A = 3.3\,V$: node voltage supply (it is also $V_{FS} = 3.3\,V$)
- $f_i = 200\,Hz$: initial frequency
- $f_e = 420\,Hz$: end frequency
- $T_d = 40\,ms$: time duration of the sine excitation
- $R = 5.5\,Hz/ms$: the sweep rate with

The input signal is generated by a waveform generator and the signal is sampled by the node.

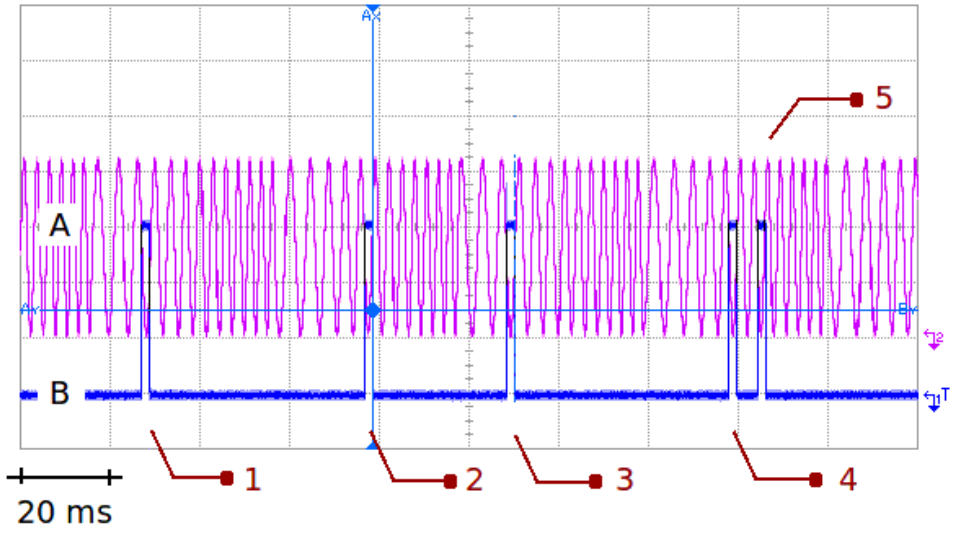


Figure 8.2: **Experimental result. It is shown the input signal (marked as *A*) and the software implemented $1-SNN$ response (marked as *B*). Approximated values are as follows: spike 1 appears around $260Hz$, spike 2 appears around $290Hz$, spike 3 appears around $260Hz$, and finally spike 5 appears around $310Hz$. We don't know at this moment the origin of the spike 4.**

8.1.3 Algorithm description

In the **Listening 8.1** is presented the algorithm implemented into the wireless sensor node, where the main features of our demonstrator are as follows: *i)* the SNN pulse is implemented by setting to high level the pin PORTB of the microntroller (marked as B in **Figure 8.2**), *ii)* the discharge rate R is considered to be constant ($R = 200$) and the potential threshold is $U_{th} = 70$; *iii)* as it has been discussed in **Chapter 4** a driver ad-hoc was implemented for higher sampling frequencies like this case where the sampling frequency us $f_s = 1kHz$.
Please observe that we use a constant discharge rate. It is equivalent to substitute the resistance with a current source that discharges the accumulated potential of the reservoir C in **Figure 8.1**

Listing 8.1: **Section of the NesC code that implement $1-SNN$ configured as described above.**

```
1
2  uint8_t potential=0;
3  norace uint8_t getdato;
4  uint8_t prescaler=0;
5
6  /* A/D interrupt handlers. Signals dataReady event with interrupts enabled */
7  AVR_ATOMIC_HANDLER(SIG_ADC) {
8
9       sei();
10      prescaler=prescaler+1;
11      if(prescaler==4){
12
13            getdato=ADCH;                          //new sample
14            getdato=getdato & 0xf0;
15
16            potential=potential+getdato;
17
18            if (potential<200){
19              potential=0;
20                              }
21
22            if (200< potential){
23              potential=potential-200;          // R =200
24                              }
25
```

```
26            if ( 70 <= potential ){
27              SET_BIT(PORTB,PORTB4);          //Marker (1) pulse up
28                                    }
29
30            if ( potential< 70 ){
31                CLR_BIT(PORTB,PORTB4);        //Marker (2) relax
32                                  }
33
34      prescaler=0;
35                  }
36
37  }
```

Results are promising and motivated by that we suggest to designers more research towards this direction. There are several open issues, however the most important issues is that the SNN can work with reduced number of bits. Clearly this type of algorithm outperforms the digital wavelet transforms studied in this work.

Appendix A

NesC Example 1

A.1 NesC Example 1

It is offered the whole code in NesC , as example, which implements the signal feature detection explained in the Chapter 7.

```
1  /**
2  * Private component or Driver to manage the ADC converter of the ATmega128
3  in order to get data in a 500 2-bytes words buffer
4  * @author: Leonardo Barboni
5  */
6
7  module ADCAcceDriverC {
8    provides {
9     async command void configure(uint8_t value1,uint8_t value2);
10    async command void start();
11    async command void stop();
12    async command void passpointer(uint16_t* p);
13           }
14                    }
15 implementation {
16
17 /* ADC Multiplexer Selection Register */
18 typedef struct{
19     uint8_t adps  : 3;  //ADC Prescaler Select Bits
20     uint8_t adie  : 1;  //ADC Interrupt Enable
21     uint8_t adif  : 1;  //ADC Interrupt Flag
22     uint8_t adfr  : 1;  //ADC Free Running Select
23     uint8_t adsc  : 1;  //ADC Start Conversion
24     uint8_t aden  : 1;  //ADC Enable
25 } ADCsra_t;
26
27 /* ADC Multiplexer Selection Register */
28 typedef struct{
29     uint8_t mux   : 5;  //Analog Channel and Gain Selection Bits
30     uint8_t adlar : 1;  //ADC Left Adjust Result
31     uint8_t refs  : 2;  //Reference Selection Bits
32 } ADCmux_t;
33
34 ADCmux_t mux_var;
35 ADCsra_t sra_var;
36 norace uint8_t getdato;
37
38 //variables and vectors
39 uint16_t* ss;
40 uint16_t pp=0;
41 uint8_t prescaler=0;
42
43
44 //variables para la SN
45 uint8_t aux=0;
46 uint8_t potential=0;
47 uint8_t aux1;
48
49 //Functions
50
51 void setting(uint8_t var1,uint8_t var2){
52      //PORTB4
53      SET_BIT(DDRB,4);            //Output pin configuration
54      CLR_BIT(PORTB,PORTB4);      //Output pin configuration   sbi(PORTE,6);
55      //SET_BIT(PORTB,PORTB4);    //Output pin configuration   sbi(PORTE,6);
56
```

```
57        //Accelerometer Power Enable PortC4;
58        SET_BIT(DDRC,4);              //Output pin configuration
59        SET_BIT(PORTC,PORTC4);        //Output pin configuration  sbi(PORTE,6);
60
61        //Old register values         //If I need for using later
62        atomic{
63        mux_var=*(ADCmux_t*)&ADMUX;
64        sra_var=*(ADCsra_t*)&ADCSRA;}
65
66        //ADCSRA=var1;      If I need to use the configuration from upper layers
67        //ADMUX=var2;
68
69        //Configure ADMUX
70        CLR_BIT(ADMUX, REFS0);        //AVCC with external capacitor at AREF pin
71        CLR_BIT(ADMUX, REFS1);
72
73        //CLR_BIT(ADMUX, ADLAR);      //ADC conversion result presentation
74        SET_BIT(ADMUX, ADLAR);        //ADC conversion result presentation
75
76        CLR_BIT(ADMUX, MUX4);         //MUX channel selection - THIS CASE
77                                      //ADC7 single ended input     MUX4..0=00111 (externo)
78        CLR_BIT(ADMUX, MUX3);         //MUX channel selection - THIS CASE
79                                      //ADC3 single ended input     MUX4..0=00011
80        SET_BIT(ADMUX, MUX2);
81        SET_BIT(ADMUX, MUX1);
82        SET_BIT(ADMUX, MUX0);
83
84        //Configure ADCSRA
85        SET_BIT(ADCSRA, ADPS2);
86        SET_BIT(ADCSRA, ADPS1);       //This configuration is  Division Factor =128
87        SET_BIT(ADCSRA, ADPS0);       //ADC prescaler Select bits
88  //return SUCCESS; function is defined as: error_t setting(uint8_t var1,uint8_t var2)
89  }
90
91  //Commands and Events
92
93  async command void configure(uint8_t value1,uint8_t value2){
94    setting(value1,value2);
95  }
96
97
98   async command void passpointer(uint16_t* p) {
99      ss=p;                                 // comparto las posiciones donde poner los datos
100                                             }
101
102
103 async command void start(){
104   SET_BIT(ADCSRA, ADEN);          //ADC Enable
105   SET_BIT(ADCSRA, ADFR);          //ADFR=1 Free Running mode selected
106   SET_BIT(ADCSRA, ADSC);          //ADSC=1 Start convertion.
107                                   //Writting zero to this bit has not effect
108   SET_BIT(ADCSRA, ADIE);          //Enable interruption
109                            }
110
111
112 async command void stop(){
113   CLR_BIT(ADCSRA, ADEN);          //ADC disable
114   CLR_BIT(ADCSRA, ADFR);          //Free Running disable
115   CLR_BIT(ADCSRA, ADIE);          //Disable interruption
116
117   //Accelerometer Power Disable;
118   CLR_BIT(PORTC,PORTC4);
119
120 }
121
122
123 /* A/D interrupt handlers. Signals dataReady event with interrupts enabled */
124   AVR_ATOMIC_HANDLER(SIG_ADC) {
125
126      sei();
127
128      prescaler=prescaler+1;
129
```

```
130     if(prescaler==4){
131
132           //SET_BIT(PORTB,PORTB4);
133           getdato=ADCH;
134           //*(ss+pp)=getdato & 0xf0;
135           getdato=getdato & 0xf0;
136           //*(ss+pp)=ADC;
137           //*(ss+pp)=ADCH;
138
139           //en esta parte implementamos the SN
140
141           potential=potential+getdato;
142
143           if (potential<200){
144             potential=0;
145                              }
146
147           if (200< potential){
148             potential=potential-200;
149                              }
150
151           //potential=potential-5;
152
153           //if (potential<100){
154           // potential=0;
155           //                 }
156
157           //*(ss+pp)=potential;          //relevamos el potential a ver como evoluciona
158
159           if ( 70 <= potential ){
160             SET_BIT(PORTB,PORTB4);
161                                  }      // Output pin configuration  sbi(PORTE,6);
162
163
164           if ( potential< 70 ){
165             CLR_BIT(PORTB,PORTB4);
166                                  }      // Output pin configuration  sbi(PORTE,6);
167
168     //esta parte de aca abajo es para la parte de alta velocidad
169     prescaler=0;
170
171     //CLR_BIT(PORTB,PORTB4);
172
173     //pp=pp+1;
174
175     //if (pp==1000){
176               //Stop all
177     //          CLR_BIT(ADCSRA, ADEN);          // ADC disable
178     //          CLR_BIT(ADCSRA, ADFR);          // Free Running disable
179     //          CLR_BIT(ADCSRA, ADIE);          // Disable interruption
180     //               }
181                 } // prescaler end
182
183   }//end de la rutina de interrupcion del ADC
184
185 }  //end ADCdriverC
186 }
```

Appendix B

NesC Example 2

B.1 NesC Example 2

It is offered an example of the main components present in a NesC code. This program has been used in order to perform some experimental validation cases presented in Chapter 5. It is composed as is described in the configuration file.

Configuration File: *MyAppC.nc*

```
1
2  #include "RadioSenseToLeds.h"
3  #include "AM.h"
4  #include "TestSerial.h"
5  #include "StorageVolumes.h"
6
7  configuration MyAppC {}
8  implementation {
9    components MainC, MyAppModuleC, LedsC;
10
11   //AD converter module
12   components ADCAcceDriverC;
13
14   //Timer
15   components new TimerMilliC();
16
17   //External Flash memory Atd45
18   components new BlockStorageC(VOLUME_BLOCKTEST);
19   //Communication to PC
20   components TestSerialC as App;
21   components SerialActiveMessageC as AM;
22
23   //Main and LedsC
24   MyAppModuleC.Boot -> MainC.Boot;
25   MyAppModuleC.Leds -> LedsC;
26
27   //Timer
28   MyAppModuleC.MilliTimer -> TimerMilliC;
29
30   //AD converter
31   MyAppModuleC.configure-> ADCAcceDriverC.configure;
32   MyAppModuleC.start     -> ADCAcceDriverC.start;
33   MyAppModuleC.stop      -> ADCAcceDriverC.stop;cases of the
34   MyAppModuleC.passpointer ->  ADCAcceDriverC.passpointer;
35
36   // Wired: Communication to PC
37   MyAppModuleC.sendtoPC -> App.sendtoPC;
38   App.Boot -> MainC.Boot;
39   App.Control -> AM;
40   //App.Receive -> AM.Receive[AM_TEST_SERIAL_MSG];
41   App.AMSend -> AM.AMSend[AM_TEST_SERIAL_MSG];
42   App.Packet -> AM;
43
44   //Wired: External Flash memory Atd45
45   MyAppModuleC.BlockRead -> BlockStorageC.BlockRead;
46   MyAppModuleC.BlockWrite -> BlockStorageC.BlockWrite;
47 }
```

File: *ADCAcceDriverC.nc*

```
1
2  module ADCAcceDriverC {
3    provides {
4     async command void configure(uint8_t value1,uint8_t value2);
5     async command void start();
6     async command void stop();
7     async command void passpointer(uint16_t* p);
8
9   }
10 }
11
12 implementation {
13
14 /* ADC Multiplexer Selection Register */
15 typedef struct
16 {
17     uint8_t adps  : 3;  //ADC Prescaler Select Bits
18     uint8_t adie  : 1;  //ADC Interrupt Enable
19     uint8_t adif  : 1;  //ADC Interrupt Flag
20     uint8_t adfr  : 1;  //ADC Free Running Select
21     uint8_t adsc  : 1;  //ADC Start Conversion
22     uint8_t aden  : 1;  //ADC Enable
23 } ADCsra_t;
24
25 /* ADC Multiplexer Selection Register */
26 typedef struct
27 {
28     uint8_t mux   : 5;  //Analog Channel and Gain Selection Bits
29     uint8_t adlar : 1;  //ADC Left Adjust Result
30     uint8_t refs  : 2;  //Reference Selection Bits
31 } ADCmux_t;
32
33 ADCmux_t mux_var;
34 ADCsra_t sra_var;
35 norace uint16_t getdato;
36
37 //variables and vectors
38 uint16_t* ss;
39 uint16_t pp=0;
40
41 //Functions
42
43 void setting(uint8_t var1,uint8_t var2){
44
45      //Accelerometer Power Enable PortC4;
46      SET_BIT(DDRC,4);            // Output pin configuration
47      SET_BIT(PORTC,PORTC4);      // Output pin configuration   sbi(PORTE,6);
48
49      //Old register values       // If I need for using later
50      atomic{
51      mux_var=*(ADCmux_t*)&ADMUX;
52      sra_var=*(ADCsra_t*)&ADCSRA;}
53
54      //ADCSRA=var1;     If I need to use the configuration from upper layers
55      //ADMUX=var2;
56      //Configure ADMUX
57      CLR_BIT(ADMUX, REFS0);      //  AVCC with external capacitor at AREF pin
58      CLR_BIT(ADMUX, REFS1);
59
60      CLR_BIT(ADMUX, ADLAR);      //  ADC conversion result presentation
61
62      CLR_BIT(ADMUX, MUX4);       //  MUX channel selection - THIS CASE
63                                  //  ADC7 single ended input     MUX4..0=00111 (externo)
64      CLR_BIT(ADMUX, MUX3);       //  MUX channel selection - THIS CASE
65                                  // ADC3 single ended input     MUX4..0=00011
66      SET_BIT(ADMUX, MUX2);
67      SET_BIT(ADMUX, MUX1);
68      SET_BIT(ADMUX, MUX0);
69
70      //Configure ADCSRA
71      SET_BIT(ADCSRA, ADPS2);
72      SET_BIT(ADCSRA, ADPS1);     // This configuration is  Division Factor =128
```

```
73      SET_BIT(ADCSRA, ADPS0);     // ADC prescaler Select bits
74  //return SUCCESS; function defined as:   error_t setting(uint8_t var1,uint8_t var2)
75  }
76
77  //Commands and Events
78
79  async command void configure(uint8_t value1,uint8_t value2){
80    setting(value1,value2);
81  }
82
83   async command void passpointer(uint16_t* p) {
84      ss=p;                                  // comparto las posiciones donde poner los datos
85                                                  }
86  async command void start(){
87    SET_BIT(ADCSRA, ADEN);          // ADC Enable
88    SET_BIT(ADCSRA, ADFR);          // ADFR=1 Free Running mode selected
89    SET_BIT(ADCSRA, ADSC);          // ADSC=1 Start convertion.
90                                    // Writting zero to this bit has not effect
91    SET_BIT(ADCSRA, ADIE);          // Enable interruption
92                              }
93
94  async command void stop(){
95    CLR_BIT(ADCSRA, ADEN);          // ADC disable
96    CLR_BIT(ADCSRA, ADFR);          // Free Running disable
97    CLR_BIT(ADCSRA, ADIE);          // Disable interruption
98
99    //Accelerometer Power Disable;
100   CLR_BIT(PORTC,PORTC4);
101 }
102
103 /* A/D interrupt handlers. Signals dataReady event with interrupts enabled */
104   AVR_ATOMIC_HANDLER(SIG_ADC) {
105
106      sei();
107      //getdato=0xabcd;         // dummy test
108      *(ss+pp)=ADC;
109
110      pp=pp+1;
111
112      if (pp==1000){
113               //Stop all
114               CLR_BIT(ADCSRA, ADEN);          // ADC disable
115               CLR_BIT(ADCSRA, ADFR);          // Free Running disable
116               CLR_BIT(ADCSRA, ADIE);          // Disable interruption
117                   }
118   }
119 }  //end ADCdriverC
```

File: *MyAppModuleC.nc*

```
1
2   #include "RadioSenseToLeds.h"
3   #include "StorageVolumes.h"
4
5   module MyAppModuleC {
6
7   uses {
8       interface Leds;
9       interface Boot;
10      interface Timer<TMilli> as MilliTimer;
11
12      // ADC control
13
14      async command void configure(uint8_t value1,uint8_t value2);
15      async command void start();
16      async command void stop();
17
18      // External Flash
19
20      interface BlockRead;
21      interface BlockWrite;
```

```
22
23      // UART packet Communication
24
25      async command void sendtoPC(uint16_t var_aux);
26      async command void passpointer(uint16_t* p);
27        }
28  }
29  implementation {
30
31    uint16_t Vth=0x0200;
32    uint8_t dummy1=0;
33    uint8_t dummy2=0;
34
35    //user configuration
36    bool dobackup=FALSE;          // TRUE for writing ,
37                                  // FALSE in order to read (don't erase data),
38                                  // automatically it is actived the read procedure
39    bool readbackup=FALSE;        // TRUE for reading , FALSE in order not to read
40                                  // readbackup=!dobackup;  (in boot)
41
42    bool writedone=FALSE;
43    bool readdone=FALSE;
44    bool syncdone=FALSE;
45    bool clear=FALSE;
46    // End Memory configuration
47
48    //For action  control
49    uint8_t flag=0xff;
50    uint8_t input;
51    bool starting;
52
53    //Vectors and parameters
54    uint16_t  samples[1000];
55    uint16_t* psample=&samples[0];
56
57    uint16_t index=0;
58    uint16_t pointer=0;
59    uint16_t* datap;                // pointer to the sampled data
60    uint16_t  measure;
61    uint8_t action=0;               // for control
62
63    //for FLASH memory usage (backup)
64    uint32_t length=2*1000;                 // (size) in bytes of samples
65    uint32_t address=0x0000;
66
67    task void TiltProcess(){
68     measure=*datap;
69     if(measure> Vth) {
70           call Leds.led0On();
71           SET_BIT(PORTC,PORTC2);           // Sound ON : Output power enable
72                         }
73     else {call Leds.led0Off();
74           CLR_BIT(PORTC,PORTC2);           // Sound OFF :Output power disable}
75           }
76                           }
77
78    task void memory(){
79      if(dobackup==TRUE && clear==TRUE ){
80            call BlockWrite.write(address, samples, length);
81            dobackup=FALSE;
82                         }
83      if(readbackup==TRUE){
84           call BlockRead.read(address, samples, length);
85           readbackup=FALSE;}
86                          }     //end Task
87
88    task void startup(){
89
90        call Leds.led0On();
91        if (flag==0x11) {
92            // Active for sampling and storage
93            dobackup=TRUE;
94            readbackup=FALSE;
```

```
95          call BlockWrite.erase();              // it must be done if we have to write
96
97          call configure(dummy1,dummy2);        // configure de ADC
98          action=0;                             // action 0: sampling and storage
99          flag=0x00;                            // flag null, if not, it is detected in
100                                                // the next if start ADC
101         call start();
102
103             index=1000;
104             call MilliTimer.startPeriodic(2000);
105                     } // end if flag==0x11
106
107       if (flag==0x22) {
108          // Active for memory lecture only, sampling is not implemented
109          call stop();
110          dobackup=FALSE;
111          readbackup=TRUE;
112          post memory();
113          action=2;
114          call MilliTimer.startPeriodic(150);
115                       } // end if flag==0x22
116     }
117
118   //Boot sequence
119   event void Boot.booted() {
120
121        starting=TRUE;
122        call passpointer(psample);
123        call MilliTimer.startPeriodic(5000);      // I wait 2 secods to start action
124                       }
125
126   event void MilliTimer.fired() {
127
128     if (starting==TRUE) {
129         //USER CONFIGURATION
130         flag=0x22;                          // active to send to PC
131         //flag=0x11;                        // active for sampling and storage
132         starting=FALSE;
133         action=4;                  // do nothing
134         post startup();
135                     }
136
137     // CASE 0: I am sampling and buffering
138     // remember the time in 6ms or whatever
139     if (action==0) {
140
141        if (index==1000) {
142            call MilliTimer.stop();                       // stop timer
143            call stop();                                  // stop ADC
144            action=5;                                     // do nothing
145            post memory();
146                        }
147
148        if (index<1000) {
149              //samples[index]=*datap;            // datap es la posicion compartida con el
150                                                  //ADC si es sampling de baja velocidad
151              index=index+1;
152                        }
153                } //End CASE 0
154
155     //CASE 1:  dummy test
156     if (action==1) {
157         post TiltProcess();
158      } //End CASE 1
159
160     // CASE 2:  send to PC
161     if (action==2 && readdone==TRUE ) {
162
163       if (pointer==1000) {
164          call MilliTimer.stop();
165          call Leds.led1On();                      // end of transmission -Led Green
166                        }
167
```

```
168         if (pointer<1000) {
169            call sendtoPC( samples[pointer] );
170            pointer=pointer+1;
171                        }
172                          } //End CASE 2
173  }
174
175   // Interfaces that manage the external memory
176
177   // From BlockRead
178   event void BlockRead.readDone
179   (storage_addr_t addr, void* buf, storage_len_t len, error_t error){
180         readdone=TRUE;
181                                                                    }
182   event void BlockRead.computeCrcDone
183   (storage_addr_t addr, storage_len_t len,uint16_t crc, error_t error){}
184
185   // From BlockWrite
186   event void BlockWrite.writeDone
187   (storage_addr_t addr, void* buf, storage_len_t len,error_t error){
188         writedone=TRUE;
189            call Leds.led2On();
190         if(error == SUCCESS ) {
191           call BlockWrite.sync();}
192
193 }
194   event void BlockWrite.eraseDone(error_t error){
195          clear=TRUE;
196          if(error != SUCCESS ) {
197            clear = FALSE;
198            call BlockWrite.erase();}
199   }
200   event void BlockWrite.syncDone(error_t error){
201          syncdone=TRUE;
202          //call Leds.led0Toggle();                    //Led red
203          if(error != SUCCESS ) {
204            syncdone= FALSE;}
205            }
206 }
```

File: *TestSerialC.nc*

```
1
2  #include "Timer.h"
3  #include "TestSerial.h"
4
5  module TestSerialC {
6
7    provides{
8      async command void sendtoPC(uint16_t var_aux);
9    }
10   uses {
11     interface SplitControl as Control;
12     interface Boot;
13     interface AMSend;
14     interface Packet;
15   }
16 }
17 implementation {
18
19   message_t packet;
20   uint16_t counter = 0;
21
22   event void Boot.booted() {
23     call Control.start();
24   }
25   async command void sendtoPC(uint16_t var_aux){
26     counter++;
27     if(1==1){
28       test_serial_msg_t* rcm = (test_serial_msg_t*)call Packet.getPayload(&packet, NULL);
29       counter=var_aux;
```

```
30        rcm->dataPC[0] = counter;
31        call AMSend.send(AM_BROADCAST_ADDR, &packet, sizeof(test_serial_msg_t));
32              }
33    }
34    event void AMSend.sendDone(message_t* bufPtr, error_t error) {}
35    event void Control.startDone(error_t err) {}
36    event void Control.stopDone(error_t err) {}
37  }
```

File: *TestSerial.h*

```
1
2   #ifndef TEST_SERIAL_H
3   #define TEST_SERIAL_H
4
5   typedef nx_struct test_serial_msg {
6     nx_uint16_t dataPC[7];
7   } test_serial_msg_t;
8
9   enum {
10    AM_TEST_SERIAL_MSG = 9,
11  };
12
13  #endif
```

File: *Storage.h*

```
1
2   * Copyright (c) 2002-2005 Intel Corporation
3    * All rights reserved.
4    *
5    * This file is distributed under the terms in the attached INTEL-LICENSE
6    * file. If you do not find these files, copies can be found by writing to
7    * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
8    * 94704.  Attention:  Intel License Inquiry.
9    */
10
11  /*
12   * @author: Jonathan Hui <jwhui@cs.berkeley.edu>
13   * @author: David Gay <david.e.gay@intel.com>
14   * @version $Revision: 1.5 $ $Date: 2008/06/11 00:46:28 $
15   */
16
17  #ifndef STORAGE_H
18  #define STORAGE_H
19
20  typedef uint8_t volume_id_t;
21  typedef uint32_t storage_addr_t;
22  typedef uint32_t storage_len_t;
23  typedef uint32_t storage_cookie_t;
24
25  enum {
26    SEEK_BEGINNING = 0
27  };
28
29  #include "Storage_chip.h"
30
31  #endif
```

Appendix C

Prototype board used in Chapter 2

C.1 PCB layout

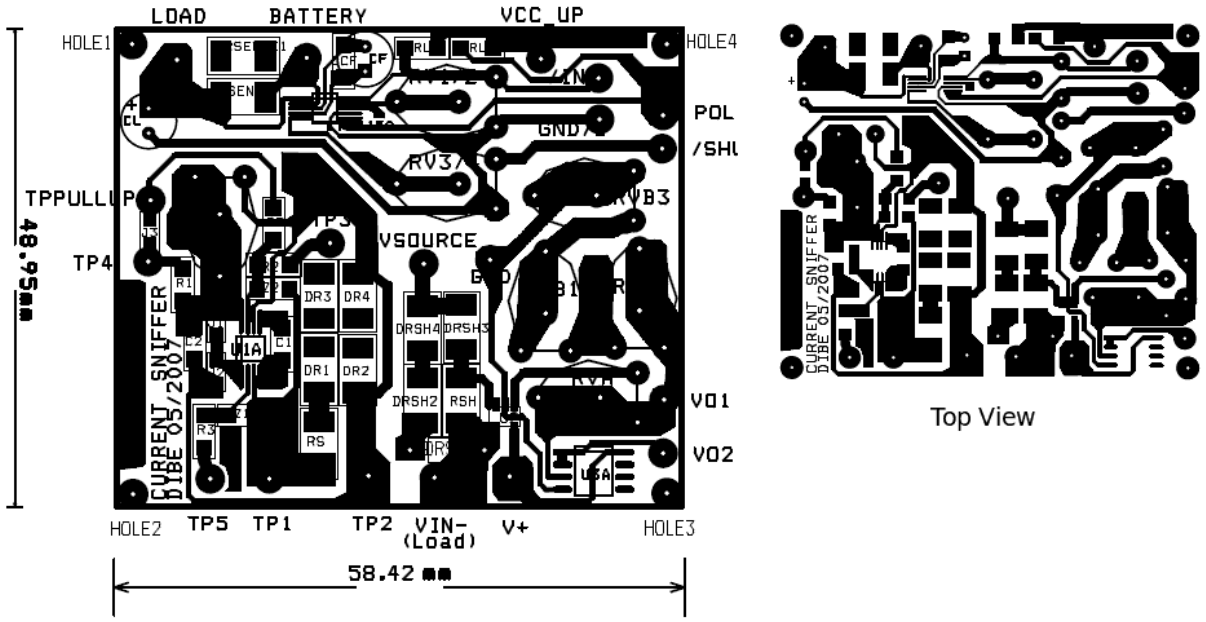


Figure C.1: **Printed circuit board layout.**

C.2 Schematics

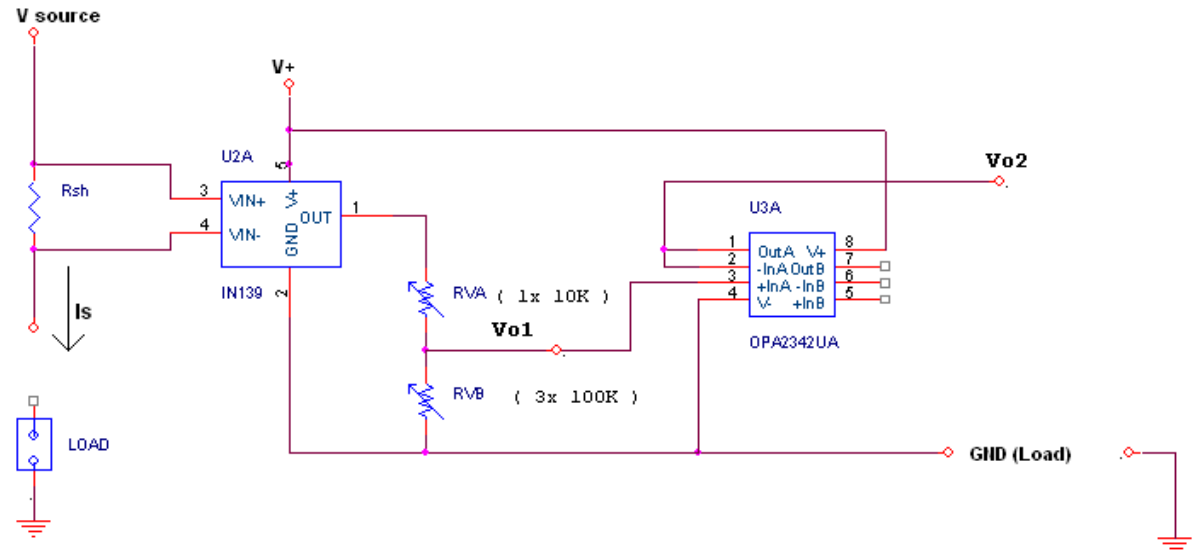


Figure C.2: **Schematic for the High Side Current Shunt Monitor INA139.**

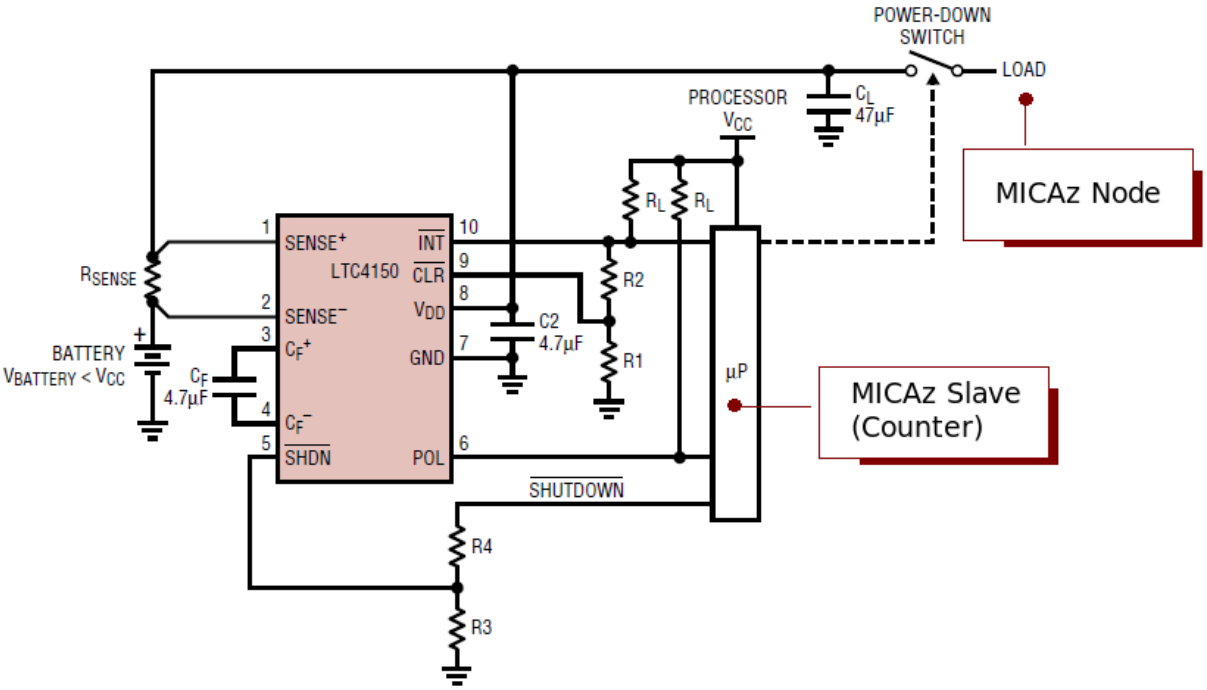


Figure C.3: **Schematic for the Coulomb Counter LTC4150. The implemented circuit is suggested in the datasheet (Figure 6 from [LTC4150 Datasheet], adapted).**

Appendix D

Battery Voltage Model

D.1 Background

Nowadays, embedded wireless microsystem are battery powered. This appendix offers an approach in order to analytically relate the battery voltage with battery current consumption.
Chemical reactions inside the battery are well known and reported in books of electrochemistry. Nevertheless, from the embedded wireless designer point of view, the battery behavior must be described in terms of voltage and battery current consumption. Such issue is still open and an unified and accurate methodology has not arisen yet among designers [**Cloth, L. (2007)**].
The work aim is to attempt to clearly provide a closed model to relate two important electrical parameters: *battery voltage and the battery current consumption.*

The battery converts chemical energy to electrical energy. In **Figure D.1**, it is briefly shown the battery architecture. The following description can be obtained from every basic books of electrochemistry, we only provide the generalization for the mathematical approach.
A battery is built with two electrodes marked with (2) and (3) in **Figure D.1** and separate by an electrolyte (4) and a separator (1). Each electrode and electrolyte conform a battery cell. One battery has 2 cells.
The electrode positive is defined as **cathode** (3) and the electrode negative as **anode** (2). The separator (1) is a porous material that enables electro active species (ions) to flow between cells but prevents the electrolytes mixing.
The cathode is composed by an electronic conductor built with a moist paste that contain a chemical specie R_1 (usually a metal). Similarly, the anode has the chemical specie S_2.

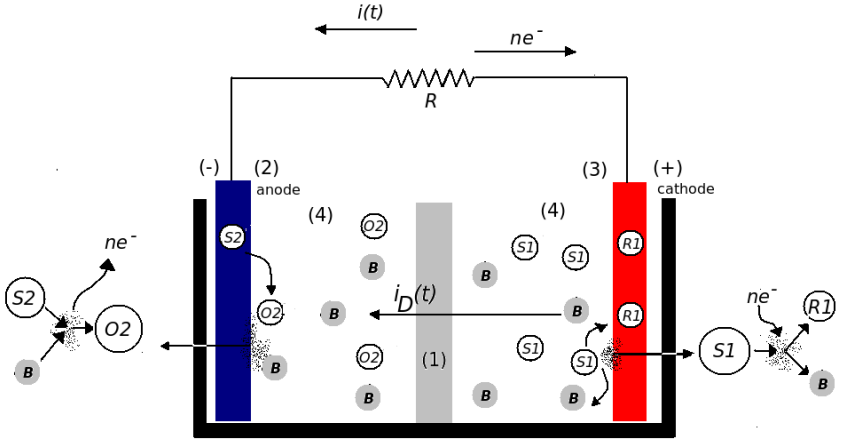


Figure D.1: **Battery model from the electro active chemical species.**

For better understanding as an example, in a $Zn-Cu$ battery, such species are Zn (for S_2) and Cu (for R_1).

The electrolyte is a solution that contains electro active species O_2, that would have positive charge (or cations in the cathode side) and electro active species S_1 that could have negative charge (or anions in the anode side).
The electro active specie and its associated charge depends on the battery technology and below we offer examples. Moreover, the electrolyte has another electro active specie B. The specie B can be moved

through the electrolyte according to Fick's diffusion laws.
Once the battery is connected to the circuit, it consumes current and it means that a flow of electrons is established between electrodes. It originates electrochemical reactions in the interface electrode-electrolyte. In the cathode, the chemical specie S_1 is reduced making as residue the electrical neutral reduced specie. R_1 Similarly in the anode, the specie S_2 suffers oxidation producing as residue the specie O_2.
It can be expressed as follows:

At anode (oxidation): $S_{2\ (solid)} + B_{(solution)} \longrightarrow O_{2\ (solution)} + ne^-$
At the cathode (reduction): $S_{1\ (solution)} + ne^- \longrightarrow R_{1\ (solid)} + B_{(solution)}$

In the cathode, the S_1 species are broken and the new reduced specie is caught by the electrode. Moreover, there is a residual specie B which migrates to the opposite cell (through the separator), where it will be recombined with the anode species during the oxidation process.
The S_2 species are expelled from the anode because of the oxidation process. Once the oxide specie is moving in the electrolytic, it creates the species O_2 by coupling to species B. Then, there is a unbalance of ionic charge that originates a diffusion process of the electro active species B through the electrolytic (from cathode to anode).
Below, we analyze the abstraction of the model, but first of all we offer some classic examples present in every book of electrochemical.

Battery example 1: $Zn - Cu$ Battery
In the case we have that: $S_2 = Zn$, $O_2 = ZnS0_4$, $B = 2SO_4^-$, $S_1 = CuS0_4$ and $R_1 = Cu$

Battery example 2: Alkaline Battery
For the anode: $S_2 = Zn$, $O_2 = Zn(OH)_2$ and $B = 2OH^-$. When the oxide S_2 and B are combined, the new specie is $O_2 = Zn(OH)_2$.
For the cathode, we have that $S_1 = MnO_2 + H_20$ (it is not an unique specie, the reaction requires both), and $R_1 = Mn_2O_3$, $B = 2OH^-$.
Similarly with respect to the anode case, H_20 molecules are presented in solution and also participates in the reaction.
Reactions are:
anode: $Zn(s)2OH(aq) \rightarrow Zn(OH)_2(s) + 2e^-$
cathode: $2MnO_2(s) + 2H_20 + 2e^- \rightarrow Mn_2O_3(s) + H_20 + 2OH(aq)^-$

The cathode is composed by Mn_2O_3. Near the cathode there is a paste of MnO_2 which is the battery combustible. The electrolyte is composed by H_20 and by a solution of $2OH^-$.

Battery example 3: lithium battery, reactions are as follows [**Gao, L. (2007)**]:
cathode : $Li_{1-x}CoO_2 + xLi^+ + xe^- \rightarrow LiCoO_2$
anode: $CLi_x \rightarrow C + xLi^+ + xe^-$.

The cathode is made with carbon and the lithium ions are the electro active specie which migrates from the anode cell to the cathode. Please, notice how in this battery type, the ion Li^+ is created at the anode, instead of the cathode, and then it has to migrate to the cathode in order to be used during the reduction reaction.

From herein we take the alkaline battery as case study.

D.2 Novel expression for the alkaline battery $V - I$ characteristic

The electrochemical reaction starts to reduce the concentration of species in the cell, and finally, once the concentration at one of the electrode surface drops below a threshold for which the reaction can not be produced, the battery is considered depleted.
The model presented below is based on the analytical model Kinetic Battery Model of Manwell and McGowan [**KiBaM**], however this work enhances the state of the art because we add both cells (anode-cathode for more realistic battery modeling) and we relate the current consumption (not the consumed

charge) with the battery voltage.

The alkaline battery voltage and species concentrations are correlated by the well-known Nernst's Equation as follows:

$$V_B = V_{Bo} - \frac{c}{n} \ln \left(\frac{\left[OH^-_{cathode}\right]^2}{\left[OH^-_{anode}\right]^2} \right) \tag{D.1}$$

Where:

- V_B: battery voltage
- V_{Bo}: battery voltage at time $t = 0$. For the alkaline battery the fresh voltage is $V_{Bo} = 1.6\,V$
- $c = \dfrac{RT}{F}$, at $25°C$ $c = RT/F = 25.679mV$
- $R = 8.314\ JK^{-1}mol^{-1}$: universal gas constant:
- T: temperatura (K)
- n: number of electrons transferred in the cell reaction (for the alkaline battery $n = 2$)
- $F = 9.649\ Cmol^{-1}$: Faraday constant
- $\left[OH^-_{cathode}\right]$, $\left[OH^-_{anode}\right]$: concentration of the ions in the cells of the alkaline battery.

Therefore, we have to model the ions' concentration behavior. At first glance it is easy to see that $\left[OH^-_{cathode}\right]$ increases and $\left[OH^-_{anode}\right]$ decreases. We define: $Q_c(t) = \left[OH^-_{cathode}\right]$ and $Q_a(t) = \left[OH^-_{anode}\right]$

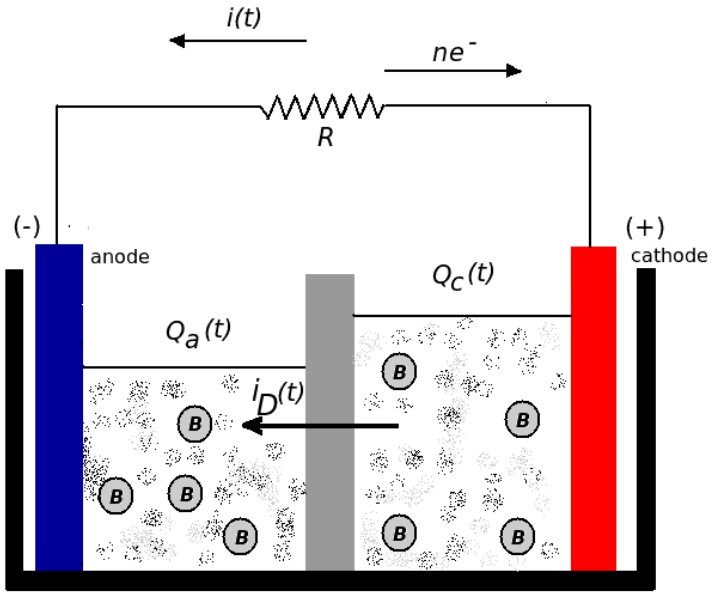


Figure D.2: **Battery model with the electro active chemical species. Model based on [KiBaM]**

Then, with reference to **Figure D.2** and expression (D.1) for the battery voltage the proposed model is expressed as follows:

$$\frac{dQ_c(t)}{dt} = \alpha i(t) - \lambda i_D(t) \tag{D.2}$$

$$\frac{dQ_a(t)}{dt} = -\alpha i(t) + \lambda i_D(t) \tag{D.3}$$

The diffusion current is assumed to be lineal:

$$i_D(t) = \mu \left(Q_c(t) - Q_a(t) \right) \tag{D.4}$$

Where μ, α and λ are unknown physical constants which depend on the battery geometry and chemical species. They have to be determined by battery characterization.
We express equations in Laplace domain:

$$sQ_c - Q_c(t_0) = \alpha i(s) - \lambda\mu\left(Q_c(s) - Q_a(s)\right) \tag{D.5}$$

$$sQ_a - Q_a(t_0) = -\alpha i(s) + \lambda\mu\left(Q_c(s) - Q_a(s)\right) \tag{D.6}$$

In order to simplify notation, we define: $Q_{a0} = Q_a(t_0)$, $Q_{c0} = Q_c(t_0)$, $\beta = \lambda\mu$.

Thus:

$$(s+\beta)\,Q_c(s) = Q_{c0} + \alpha i(s) + \beta Q_a(s) \tag{D.7}$$

$$(s+\beta)\,Q_a(s) = Q_{a0} - \alpha i(s) + \beta Q_c(s) \tag{D.8}$$

Then:

$$Q_a(s) = \frac{Q_{a0} - \alpha i(s) + \beta Q_c(s)}{s+\beta} \tag{D.9}$$

We calculate $Q_c(s)$

$$(s+\beta)Q_c(s) = Q_{c0} + \alpha i(s) + \beta\frac{Q_{a0} - \alpha i(s) + \beta Q_c(s)}{s+\beta} \tag{D.10}$$

$$\left(s + \beta - \frac{\beta^2}{s+\beta}\right)Q_c(s) = Q_{c0} + \alpha i(s) + \beta\frac{Q_{a0} - \alpha i(s)}{s+\beta} \tag{D.11}$$

$$\frac{s^2 + 2\beta s}{s+\beta}Q_c(s) = \frac{Q_{c0}(s+\beta) + \alpha i(s)(s+\beta) + \beta Q_{a0} - \alpha\beta i(s)}{s+\beta} \tag{D.12}$$

$$s(s+2\beta)Q_c(s) = Q_{c0}(s+\beta) + \alpha s i(s) + \beta Q_{a0} \tag{D.13}$$

Finally:

$$Q_c(s) = \frac{Q_{c0}}{s+2\beta} + \frac{\alpha i(s)}{s+2\beta} + \beta\frac{Q_{a0} + Q_{c0}}{s(s+2\beta)} \tag{D.14}$$

We come back to time domain for each term, therefore:

$$L^{-1}\left\{\frac{Q_{c0}}{s+2\beta}\right\} = Q_{c0}e^{-2\beta t} \qquad t \geq 0 \tag{D.15}$$

$$L^{-1}\left\{\frac{\beta(Q_{c0}+Q_{a0})}{s(s+2\beta)}\right\} = \beta(Q_{c0}+Q_{a0})\int_0^t e^{-2\beta\tau}d\tau = \frac{1}{2}(Q_{c0}+Q_{a0})(1-e^{-2\beta t}) \qquad t \geq 0 \tag{D.16}$$

Finally we analyze the term with the current. To do this, we assume a staircase current consumption profile as **Figure D.3 shows**, it is expressed as:

$$i(t) = \sum_{K=1}^{N} I_K(u(t-t_{K-1}) - u(t-t_K)) \tag{D.17}$$

The number N and time t holds:

$$t \leq t_n = \sum_{K=1}^{N}\Delta t - K \tag{D.18}$$

This means that within the time interval $[0, t_N]$ there are N intervals of the form $[t_{K-1}, t_K]$, $K = 0, 1, 2, \cdots$.
The delay unit step s-transform is:

$$L\left\{u(t-t_K)\right\} = \frac{e^{-t_K s}}{s} \tag{D.19}$$

Therefore:

$$L\left\{i(t)\right\} = i(s) = \frac{1}{s}\sum_{K=1}^{N} I_K\left(e^{-t_{K-1}s} - e^{-t_K s}\right) \tag{D.20}$$

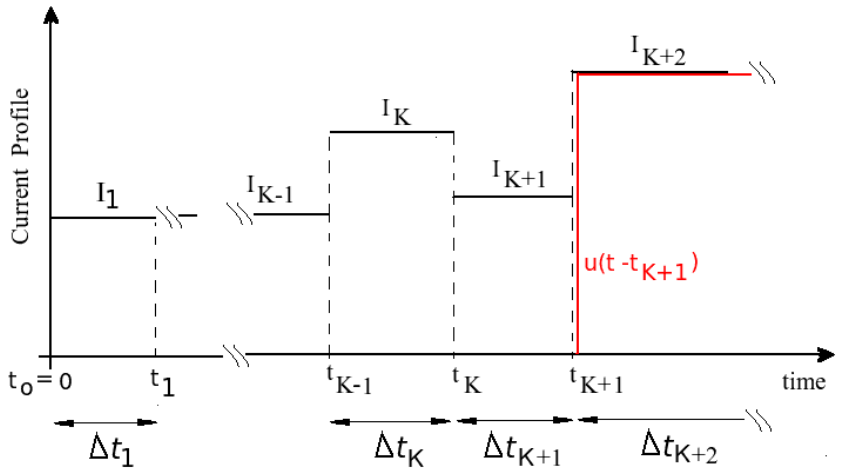


Figure D.3: **Assumed current consumption profile with $\Delta t_K \to 0$.**

We return to time domain, therefore we obtain:

$$L^{-1}\left\{\frac{\alpha i(s)}{s+2\beta}\right\} = L^{-1}\left\{\left(\frac{\alpha}{s+2\beta}\right)\frac{1}{s}\sum_{K=1}^{N} I_K\left(e^{-t_{K-1}s} - e^{-t_K s}\right)\right\} \tag{D.21}$$

$$L^{-1}\left\{\frac{\alpha i(s)}{s+2\beta}\right\} = \alpha L^{-1}\left\{\frac{1}{s}\sum_{K=1}^{N} I_K\left(\frac{e^{-t_{K-1}s}}{s+2\beta} - \frac{e^{-t_K s}}{s+2\beta}\right)\right\} \tag{D.22}$$

We have that:

$$L^{-1}\left\{\frac{e^{-t_K s}}{s+2\beta}\right\} = e^{2\beta(t-t_K)}u(t-t_K) \tag{D.23}$$

Therefore:

$$q_c(t) = \alpha L^{-1}\left\{\frac{1}{s}\sum_{K=1}^{N} I_K\left(\frac{e^{-t_{K-1}s}}{s+2\beta} - \frac{e^{-t_K s}}{s+2\beta}\right)\right\} \tag{D.24}$$

$$q_c(t) = \alpha\sum_{K=1}^{N}\int_0^t I_K\left\{e^{-2\beta(\tau-t_{K-1})}u(\tau-t_{K-1}) - e^{-2\beta(\tau-t_K)}u(\tau-t_K)\right\}d\tau \tag{D.25}$$

And thus:

$$q_c(t) = \alpha\sum_{K=1}^{N} I_K\left\{\int_{t_{K-1}}^t e^{-2\beta(\tau-t_{K-1})}d\tau - \int_{t_K}^t e^{-2\beta(\tau-t_K)}d\tau\right\} \tag{D.26}$$

$$q_c(t) = \alpha\sum_{K=1}^{N} I_K\left\{\frac{1-e^{-2\beta(t-t_{K-1})}}{2\beta} - \frac{1-e^{-2\beta(t-t_K)}}{2\beta}\right\} \tag{D.27}$$

$$q_c(t) = \frac{\alpha}{2\beta}\sum_{K=1}^{N} I_K\left\{e^{-2\beta(t-t_K)} - e^{-2\beta(t-t_{K-1})}\right\} = \alpha\frac{e^{-2\beta t}}{2\beta}\sum_{K=1}^{N} I_K\left\{e^{2\beta t_K} - e^{2\beta t_{K-1}}\right\} \tag{D.28}$$

Since $t_K = t_{K-1} + \Delta t_K$ with $2\beta\Delta t_K \ll 1$, thus by using $e^x \simeq 1 + x \quad , x \ll 1$. It is easy to see that $e^{2\beta t_k} - e^{2\beta t_{k-1}} \simeq e^{2\beta t_{K-1}} 2\beta\Delta t_K$ and the previous expression can be written as:

$$q_c(t) = \alpha\sum_{K=1}^{N} I_K e^{2\beta(t_{K-1}-t)}\Delta t_K = \alpha\sum_{K=1}^{N} e^{2\beta(t_{K-1}-t)}q_K \tag{D.29}$$

Where $q_K = I_K \Delta_K t_K$ is the charge delivered for the battery in $[t_{K-1}, t_K]$.
Finally:

$$Q_c(t) = \frac{1}{2}(Q_{c0} + Q_{a0})(1 - e^{-2\beta t}) + Q_{c0}e^{-2\beta t} + \alpha \sum_{K=0}^{N} q_K e^{2\beta(t_{K-1} - t)} \tag{D.30}$$

On the other hand, we observe that:

$$\frac{dQ_c(t)}{dt} + \frac{dQ_a(t)}{dt} = 0 \tag{D.31}$$

Hence $Q_a(t) = Q_{c0} + Q_{a0} - Q_c(t)$.

Previously we assumed that the battery voltage can be expressed as:

$$V_B(t) = V_{B0} - c \ln\left(\frac{[OH^-]_{cathode}}{[OH^-]_{anode}}\right) \tag{D.32}$$

With $Q_c(t) = [OH^-]_{cathode}$ and $Q_a(t) = [OH^-]_{anode}$

$$V_B(t) = V_{B0} - c \ln\left(\frac{Q_c(t)}{Q_a(t)}\right) \tag{D.33}$$

The final expression is:

$$V_B(t) = V_{B0} - c \ln\left(\frac{Q_c(t)}{Q_{c0} + Q_{a0} - Q_c(t)} \frac{Q_{a0}}{Q_{c0}}\right) \tag{D.34}$$

It was shown previously:

$$Q_c(t) = \frac{1}{2}(Q_{c0} + Q_{a0})(1 - e^{-2\beta t}) + Q_{c0}e^{-2\beta t} + \alpha \sum_{K=0}^{N} q_K e^{2\beta(t_{K-1} - t)} \tag{D.35}$$

Some observations deserve attention: the number N and time t are related by $t \leq t_N = \sum_{K=1}^{N} \Delta t_K$, moreover the values $Q_{c0}, Q_{a0}, \alpha, \beta$ are known in advance. They have to be measured with empirical battery characterization. However it is possible to reduce the model by re-writing the equation (D.35) as follows:

$$\lambda_c(t) = \frac{Q_c(t)}{Q_{c0}} = \frac{1}{2}\left(1 + \frac{Q_{a0}}{Q_{c0}}\right)(1 - e^{-2\beta t}) + e^{-2\beta t} + \frac{\alpha}{Q_{c0}} \sum_{K=1}^{N} e^{2\beta(t_{K-1} - t)} q_K \tag{D.36}$$

Where it is possible to define: $\lambda^{-1} = \dfrac{Q_{a0}}{Q_{c0}}$ and $\sigma = \dfrac{\alpha}{Q_{c0}}$. Therefore:

$$\lambda_c(t) = \frac{Q_c(t)}{Q_{c0}} = \frac{1}{2}\left(1 + \lambda^{-1}\right)(1 - e^{-2\beta t}) + e^{-2\beta t} + \sigma \sum_{K=1}^{N} e^{2\beta(t_{K-1} - t)} q_K \tag{D.37}$$

Finally, if it is assumed that at $t = 0$, the battery is a fresh one and it starts to provide current, the previous expression can be reordered to write the final battery voltage expression:

$$V_B(t) = V_{Bo} + c \ln\left[\left(\frac{Q_{c0} + Q_{a0}}{Q_c(t)} - 1\right)\frac{Q_{c0}}{Q_{a0}}\right] = V_{Bo} + c \ln\left[\left(\frac{1 + \lambda^{-1}}{\lambda_c(t)} - 1\right)\lambda\right] \tag{D.38}$$

The values that are not known in advance are λ , β , σ:

D.3 Preliminary numerical results

It has been implemented a program in order to extract the parameters of equation (D.38) for fitting the equation by least-squares criteria to experimental data. This *fitting* problem has been revealed as highly non-lineal.
We implemented programs in MATLAB, Scilab and Java. Only by using Java, interesting results have arisen after several days of computation. Results are depicted in **Figure D.4**. By means of other tools it has been impossible to obtain results due to the large amount of days performing computation without completion.
In **Figure D.4** it is depicted two fitting cases. The curve marked with circles (curve C_m) is a measured battery voltage discharge of a couple of batteries. It is the same for both cases A and B. Moreover, such batteries are the power supply of the MicaZ node.
The MicaZ node is programmed to transmit 1 packet every $50\,ms$ (i.e. transmission rate of 20 $packets/second$) at maximum output power level (i.e. 0 dBm). The voltage behavior has 3 zones clearly identified; zone 1 and 3 where the node is performing transmissions and zone 2 where the node sleeps and the battery suffers the relaxation process.
Curves C_1, C_2 and C_3 are the best found fit output curves that approximate the empirical curve with least-square criterion.
It clear that results are interesting and promissory, however, more research must be addressed.

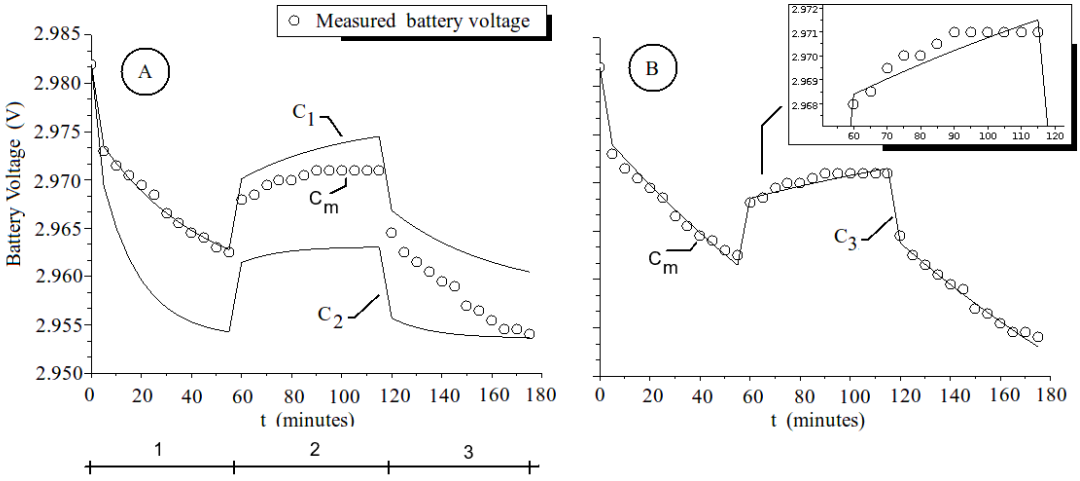


Figure D.4: **Algorithm working**

Appendix E

Modulation Methods in Wireless Sensor Nodes

E.1 Modulation Methods in Wireless Sensor Nodes

In literature it is offered and discussed modulation methods **[Razavi, B. (1998)]**, **[Carlson, B. (1986)]**. We focus attention on the modulation techniques compliant with the IEEE Std. 802.15.4 PHY **[IEEE Standard 802.15.4-2006]** .
The goal of this appendix is to offer an overview of the different modulation techniques for embedded wireless sensors nodes , and how the sub-sampling technique could be used for implementing demodulation.
By using sub-sampling several used reception circuits in the radio transceiver could become unnecessary, with the associated power consumption decrease.

Table E.1: **Frequency bands and data rates.**

Frequency Band (MHz)	Chip Rate (kchips/s)	Modulation	Bit Rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868-868.6	300	BPSK	20	20	Binary
902-928	600	BPSK	40	40	Binary
2400-2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

E.2 Phase Modulation methods

The $M-ary\ PSK$ is expressed as:

$$s_m(t) = A\sum_k cos\left(2\pi f_c t + \phi_k\right) p_b\left(t - kT_b\right) \tag{E.1}$$

Where:

- $\phi_k = \pi\left(2a_k + N\right)/M$: it is the phase shift in the time interval $[kT_b, (k+1)T_b]$, with:
  N : an integer that could be selected 0 or 1.
  M : number of discrete amplitudes.
  a_k : it is the sequence of digits of the input stream at rate $f_b = 1/T_b$, $a_k = 0, 1, \ldots, M-1$.
  The input digital signal (baseband) is assumed to be of the form $x_i(t) = \sum_k a_k p(t - kT_b)$.
- T_b : time interval. For the PSK modulation it is also named bit-time, for $QPSK$ is also defined in literature as dibit-time (a dibit is a quaternay symbol).
- f_c : carrier wave frequency. It has to be $T_b = Nf_s$, where N is as integer (i.e. T_b and f_c should to be harmonically related, being this relationship specified in the used communication standard.)
- $p_b(t)$: it is very common to use the *rectangular pulse* of width T_b. However it also could be used the *raised-cosine pulse shaping*.
- A : amplitude.

By using trigonometric expansions it is possible to write:

$$cos\,(2\pi f_c t + \phi_k) = cos\,(\phi_k)\,cos\,(2\pi f_c t) - sin\,(\phi_k)\,sin\,(2\pi f_c t) \tag{E.2}$$

Thus, the signal (E.1) for $kT_b < t < (k+1)T_b$ becomes:

$$s_m(t) = s_i(t - kT_b) - s_q(t - kT_b) \tag{E.3}$$

With:

$$s_q(t) = AQ_k sin\,(2\pi f_c t)\,p_b(t) \quad \texttt{with} \quad Q_k = sin\,(\phi_k) \tag{E.4}$$

And:

$$s_i(t) = AI_k cos\,(2\pi f_c t)\,p_b(t) \quad \texttt{with} \quad I_k = cos\,(\phi_k) \tag{E.5}$$

E.2.1 *BPSK*

To obtain $BPSK$ (or also know as binary PSK or $2-PSK$) signals, it is selected $N = 0$, $M = 2$ and $a_k = 0$, 1; then the signal becomes $s_0(t)$ for the time-bit slot where $a_k = 0$ and $s_1(t)$ for the time-bit slot where $a_k = 1$, where:

$$\begin{aligned} s_0(t) &= Acos\,(2\pi f_c t) \\ s_1(t) &= Acos\,(2\pi f_c t + \pi) = -s_0(t) \end{aligned} \tag{E.6}$$

In others words:

$$I_k = \pm 1, \quad Q_k = 0 \tag{E.7}$$

An example is depicted in **Figure E.1**.

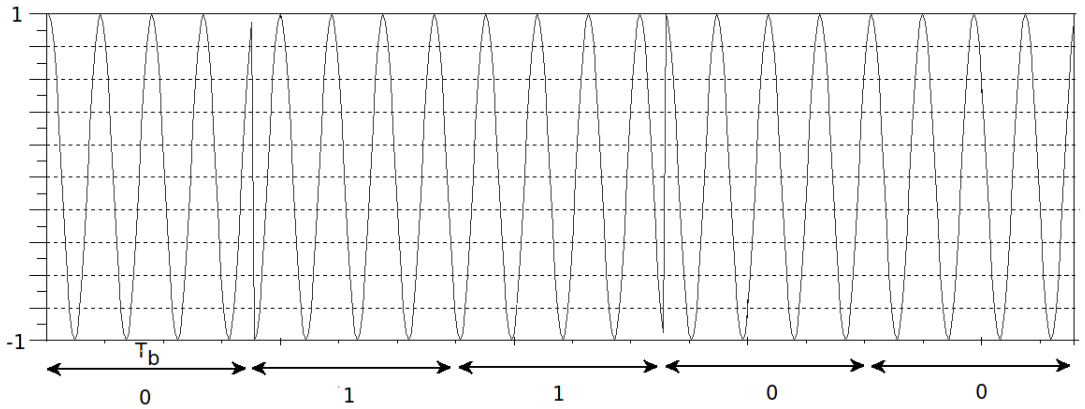


Figure E.1: $BPSK$ **signal. For this example we use the input bit stream** $a_k = [0\ 1\ 1\ 0\ 0]$, $f_c = 902\,MHz$ **and** $T_b = \dfrac{4}{f_c}$ **(please take into account that we use this last value for visualization purpose, actually, according to the Table E.1 the bit rate must be** $R_b = \dfrac{1}{T_b} = 600\,k\,chips/s$ **)**

According to **[Razavi, B. (1998)]** there are several possible receiver architectures. A particular one is the subsampling receiver. The theory is discussed in **[Razavi, B. (1998)]** where it is highlighted that the design of the local oscillator and synthesizer circuit can be simplified and trade offs relaxed. Nevertheless, an important drawback difficult the implementation and that is: aliasing of noise. The subsampling amplifies the sampling circuit noise and the clock phase noise as is described in **[Razavi, B. (1998)]**. The demodulated $BPSK$ signal by using subsampling in absence of noise is shown in **Figure E.2**.

E.2.2 *QPSK*

Quaternary or quadriphase PSK (also knows as $4-PSK$).
This modulation encodes two bits per symbol with four phases as follows: it is selected $N = 1$, $M = 4$ and $a_k = 0, 1, 2, 3$ (two bits per symbol), hence, the phase takes the values: $\phi_0 = \pi/4$, $\phi_1 = 3\pi/4$, $\phi_2 = 5\pi/4$ and $\phi_3 = 7\pi/4 = -\pi/4$. The signal constellation is shown in **Figure E.3**.

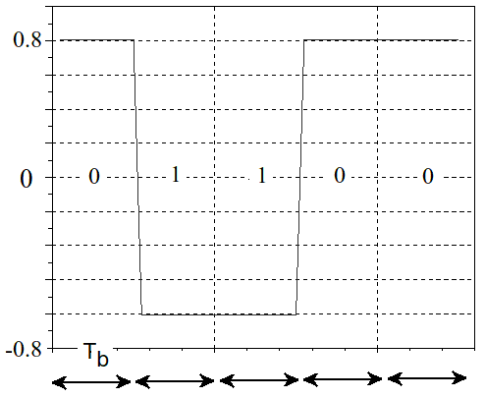


Figure E.2: **Sequence of source digits (baseband signal) obtained by subsampling the $BPSK$ signal shows in Figure E.1. The sampling frequency is $f_s = 0.4\,MHz$.**

In others words, the $QPSK$ modulation uses 4 phases, each possible phase represents two bits of the input data.
For instance, if we assume the input binary stream [0 0 0 1 1 0 1 1 0 1], then, a_k = [(0 0), (0 1), (1 0), (1 1), (0 1)] = $[0, 1, 2, 3, 1]$ and the phase mapping for each couple of bits is: $\phi_k = [\pi/4,\ 3\pi/4,\ 5\pi/4,\ 7\pi/4,\ 3\pi/4]$.

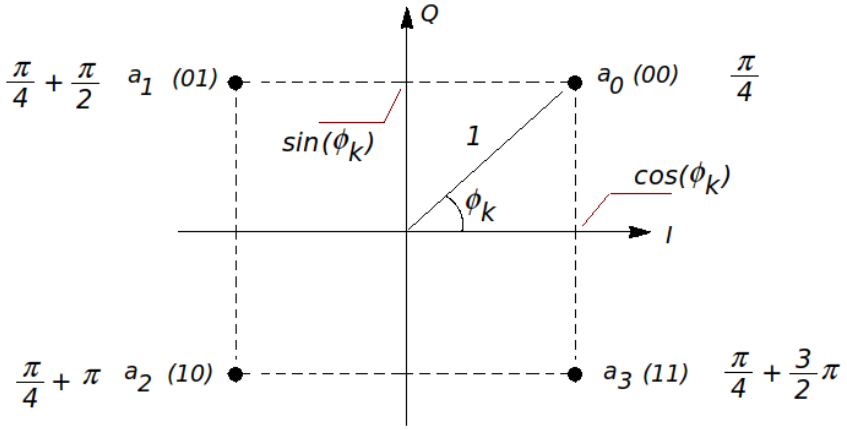


Figure E.3: $QPSK$ **signal constellation.**

E.2.3 $O-QPSK$

In this section we show how to build the Offset-keyed $QPSK$ signal. Let's assume that the input binary stream is [1 1 0 0 0 1 1 0]. This stream is separated in odd and even bits (or in phase and quadrature bits) to obtain $I = [1\ 0\ 0\ 1]$ and $Q = [1\ 0\ 1\ 0]$.
By using the new bit streams, it is generated a $QPSK$ signal, where the symbols have the following format: $a_k = [(I_1, Q_1),\ (I_2, Q_1),\ (I_2, Q_2), (I_2, Q_2), \ldots]$
In our case, we have that: a_k = $[(1,1),\ (0,1),\ (0,0),\ (0,0),\ (0,1)\ (1,1)\ (1,0)]$ and ϕ_k = $[7\pi/4,\ 3\pi/4,\ \pi/4,\ \pi/4,\ 3\pi/4,\ 7\pi/4,\ 5\pi/4]$. The example is depicted in **Figure E.4**.

In **Figure E.5** it is implemented the demodulation by **subsampling** (i.e. only a specialized ADC would be implemented into the wireless sensor node.)

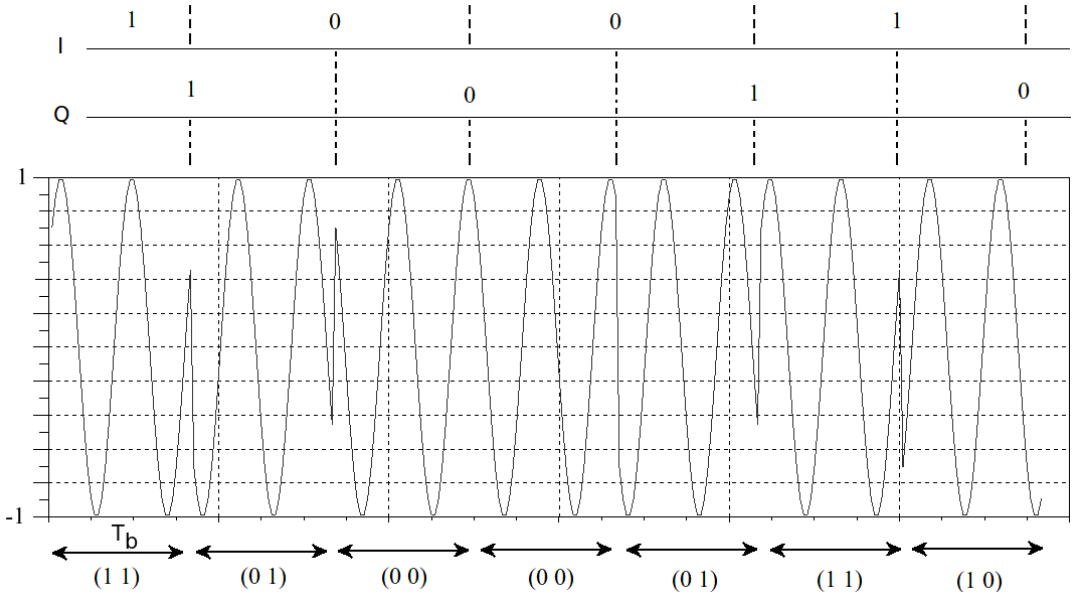


Figure E.4: $O-QPSK$ **signal. For this example we use** $f_c = 2.4\,GHz$ **and** $T_b = \frac{2}{f_c}$ **(please, take into account that we use this last value for visualization purpose, actually, according to the Table E.1 the bit rate must be** $R_b = \frac{1}{T_b} = 2\,M\,chips/s$ $T_b = 0.5\,\mu\,s$ **)**

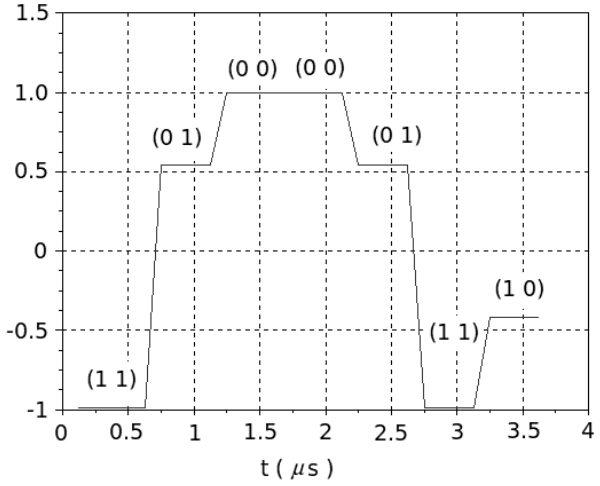


Figure E.5: **Demodulated** a_k **symbols of the** $O-QPSK$ **signal shown in Figure E.4 . For this example we use** $f_c = 2.4\,GHz$ **and** $T_b = 0.5\mu s$**, according to the Table E.1. The sampling frequency is** $f_s = 8\,MHz$

Bibliography

[**ADXL202, Datasheet**] *Analog Device Technical Data Low-Cost 2 g Dual-Axis Accelerometer with Duty Cycle Output ADXL202E Rev.A* Available Online: www.analog.com
Cited in Page(s): 76

[**Aeschlimann, F. (2004)**] F. Aeschlimann; E. Allier; L. Fesquet; M. Renaudin; *Asynchronous FIR filters: towards a new digital processing chain.* Proceedings. 10th International Symposium on Asynchronous Circuits and Systems, 19-23 April 2004 Page(s): 198-206 DOI 10.1109/ASYNC.2004.1299303
Cited in Page(s): 3

[**Agha, K. A. (2009)**] K. A. Agha; M. Bertin; T. Dang; A. Guitton; P. Minet; T. Val; J. Viollet; *Which Wireless Technology for Industrial Wireless Sensor Networks ? The Development of OCARI Technology* IEEE Transactions on Industrial Electronics, Vol. 56, No. 10, Octuber 2009 0278-0046/$26.00
Cited in Page(s): 9

[**Agostinho, P.R. (2008)**] P.R. Agostinho; S.A.P. Haddad; J.A. De Lima; W.A. Serdijn; *An ultra low power CMOS pA/V transconductor and its application to wavelet filters Algorithm.* Analog Integr. Circ. Sig. Process. (2008) 57:19-27 DOI 10.1007/s10470-008-9193-6; Springer
Cited in Page(s): 111, 112, 113

[**Ahn, H. (2009)**] Hyo-Sung Ahn; K. H. Ko; *Simple Pedestrian Localization Algorithms Based on Distributed Wireless Sensor Networks.* IEEE Transactions on Industrial Electronics, Vol. 56, No. 10, Octuber 2009 0278-0046/$26.00
Cited in Page(s): 9

[**Ammar, Y. (2008)**] Y. Ammar; A. Buhrig; M. Marzencki; B. Charlot; S. Basrour; K. Matou; M. Renaudin; *Wireless sensor network node with asynchronous architecture and vibration harvesting micro power generator.* Proceedings of the Joint Conference on Smart Objects and Ambient Intelligence (Grenoble, France, 2005). Page(s): 287-292, ISBN:1-59593-304-2
Cited in Page(s): 3

[**Anderson, R. (2004)**] R. Anderson; Data Acquisition Group *Understanding Ratiometric Conversions* Application Report SBAA110 - March 2004 Copyright ©2004, Texas Instruments Incorporated. Online http://focus.ti.com/lit/an/sbaa110/sbaa110.pdf
Cited in Page(s): 59

[**AN053, App. Note**] *Application Note AN053 Measuring power consumption with CC2430 and Z-Stack* Available Online: http://focus.ti.com/lit/an/swra144/swra144.pdf
Cited in Page(s): 11

[**AN693, App. Note**] *Understanding A/D Converter Performance Specifications DS00693A.* S. Bowling; Microchip Technology Inc.; Available Online: www.microchip.com
Cited in Page(s): 64, 65

[**ATmega128, Datasheet**] *8-bit AVR Microcontroller ATmega128 and ATmega128L Technical Data Rev. 2467O-AVR-10/06* Available Online: www.atmel.com
Cited in Page(s): ix, 1, 45, 46

[**AT45DB, Datasheet**] *4-megabit , 2.5-volt Only Serial DataFlash Thecnical Data Rev. 1432D-01/01* Available Online: www.analog.com
Cited in Page(s): 1

[**AVR201, App. Note**] *AVR201: Using the AVR Hardware Multiplier Rev. 1631C-AVR-06/02* Available Online: www.atmel.com
Cited in Page(s): 50

[**AVR223, App.Note**] *AVR223: Digital Filters with AVR Rev. 2527B-AVR-07/08* Available Online: www.atmel.com
Cited in Page(s): 50

[**AVR2002, App. Note**] *8-bit Instruction Set Instruction Set Nomenclature Rev. 0856D-AVR-08/02* Available Online: www.atmel.com
Cited in Page(s): 1

[**Awang, A. (2007)**] A. Awang; M. H. Suhaimi; *RIMBAMON: A Forest Monitoring System Using Wireless Sensor Networks* Proceedings of the International Conference on Intelligent and Advanced Systems 2007, 1101-1106, 25-28 Nov. 2007 1-4244-1355-9/07/$25.00 ©2007 IEEE
Cited in Page(s): 2

[**Bajwa, W. (2007)**] W. Bajwa; J. Haupt; A. Sayeed; R. Nowak; *Compressive Wireless Sensing.* Proceedings of the 5th international conference on Information Processing in Sensor Networks 2006 Page(s): 134-142 ISBN: 1-59593-334
Cited in Page(s): 3

[**Baker, G.A. (1975)**] G. A. Baker; *Essentials of Pade approximants.* Book, New York, Academic Press, 1975. ISBN 012074855X
Cited in Page(s): 111

[**Barberis, A. (2007)**] A. Barberis; L. Barboni; M. Valle; *Evaluating Energy Consumption in Wireless Sensor Networks Applications.* Proceedings of The 10th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools-Germany ISBN 0-7695-2978-X/07.
Cited in Page(s): 10

[**Barboni, L. (2008)**] L.Barboni; M.Valle; *Experimental Analysis of Wireless Sensor Nodes Current Consumption.* Proceedings of The Second International Conference on Sensor Technologies and Application August,2008 France. 978-0-7695-3330-8/08 $25.00 ©2008 IEEE DOI 10.1109/SENSORCOMM.2008.14
Cited in Page(s): 3

[**Barsanti, R.J. (2007)**] R.J. Barsanti; Ch. Lehman; *Applications of a wavelet-based Receiver for the Coherent Detection of FSK Signals.* 40th Southeastern Symposium on System Theory University of New Orleans New Orleans USA, March 16-18, 2008 978-1-4244-1807-7/08 IEEE.
Cited in Page(s): 84

[**Bohte, S.M (2002)**] S. M. Bohte; J. N. Kok; H. La Poutren; *Spike-prop: error-backpropagation in multi-layer networks of spiking neurons.* In Neurocomputing, 48 (1-4): 17-37, 2002.
Cited in Page(s): 124

[**Boero, F. (2007)**] F.Boero; A.Leoncini; *Tesi di Laurea- Reti di Sensori Radio: Caraterizzazione del Canale di Trasmissione e Misura di Parametri Ambientali.* Anno Accademico 2006-2007- DIBE Universita degli Studi di Genova, Italy
Cited in Page(s): 35, 36

[**Bougard, B. (2005)**] B. Bougard; F. Catthoor; D.C. Daly; A. Chandrakasan; W. Dehaene; *Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives* Proceedings of the conference on Design, Automation and Test in Europe 2005, Page(s): 196-201 ISSN:1530-1591
Cited in Page(s): 10, 11

[**Cantatore, E. (2006)**] E. Cantatore; M. Ouwerkerka; *Energy scavenging and power management in networks of autonomous microsensors.* Copyright ©2006 Elsevier Ltd. DOI: 10.1016/j.mejo.2006.04.014
Cited in Page(s): 3

[**Carlson, B. (1986)**] B. Carlson; *Communication Systems. An Introduction to Signal and Noise in Electrical Communications* Third Edition 1986 McGraw-Hill Publisher ISBN: 0-07-100560-9
Cited in Page(s): 60, 61, 67, 147

[**CC2420 Radio Device**] *CC2420 2.4 GHz IEEE 802.15.4 ZigBee-ready RF Transceiver Chipcon* Product from Texas Instruments; Available Online: http://focus.ti.com/docs/prod/folders/print/cc2420.html
Cited in Page(s): 1, 14, 18, 19, 31, 33, 35, 40, 43

[**Chandrakasan, A. (2002)**] A. Chandrakasan; A. Wang; *Energy-efficient DSPs for wireless sensor networks* Signal Processing Magazine, IEEE. Jul 2002, Volume 19, Issue 4, Page(s): 68-78 ISSN: 1053-5888
Cited in Page(s): 3

[**Chang, M. (2006)**] M. Chang; *Evaluation of Accelerometers Mounted on Wireless Sensor Motes* Joint technical report with Cambridge University) Technical Report no. 06/02 ISSN: 0107-8283 Available Online: www.diku.dk/hjemmesider/ansatte/marcus
Cited in Page(s): 86

[**Chen, S.W.M. (2007)**] S.W.M. Chen; R.W. Brodersen; *A Subsampling Radio Architecture for Ultrawideband Communications.* IEEE Transactions on Signal Processing Volume 55, Issue 10, Oct. 2007 Page(s): 5018-5031 DOI 10.1109/TSP.2007.896056
Cited in Page(s): 71

[**Chiu, Y. (2005)**] Y. Chiu; B. Nikolic; P.R. Gray; *Scaling of analog-to-digital converters into ultra-deep-submicron CMOS.* Proceedings of the IEEE Custom Integrated Circuits Conference, 21-21 Sept 2005 Page(s): 375-382 DOI: 10.1109/CICC.2005.1568684
Cited in Page(s): 58

[**Cho, Y. (2007)**] Y. Cho; Y. Kim; N. Changt; *PVS: Passive Voltage Scaling for Wireless Sensor Networks.* In proceedings of the ISLPED 07, August 27-29, 2007, Portland, Oregon, ACM 9781595937094/ 07/0008 $5.00
Cited in Page(s): 13

[**Cloth, L. (2007)**] L. Cloth; M.R. Jongerden; B.R. Haverkort; *Computing Battery Lifetime Distributions.* 37th Annual IEEE IFIP International Conference on Dependable Systems and Networks (DSN'07) 0-7695-2855-4/07 2007
Cited in Page(s): 140

[**Dallago, E. (2007)**] E. Dallago; P. Malcovati; D. Miatton; T. Ungaretti; G. Venchi; *Analysis of sigma-delta converters for MEMS sensors using power supply voltage as reference.* Proceedings of Circuits, Devices and Systems,Vol. 153, October 2006 pp:473-479 DOI10.1049/ip-cds:20060079
Cited in Page(s): 59

[**Dlugosz, R. (2007)**] R. Dlugosz; K. Iniewski; *Flexible Architecture of Ultra-Low-Power Current-Mode Interleaved Successive Approximation Analog-to-Digital Converter for Wireless Sensor Networks.* Hindawi Publishing Corporation VLSI Design (2007), Article 3, ISSN:1065-514X
Cited in Page(s): 13, 58

[**Dondi, D. (2008)**] D. Dondi; A. Bertacchini; D. Brunelli; L. Larcher; L. Benini; *Modeling and Optimization of a Solar Energy Harvester System for Self-Powered Wireless Sensor Networks.* IEEE Transactions on Industrial Electronics, Vol. 55, No. 7, July 2008 0278-0046/$26.00
Cited in Page(s): 9

[**Fang, Z. (2006)**] Z. Fang; Z. Zhaol; H. Zeng; Q. Wang; H. Dong; P. Guol; Y. Zhang; *Ultra-Low Power WSN Node with Integrated THP Sensor.* 1st IEEE International Conference on Nano/Micro Engineered and Molecular Systems, 18-21 Jan. 2006 Page(s):813-816 DOI 10.1109/NEMS.2006.334902
Cited in Page(s): 3

[**Ferentinos, K. P. (2006)**] K. P.Ferentinos; T.A. Tsiligiridis; *Adaptive Design Optimization of Wireless Sensor Networks Using Genetic Algorithms.* In Computer Networks Elsevier 51 (2007) Paage(s) 1031-1051, 1389-1286/$2006 B.V DOI:10.1016/j.comnet.2006.06.013
Cited in Page(s): 29

[**Gao, L. (2007)**] L. Gao; S. Liu; R. A. Dougal; *Dynamic Lithium-Ion Battery Model for System Simulation.* IEEE Transactions on Components and Packaging Technologies, Vol. 25, No. 3, September 2002 Page(s): 495-505 521-3331/0217.00 2002 IEEE
Cited in Page(s): 141

[**Gay, P. (2003)**] P. Gay; R. Levis; M. Welsh; E. Brewer; D. Culler; *The nesc language: A holistic approach to networked embedded systems.* In Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation. ACM Press, 2003, Page(s). 1-11
Cited in Page(s): 14

[**Gray, R. M. (1990)**] Robert M. Gray *Quantization Noise Spectra* IEEE Transactions on Information Theory Vol. 36, No. 6, November 1990 0018-9448/90/ 1100-1220$01.00 1990 IEEE
Cited in Page(s): 68

[**Greenstein, B. (2006)**] B. Greenstein; C. Mar; A. Pesterev; S. Farshchi; et.al; *Capturing high-frequency phenomena using a bandwidth-limited sensor network.* Proceedings of the 4th international conference on Embedded networked sensor systems, 2006. Colorado, USA Page(s): 279-292 ISBN:1-59593-343-3
Cited in Page(s): 86, 120

[**Gungor, Vehbi C. (2009)**] V.C. Gungor; Gerhard P. Hancke; *Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches.* IEEE Transactions on Industrial Electronics, Vol. 56, No. 10, October 2009 0278-0046/$26.00
Cited in Page(s): 9

[**Gunst, A.W. (2004)**] A.W. Gunst; A.B.J. Kokkeler; *Modeling Correlation of Quantized Noise and Periodic Signals.* IEEE Signal Processing Letters Vol. 11, No 10, October 2004 Page(s): 802.805, 1070-9908/04$20.00 ©2004 IEEE
Cited in Page(s): 64

[**Haddad, S.A.P. (2009)**] S.A.P. Haddad; W. A. Serdijn; *Ultra Low-Power Biomedical Signal Processing. An Analog Wavelet Filter Approach for Pacemakers* ISBN: 978-1-4020-9072-1 Springer Science 2009
Cited in Page(s): 67

[**Haddad, S.A.P (2005)-1**] S.A.P. Haddad; S. Bagga; W.A. Serdijn; *Log-domain wavelet bases.* IEEE Transactions on Circuits and Systems I: Regular Papers, Volume 52, Issue 10, Oct. 2005 Page(s): 2023-2032 DOI 10.1109/TCSI.2005.853360
Cited in Page(s): 111, 112, 113

[**Haddad, S.A.P (2005)-2**] S.A.P. Haddad; J.M.H. Karel; R.L.M. Peeters; R.L. Westra; W.A. Serdijn; *Analog complex wavelet filters.* IEEE International Symposium on Circuits and Systems, 2005. (ISCAS 2005) Page(s): 3287-3290 Vol.4 ISBN: 0-7803-8834-8
Cited in Page(s): 112

[**Haddad, S.A.P (2005)-3**] S.A.P. Haddad; J.M.H. Karel; R.L.M. Peeters; R.L. Westra; W.A. Serdijn; *Ultra low-power analog Morlet wavelet filter in 0.18 μm BiCMOS technology.* Solid-State Circuits Conference, 2005. ESSCIRC 2005. Proceedings of the 31st European 12-16 Sept. 2005, Page(s): 323-326 DOI 10.1109/ESSCIR.2005.1541625
Cited in Page(s): 84

[**Haykin,S. (1996)**] S. Haykin; *Adaptive Filter Theory*
Third Edition -Printice Hall Inc. 1996 ISBN 0-13322760-X
Cited in Page(s): 60, 67, 73

[**Ho, K.C. (2000)**] K.C. Ho; W. Prokopiw; Y.T. Chan; *Modulation identification of digital signals by the wavelet transform.* IEE Proc. Radar Sonar Navig. August 2000 Volume 147, Issue 4, Pages(s): 169-176 DOI:10.1049/ip-rsn:20000492
Cited in Page(s): 84

[**Holger, K. (2005)**] K. Holger; A. Willig; *Protocols and Architectures for Wireless Sensor Networks.* Publisher John Wiley Sons. ISBN-13 978-0-470-09510-2. 2005
Cited in Page(s): 10, 11

[**IEC 9899:TC2**] *Programming languages-C International Standard ©ISO/IEC ISO/IEC 9899:TC2*; Available Online: www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf
Cited in Page(s): 48, 49

[**IEEE Standard 802.15.4-2006**] *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)* Available Online: http://standards.ieee.org/getieee802/802.15.html
Cited in Page(s): 1, 147

[**INA139 Datasheet**] *High-Side Measurement Current Shut Monitor INA 139. Datasheet Burr-Brown Products from Texas Instruments.* Available Online : http://focus.ti.com
Cited in Page(s): 14

[**Jaffard, S. (2004)**] S. Jaffard; Y. Meyer; *Wavelet methods for pointwise regularity and local oscillations of functions.* Memoirs of the American Mathematical Society, Volume 123, Number 587, September 1996 ISSN 0065-9266
Cited in Page(s): 85

[**Jain, A. (2004)**] A. Jain; E.Y. Chang; *Adaptive Sampling for Sensor Networks.* Proceedings of the First International Workshop on Data management for Sensor Networks Toronto, Canada 2004 Page(s): 10-16 http://doi.acm.org/10.1145/1052199.1052202
Cited in Page(s): 3

[**Jaing, B. (2007)**] B. Jiang; J.R. Smith; M. Philipose; S. Roy; K. Sundara-Rajan; A.V. Mamishev; *Energy Scavenging for Inductively Coupled Passive RFID Systems.* Transactions on Instrumentation and Measurement, Volume 56, Issue 1, Feb. 2007, Page(s):118-125
Cited in Page(s): 3

[**Jiang, X. (2007)**] X. Jiang; J. Taneja; J. Ortiz; A. Tavakoli; P. Dutta; J. Jeong; D. Culler; P. Levis; S. Shenker; *An Architecture for Energy Management in Wireless Sensor Networks.* Special issue on the Workshop on wireless sensor network architecture Volume 4, Issue 3 ,2007. Page(s): 31-36 ISSN 1551-3688
Cited in Page(s): 10, 29

[**Jiang, X. (2007)-2**] X. Jiang; P. Dutta; D. Culler; I. Stoica; *Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale.* Proceeding of The Sixth International Conference on Information Processing in Sensor Networks, April 25-27, 2007 ACM 978-1-59593-638-7/07/0004 Page(s): 186-195
Cited in Page(s): 13, 14

[**Jin, Q.(1994)**] Q. Jin; K.M. Wong; Q. Wu; *Optimum Wavelet Design for Transient Detection.* IEEE Sevent SP Workshop on Statistical Signal and Array Processing June 26-29,1994 Page(s): 255-258
Cited in Page(s): 85

[**Kan, B. (2007)**] B. Kan; L. Cai; L. Zhao; Y. Xu; *Energy Efficient Design of WSN Based on An Accurate Power Consumption Model.* In proceedings of the Wireless Communications, Networking and Mobile Computing (WiCom 2007), Page(s): 2751-2754 1-4244-1312-5/07/$25.00 IEEE
Cited in Page(s): 10, 27

[**Karel, J.M.H. (2005)**] J.M.H. Karel; R.L.M. Peeters; R.L. Westra; S.A.P. Haddad; W.A. Serdijn; *Wavelet Approximation For Implementation In Dynamic Trasnlinear Circutis.* In proceedings of IFAC World Congress 2005, Prague, July 4-8.
Cited in Page(s): 113, 114

[**KiBaM**] http://www.ceere.org/rerl/projects/software/batteryModel.html
Cited in Page(s): 141, 142

[**Kim, S. (2007)**] S. Kim; S. Pakzad; D. Culler; J. Demmel; G. Fenves; S. Glaser; M. Turon; *Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks.* In 6th International Symposium on Information Processing in Sensor Networks, Cambridge, MA, 25-27 April 2007, Page(s):254-263 ISBN: 978-1-59593-638-7
Cited in Page(s): 2, 86

[**Kim, Y. (2008)**] Y. Kim; R.G. Evans; W.M. Iversen; *Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network.* In IEEE Transactions on Instrumentation and Measurement Vol. 57, No. 7, July 2008 ISSN: 0018-9456 Page(s): 1379-1387
Cited in Page(s): 3

[**Konstantakos, V. (2007)**] V. Konstantakos; A. Chatzigeorgiou; S. Nikolaidis; Th. Laopoulos; *Energy Consumption Estimation in Embedded Systems.* IEEE Transactions on Instrumentation and Measurement, Vol. 57, No. 4, April 2008 Page(s): 797-804
Cited in Page(s): 14

[**Kyaw, Z.T. (2007)**] Z.T. Kyaw; C. Sen; *Using the CC2430 and TIMAC for low-power wireless sensor applications: A power consumption study.* Available Online: http://focus.ti.com.cn/cn/lit/an/slyt295/slyt295.pdf
Cited in Page(s): 11

[**Laffea, L. (2006)**] L.Laffea; R.Monson; R.H.S. Oncley; J.Sun et.al; *Comprehensive Monitoring of CO_2 Sequestration in Subalpine Forest Ecosystems and Its Relation to Global Warming.* SenSys 06, 423-425 November 1-3, 2006 ACM 1-59593-343-3/06/0011.
Cited in Page(s): 2

[**Lopez, M.A. (2008)**] M.A.Lopez; M.Valle; *Soft Acoustic Event Detectors for Limited Resources Platforms.* In ISCCSP 2008, Malta, 12-14 March 2008 978-1-4244-1688-2/08/$25.00-2008 IEEE
Cited in Page(s): 76

[**LTC4150 Datasheet**] *Coulomb Counter - Battery Gas Gauge LTC4150 from Linear Technologies.* Available Online: www.linear.com
Cited in Page(s): 14, 139

[**Lu, B. (2006)**] B. Lu; T.G. Habetler; R.G.Harley; *A Novel Motor Energy Monitoring Scheme using Wireless Sensor Networks.* Industry Applications Conference, 2006. 41st IAS Annual Meeting. Conference Record of the 2006 IEEE; Volume 5, Issue , 8-12 Oct. 2006 Page(s):2177-2184 DOI 10.1109/IAS.2006.256844
Cited in Page(s): 2

[**Lu, B. (2009)**] B. Lu; V.C. Gungor; *Online and Remote Motor Energy Monitoring and Fault Diagnostics using Wireless Sensor Networks.* IEEE Transactions on Industrial Electronics, Vol. 56, No. 11, November 2009 0278-0046/$26.00
Cited in Page(s): 9

[**Luo, X. (2008)**] X. Luo; Georgios B. Giannakis; *Energy-constrained optimal quantization for wireless sensor networks.* EURASIP Journal on Advances in Signal Processing Volume 2008 (January 2008) Regular Issue Articles, Article No. 73 ISSN:1110-8657
Cited in Page(s): 5

[**Lynch, J.P. (2004)**] J.P. Lynch; Y. Wang; A. Sundararajan; K.H. Law; A.S. Kiremidjian; *Wireless Sensing For Structural Health Monitoring of Civil Structures* . Proceedings of International Workshop on Integrated Life-Cycle Management of Infrastructures, Hong Kong, December 9-11, 2004. Available Online: http://eil.stanford.edu/publications
Cited in Page(s): 86

[**Mallat, S.G. (1992)**] S.G. Mallat; W.L. Hwang; *Singularity Detection and Processing with Wavelets.*. IEEE Trans. on Information Theory Vol 38, No. 2, March 1992, Page(s): 617-643, 0018-9448/92
Cited in Page(s): 85

[**Mallat, S.G. (1999)**] Stephane Mallat; *A Wavelet Tour of Signal Processing* Second Edition. Academic-Press Elsevier 1998-1999 ISBN-13: 978-0-12-466606-1
Cited in Page(s): 83

[**Martin, T.L. (1999)**] Thomas L. Martin; *Balancing Batteries, Power and Performance: System Issues in CPU Speed-Setting for Mobile Computing* A Dissertation Submitted to the Graduate School in Partial Fulfilment of the Requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering CARNEGIE MELLON UNIVERSITY August, 1999
Cited in Page(s): 23

[**Mathur, G. (2006)**] G. Mathur; P. Desnoyers; D.Ganesan; P. Shenoy; *UltraLow Power Data Storage for Sensor Networks* In IPSN 06, April 19-21, 2006, Nashville, Tennessee, USA. Copyright 2006 ACM 1595933344/06/0004
Cited in Page(s): 13

[**Matsuzawa, A. (2007)**] A. Matsuzawa *Design Challenges of Analog-to-Digital Converters in Nanoscale CMOS.* IEICE Trans.. Electron. Vol.E90-C, No.4 April 2007 DOI: 10.1093/ietele/e90c.4.779 Page(s): 779-785 Special Section on Low-Power, High-Speed LSIs and Related Technologies.
Cited in Page(s): 58

[**McIntire, D. (2006)**] D. McIntire; K. Ho; B. Yip; A. Singh; W. Wu; W. J. Kaiser; *The Low Power Energy Aware Processing (LEAP) Embedded Networked Sensor System.* Proceding of The IPSN06, April 19-21, 2006 Nashville, Tennessee, USA Copyrigth 2006 Page(s): 449-457, ACM 1-59593-334-4/06/0004$5.00
Cited in Page(s): 17

[**MICAz (Crossbow)**] *Technical data MPR-MIB Users Manual Revision B, June 2006 PN: 7430-0021-07* Available Online: www.xbow.com
Cited in Page(s): 1, 14, 18

[**MLC, Mote Battery Life Calculator**] *Expected battery life vs. system current usage and duty cycle, PowerManagement.xls.* Excel Worksheet from Crossbow Technology. [Online]: www.xbow.com/Support/Support_pdf_files/PowerManagement.xls
Cited in Page(s): 11, 12, 20, 23, 34

[**MMA7360 Datasheet**] *Freescale Semiconductor Technical Data ±1.5g, ±6g Three Axis Low-g Micromachined Accelerometer Document Number: MMA7360L Rev 2, 8/2007.* Available Online: www.freescale.com
Cited in Page(s): 86

[**Montepaldi FARM WSN TestBed**] *Montepaldi FARM WSN TestBed 2nd GoodFood Newsletters 27-02-2006.* Online : http://www.goodfood-project.org/
Cited in Page(s): 28

[**MOTEWORK Tool (Crossbow)**] *MoteWork ™Crossbow - Platform for the development of Wireless Sensor Network from Crossbow.* Available Online: www.xbow.com/Products/productdetails.aspx?sid=154
Cited in Page(s): 14

[**MTS/MDA Board Datasheet**] *Technical data. MTS/MDA Sensor Board Users Manual Revision A, June 2007 PN: 7430-0020-05* Available Online: www.xbow.com
Cited in Page(s): 16

[**O'Driscoll, S. (2005)**] S. O'Driscoll; T.H. Meng; *Adaptive ADC design for neuro-prosthetic interfaces: base ADC cell.* Proceedings of the 2005 European Conference on Circuit Theory and Design, 28 Aug-2 Sept. 2005 Vol. 1, Page(s): 301-304 DOI 10.1109/ECCTD.2005.1522970
Cited in Page(s): 59, 71

[**OMNeT++**] OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, with an Eclipse-based IDE and a graphical runtime environment. Domain-specific functionality (support for simulation of communication networks, queuing networks, performance evaluation, etc.) [Online] : http://www.omnetpp.org/
Cited in Page(s): 34

[**Panigrahi, D. (2001)**] D. Panigrahi; S. Dey; R. Rao; K. Lahiri; C. Chiasserini; A. Raghunathan; *Battery Life Estimation of Mobile Embedded Systems.* The 14th International Conference on VLSI Design (VLSID 01) January 03-January 07,ISBN: 0-7695-0831-6, DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/ICVD.2001.902640
Cited in Page(s): 13

[**Peebles, P. (1987)**] Peebles Peyton Z.; *Digital Communication Systems* Publisher Prentice-Hall International,Inc. Printed 1987 ISBN 0-13-211962-5
Cited in Page(s): 68

[**Perez-Alcazar, P.R. (2002)**] P.R. Perez-Alcazar; A. Santos; *Relationship between sampling rate and quantization noise.* In 14th International Conference on Digital Signal Processing 2002 Volume: 2, Page(s): 807-810 DOI: 10.1109/ICDSP.2002.1028213 0-7803-7503-3/02/$17.00 ©2002 IEEE
Cited in Page(s): 68

[**Plassche, R. (1987)**] Rudy van de Plassche; *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters* Kluwer Academic Publisher 2nd Edition Printed 2003; ISBN 1-4020-7500-6
Cited in Page(s): ix, 62, 64, 65, 71

[**Porter, J. (2005)**] J. Porter; P. Arzberger et.al; *Wireless Sensor Networks for Ecology BioScience* Vol. 55 No. 7, 561-572 , July 2005 (http://www.coralreefeon.org/bioscience%20wireless%20networks%202005.pdf)
Cited in Page(s): 2

[**Prakasam, P. (2008)**] P. Prakasam; M. Madheswaran; *Digital Modulation Identification Model Using Wavelet Transform and Statistical Parameters.* Hindawi Publishing Corporation Journal of Computer Systems, Networks, and Communications, Volume 2008, Article ID 175236, 8 pages, DOI:10.1155/2008/175236
Cited in Page(s): 84

[**Qingxiu, H. (2003)**] H. Qingxiu; H. Yigang; *Analog CMOS High-Frequency Continuous Wavelet Transform Implemented By Instantaneous Companding Circuits.* Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing Changsha, China, October 2003, 0-7803-7925-xl03/$17.0002003 IEEE
Cited in Page(s): 111

[**Rabaey, J.M. 2000**] J.M. Rabaey; M.J. Ammer; J.L. da Silva Jr.; D. Patel; S. Roundy; *Pico-Radio supports ad hoc ultra-low power wireless networking.* IEEE Computer, July 2000, Volume: 33, Issue: 7 Page(s): 42-48.
Cited in Page(s): 3

[**Rakhmatov, D.N. (2001)**] D. Rakhmatov; S. Vrudhula; *An analytical high-level battery model for use in energy management of portable electronic systems.* Proceedings of the 2001 IEEE ACM international conference on Computer-aided design Page(s): 488-493, ISBN:0-7803-7249-2
Cited in Page(s): 23

[**Rakhmatov, D.N. (2003)**] D. Rakhmatov; S. Vrudhula; *Energy Management for Battery-Powered Embedded Systems.* ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003, Page(s): 277-324, ACM 1539-9087/03/0800-0277 $5.00
Cited in Page(s): 13

[**Rao, R. (2003)**] R. Rao; S. Vrudhula; D. Rakhmatov; *Battery Modeling for Energy-Aware System Design.* Published by the IEEE Computer Society 0018-9162/03/$17.00 ©2003 IEEE
Cited in Page(s): 13

[**Rao, R. (2005)**] R. Rao; S. Vrudhula; N. Chang; *Battery optimization vs energy optimization: Which to choose and when ?* International Conference on Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM 6-10 Nov. 2005 Page(s): 439-445 0-7803-9254-X/05/$20.00 ©2005 IEEE
Cited in Page(s): 13

[**Razavi, B. (1998)**] B. Razavi; *RF Microelectronics.* Printice Hall PTR 1998 ISBN: 0-13-887571-5
Cited in Page(s): 147, 148

[**Sakunia, S. 2007**] S. Sakunia; S. Bhalerao; A. Chaudhary; M. Jyotishi; M. Dixit; R. Jumade; R. Patrikar; *UbiSens: Achieving Low Power Wireless Sensor Nodes.* IFIP International Conference on Wireless and Optical Communications Networks, 2-4 July 2007, Page(s): 1-6, DOI: 10.1109/WOCN.2007.4284225.
Cited in Page(s): 3

[**Sawigun, C. (2009)**] C. Sawigun; M. Grashuis; R. Peeters; W. Serdijn; *Nanopower Sampled Data Wavelet Filter Design Using Switched Gain Cell Technique.* In proceedings of The IEEE International Symposium on Circuits and Systems May 2009 (ISCAS 2009), Page(s): 545-548 ISBN 978-1-4244-3828-0/09 IEEE
Cited in Page(s): 84

[**Sangiacomo, F. (2007)**] F. Sangiacomo; A. Sciutto; *Reti di Sensori Radio: Algoritmi di Signal Processing per Rilevazione di Eventi nel Caso di Piattaforme Hardware con Risorse Limitate.* Tesi di Laurea: Corso de Laurea in Ingegneria Elettronica 2007- DIBE University Degli Studi di Genova.
Cited in Page(s): 16

[**Saponara, S. (2007)**] S. Saponara; L. Fanucci; P. Terreni; *Architectural-Level Power OPtimization of Microcontroller Cores in Embedded Systems.* IEEE Transactions on Industrial Electronics, Vol. 54, No. 1, February 2007 0278-0046/$25.00
Cited in Page(s): 10

[**Sarkar, T.K. (1992)**] T.K. Sarkar; C. Su; R. Adve; M. Salazar-Palma; L. Garcia-Castillo, R.R. Boix; *A Tutorial on Wavelets from an Electrical Engineering Perspective, Part 1: Discrete Wavelet Techniques.* IEEE Antennas and Propagation Magazine, Vol.40, No. 5, Octuber 1998 1045-9243/98/$10.00©1998 IEEE
Cited in Page(s): 83

[**Serra-Graells, F. (2003)**] F. Serra-Graells; A. Rueda; J.L. Huertas; *Low-Voltage CMOS Log Companding Analog Design.* Kluwer Academic Publishers 2003 eBook ISBN: 0-306-48721-7 Print ISBN: 1-4020-7445-X
Cited in Page(s): 116

[**Shelby, Z. (2005)**] Z. Shelby; C. Pomalaza-Raez; H. Karvonen; J. Haapola; *Energy Optimization in Multihop Wireless Embedded and Sensor Networks.* In the International Journal of Wireless Information Networks, Vol. 12 No. 1 January 2005 DOI:10.1007/s10776-005-5166-1
Cited in Page(s): 29, 41

[**SLEWS Proyect**] *Project SLEWS (Sensor based Landslide Early Warning System)* . [Online] Available : http://www.winsoc.org./dissemination.htm
Cited in Page(s): 2

[**SmartRF Studio**] *SmartRF®Studio is a Windows application that can be used to evaluate and configure Low Power RF ICs from Texas Instruments.* Available Online: http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html
Cited in Page(s): 26, 35

[**Tian, Q. (2006)**] Q. Tian; N. Yamauchi; *A new Method of Noise Removal for Body Vibration Signal in Wireless Sensor Networks.* International Journal of Innovative Computing, Information and Control Volume 2, Number 6, December 2006, ICIC International ISSN 1349-4198
Cited in Page(s): 84

[**TinyOS**] Available Online: http://www.tinyos.net
Cited in Page(s): 11, 16, 21, 56

[**Toumazou, C. (2002)**] Edited by C. Toumazou; G. Moschytz; B. Gilbert; *Trade-offs in Analog Circuit Design.* Kluwer Academic Publishers 2002 ISBN 1-4020-7037-3 **Cited in Page(s):** 59, 116

[**Tsea, P.W. (2004)**] P.W. Tsea; W. Yang; H.Y. Tam; *Machine Fault Diagnosis Through an Effective Exact Wavelet Analysis.* Journal of Sound and Vibration 277 (2004) Page(s): 005-1024, 2003 Elsevier Ltd. All rights reserved. DOI:10.1016/j.jsv.2003.09.031
Cited in Page(s): 84

[**Valle, M. (Ed.) (2007)**] Maurizio Valle (Ed.) *Smart Adaptive Systems on Silicon.* Kluwer Academic Publisher ISBN 1-4020-2743-5 (HB)
Cited in Page(s): 124

[**Verma, N. (2007)**] N. Verma; A.P. Chandrakasan, *An Ultra Low Energy 12-bit Rate-Resolution Scalable SAR ADC for Wireless Sensor Nodes* IEEE Journal of Solid-State Circuits, Vol. 42, No 6, June 2007 DOI: 10.1109/JSSC.2007.897157
Cited in Page(s): 59

[**Virtanen, K. (2005)**] K. Virtanen; M. Pankaala; A. Paasio; *A current-mode ADC with adaptive quantization.* IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005; 23-26 May 2005 Page(s): 3115-3118 Vol. 4 ISBN: 0-7803-8834-8
Cited in Page(s): 59

[**Vittoz, E.A. (2003)**] Eric A. Vittoz; *Weak Inversion in Analog and Digital Circuits.* In CCCD Workshop 2003, Lund, Oct. 2-3 [Online] http://www.es.lth.se/cccd/images/CCCD03-Weak inversion-Vittoz.pdf Author email: eric.vittoz@csem.ch
Cited in Page(s): 116

[**Vitterli, M. (1992)**] M. Vitterli; C. Herley; *Wavelets and Filter Banks: Theory and Design.* IEEE Transactions on Signal Processing Vol. 40; No. 9 Sept. 1992 1053-587X/92 $03.00 ©1992 IEEE Page(s): 2207-2232
Cited in Page(s): 83, 88, 94

[**Wadgy, M.F. (1989)**] M.F. Wadgy; Wai-man Ng; *Validity of Uniform Quantization Error Model for Sinusoidal Signals Without and With Dither.* IEEE Transactions on Instrumentations and Measurements Vol. 38 No. 3, June 1989, Page(s): 718-722 DOI 10.1109/19.32180
Cited in Page(s): 68

[**Walnut, D.F. (2004)**] David F. Walnut; *An Introduction to Wavelets Analysis* Printed 2004, Birkhauser (Spring Science and Business Media) www.birkhauser.com; ISBN:0-8176-3962-4
Cited in Page(s): 83

[**Wong, P.W. (1990)**] P.W. Wong; *Quantization Noise, Fixed-point Multiplicative Roundoff Noise and Dithering.* IEEE Transactions on Acoustic Speech and Signal Processing. Vol. 38. No. 2, February 1990; 0096-3518/90/0200-0286$01.00
Cited in Page(s): 68

[**Xing, G. (2005)**] G. Xing; C. Lu; Y. Zhang; Q. Huang; R.Pless *Minimun Power Configuration in Wireless Sensor Network.* In Proceeding of The 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing MobiHocâ05, May 25â27, 2005, Urbana-Champaign, Illinois, USA. Copyright 2005 ACM 1-59593-004-3/05/0005 ...$5.00.
Cited in Page(s): 29

[**Zhang, J.(2008)**] J. Zhang; W.Li; N. Han; J. Kan; *Forest fire detection system based on a ZigBee wireless sensor network, Frontiers of Forestry in China.* Higher Education Press, co-published with Springer-Verlang GmbH, Volume 3, Number 3, September 2008 369-374, 2008 ISSN1673-3517 (Print) 1673-3630 (Online)
Cited in Page(s): 2

[**Zuniga, M. (2004)**] M. Zuniga; B. Krishnamachari; *Analyzing the transitional region in low power wireless links.* In First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), 2004, Page(s):517-526
Cited in Page(s): 34, 35

[**Zbigniew, R. S. (Report)**] R. Struzik Zbigniew; *Local Effective Holder Exponent Estimation on the Wavelet Transform Maxima Tree.* Report Centre for Mathematics and Computer Science (CWI) Kruislaan 413, 1098 SJ Amsterdam, The Netherlands email: Zbigniew.Struzik@cwi.nl06, Online: http://old-www.cwi.nl/themes/ins1/publications/docs/St:FTAE:99.pdf
Cited in Page(s): 85