UNIVERSITÀ DEGLI STUDI DI GENOVA

Facoltà di Ingegneria

Dipartimento di Ingegneria Biofisica ed Elettronica

UNIVERSITY COLLEGE CORK

Microelectronic department

Embedded low-power system for respiration rate calculation

Tesi di Laurea Specialistica (M Sc. Thesis)

Supervisors: Prof. MARCO STORACE
Prof. EMANUEL POPOVICI

Authors: CANU ALESSANDRO
CANU MASSIMILIANO

A.A. 2009/2010

Abstract

Aim of this thesis was to develop a low power, embedded system, for the estimation of the respiratory rate (RR) using wearable sensors. In order to reach this goal, we focused our attention onto two promising and non-invasive devices: the finger pulse oximeter (or SpO2 sensor) and the chest accelerometer. As the system has been thought to work with wireless data, the sensors have been mounted on nodes in a wireless sensors network.

In particular, this work can be theoretically divided in three main parts:

- Communication: in order to read the information, a driver has been written to read the sensors and send the data, wireless, to the network node where the computation was performed. Both the communication and the data reading codes, were designed with the goal of minimizing the overall power consumption, as much as possible.

- Signal processing: different solutions for the RR estimation, using the SpO2 or the accelerometer, have been found in literature. In this thesis, we analyzed pros and cons for the different algorithms found, and we described the implementations of the methods adopted (one for each sensor).

- Performance analysis: the algorithms implemented, one for the SpO2 and one for the accelerometer, have been evaluated in terms of execution time and power consumption, a comparison among results is presented.

All of the weakness and strength points of the proposed implementations are then discussed in the last chapter, regarding results and conclusions.

The system was implemented and tested at the Microelectronics Department in the University College Cork, with the support of the Tyndall National Institute, Cork, Ireland, which provided the hardware used.

Dichiarazione del relatore

Alla Commissione di Laurea e di Diploma
Alla Commissione Tirocini e Tesi

Sottopongo la tesi redatta dagli studenti Canu Alessandro e Canu Massimiliano dal titolo *"Embedded low power system for respiration rate calculation"*.

Ho esaminato, nella forma e nel contenuto, la versione finale di questo elaborato scritto, e propongo che la tesi sia valutata positivamente assegnando i corrispondenti crediti formativi.

Il Relatore Accademico
Prof. Marco Storace

Acknowledgements

To our parents, our cousins and our friends... Grazie mille!

Contents

List of Figures

List of acronyms

ADC - Analog to Digital Converter

AM - Amplitude Modulation

AR - Autoregressive model

ASK - Amplitude Shift Keying

CWT - Complex Wavelet Transform

DCO - Digital Controlled Oscillator

DMA - Direct Memory Access

FFCDM - Fixed-Frequency Complex Demodulation

FFT - Fast Fourier Transform

FM - Frequency Modulation

FSK - Frequency Shift Keying

GFSK - Gaussian Frequency Shift Keying

HR - Heart Rate

IF - Intermediate Frequency

I^2C - Inter-Integrated Circuit

ISM - Industrial, Scientific and Medical radio band

JTAG - Joint Test Action Group

MAC - Media Access Control

OOK - On-Off Keying

PLL - Phase Looked Loop

PPG - Photoplethysmogram

PSD - Power Spectral Density

RAP - Ridge Amplitude Perturbation signal

RFP - Ridge Frequency Perturbation signal

RISC - Reduced instruction set code

RR - Respiration rate

RSSI - Received Signal Strength Indicator

SNR - Signal to Noise Ratio

SPI - Serial Peripheral Interface

SWFD - Secondary Wavelet Feature Decoupling

TFS - Time-Frequency Spectrum

UART - Universal Asynchronous Receiver Transmitter

USB - Universal Serial Bus

VCO - Voltage Controlled Oscillator

VFCD - Variable Frequency Complex Demodulation

WBAN - Wireless Body Area Network

WLAN - Wireless Local Area Network

Chapter 1

State of the Art for wireless sensor networks

1.1 The wireless context

The importance of the wireless technology has grown considerably in the last years for several reasons, and first of all is the wireless communication market. Wireless technology now reaches or is capable of reaching virtually every location on the face of the earth. Hundreds of millions of people exchange information every day using laptops, personal digital assistants (PDAs), pagers, cellular phones, and other wireless communication devices; this trend has been led by growing popularity of Internet services such as World Wide Web, e-mail, instant messaging, file transfers, RSS feed, etc [23].

The capabilities needed to deliver such services are characterized by an increasing need for data throughput in the network, and applications now in development indicate that this trend will continue at an exponential rate. Among the most popular wireless networking standards there are IEEE 802.15 (also known as Bluetooth) and the IEEE 802.11 family, which allow different devices such as laptops, PDAs, printers, cameras, mobile phones, controllers, speakers, etc. to communicate each other within a Wireless Local Area Network (WLAN, or WPAN, where P is for "personal", in case Bluetooth is employed).

This kind of network is typically limited to a home or a big office, since the range of a WLAN goes from meters (in the case of Bluetooth) to hundreds of meters for IEEE 802.11x. In order to cover a bigger area, a wired interconnection of different wireless access points (AP) or different WLANs is needed, but this is quite difficult to apply for example in covering a metropolitan area, since a new access point should be placed every 100-200 meters, with enormous costs, not mentioning the frequency of the hand-off between APs, needed in order to guarantee customers' mobility.

To overcome these limits a new technology called WiMAX (IEEE 802.16) is gaining credit, which has the potential to provide full mobile broadband wireless connectivity to the Internet over a Metropolitan Area scale (MAN), including features to protect signals from degradation in high time-varying environments. It could also be a way to reduce the problem of digital divide, namely the difficulty for isolated places to gain access to an Internet connection at a reasonable cost, due to the prohibitive cost of building a wired infrastructure in order to serve just a few customers.

Since a single base station would have an estimated range of 40-50 Km (when using directional antennas), enough capacity to allow high data rates, and relatively low implementation costs, WiMAX will probably be the next step for high-speed wireless data networks.

On the other hand, other potential wireless network applications exist. These applications, which have relaxed throughput requirements and are often measured in a few bits per day, include industrial control and monitoring, home automation and consumer electronics, security and military sensing, asset tracking and supply chain management, intelligent agriculture, and health monitoring .

Because most of these low-data-rate applications involve sensing of some physical process, networks supporting them have been called Wireless Sensors Networks (WSNs). Several key differences between more traditional ad-hoc networks and WSNs exist.

Individual nodes in a WSN have limited computational power and storage capacity. They operate on non-renewable power sources and typically employ a

short-range transceiver to send and receive messages. The number of nodes in a WSN can be several orders of magnitude higher than in an ad hoc network. Thus, algorithm scalability is an important design criterion for WSN applications.

Sensor nodes are often (depending on the application) densely deployed in the area of interest. This dense deployment can be leveraged by the application, since nodes in close proximity can collaborate locally prior to relaying information back to the base station.

Sensor networks are prone to frequent topology changes. This is due to several reasons, such as hardware failure, depleted batteries, intermittent radio interference, environmental factors, or the addition of sensor nodes. As a result, applications require a degree of inherent fault tolerance and the ability to reconfigure themselves as the network topology evolves over time.

The study of WSNs is quite challenging indeed, due to the fact it requires an enormous breadth of knowledge from a number of different disciplines. The objective of this chapter, anyway, would be just trying to give a brief overview of the main issues and applications of WSNs.

1.1.1 Applications of WSNs

The aim of this section is to give an overview of the types of applications wireless sensor networks are suitable for, before focusing on the specific application here discussed. The applications for WSNs are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring, object tracking, fire detection, land slide detection and traffic monitoring. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes. Some examples are:

- *Intelligent agriculture* – Large farms and ranches may cover several square miles, and they may receive rain only sporadically and only on some portions of the farm. Irrigation is expensive, so it is important to know which fields have received rain and which fields have not, so that irrigation may be performed more efficiently. Such an application is ideal for wireless sensor

networks, because the amount of data sent over the network can be very low (just a simple flag like rain= yes, no), and the message latency can be on the order of minutes or hours. Yet the costs and power consumption must be low enough for the entire network to cover a very wide area and to last an entire growing season.

In such application however, a WSN is capable of much more than just performing soil moisture measurements, because the network can be fitted with a near-infinite variety of chemical and biological sensors, capable of providing the farmer with a graphical view of temperature, the need for pesticides, herbicides, and fertilizers, received sunshine, and other quantities. This type of application is especially important in vineyards, where subtle environmental changes may have large effects on the value of the crop and how it is processed.

- *Military area monitoring* – is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. For example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion, or placed under sea to monitor for instance the passage of intruders in an harbor. When the sensors detect the event being monitored (heat, pressure, sound, light, electro-magnetic field, vibration, etc.), the event is reported to one of the base stations, which then takes appropriate action (e.g., send a message on the internet or to a satellite). Similarly, wireless sensor networks can use a range of sensors to detect the presence of vehicles ranging from motorcycles to train cars.

- *Greenhouse Monitoring* – WSNs are also used to control the temperature and humidity levels inside commercial greenhouses. When the temperature and humidity drops below specific levels, the greenhouse manager must be notified via e-mail or cell phone text message, or host systems can trigger misting systems, open vents, turn on fans, or control a wide variety of system responses. Because some wireless sensor networks are easy to install, they

are also easy to move as the needs of the application change.

- *Wireless Body Area Networks (WBAN)* – These are networks containing sensor nodes in close proximity to a person's body monitoring vital signals of the human body and a more intelligent node capable of handling more advanced signal processing. There are number of possible sensors that can form a WBAN, such as EEG, ECG, EMG sensors, temperature, pulse-rate sensors etc. A BAN network displaced on a patient can alert the hospital, even before he has a heart attack, through measuring changes in his vital signs; or warn a diabetic patient to inject insulin as soon as his insulin level declines.

1.1.2 Common requirements

Wireless sensor networks are still a developing technology and, as stated above, they are used in a lot of different kinds of applications, so it is easy to understand that many factors influence the optimization of a WSN design. Anyway, independently from our standpoint, there are some common critical factors: power consumption, overall costs, hardware constraints, and security.

Low cost

Cost plays a fundamental role in applications adding wireless connectivity to inexpensive or disposable products, and for applications with a large number of nodes in the network, such as wireless supermarket price tags. These potential applications require wireless links of low complexity that are low in cost relative to the total product cost [23].

To meet this objective, the communication protocol and network design must avoid the need for high-cost components, by employing relaxed tolerances wherever possible, and reduce silicon area by minimizing protocols complexity and memory requirements.

In addition, however, it should be recognized that one of the largest costs of many networks is administration and maintenance. To be a true low-cost

system, the network should be ad hoc and equipped with self-configuration and self-healing procedures. An "ad hoc" network in this context is defined to be a network without a predetermined physical distribution, or logical topology, of the nodes.

"Self-configuration" is intended as the ability of network nodes to detect the presence of other nodes and to organize themselves into a structured network without human intervention. "Self-healing" is defined to be the ability of the network to detect (and recover from) faults appearing in either network nodes or communication links, again without human intervention.

Power consumption

The single most important consideration for a wireless sensor network is power consumption. While the concept of wireless sensor networks looks practical and exciting on paper, if batteries are going to have to be changed constantly, widespread adoption will not occur. Therefore, when the sensor node is designed power consumption must be minimized. from experimental measurement we can note that, by far, the largest power consumption is attributable to the radio link itself [30].

There are a number of strategies that can be used to reduce the average supply current of the hardware and the overall network power request, and they will be largely discussed in the next section of this chapter (1.2).

Hardware constraints

Since the final network project must have a low cost and have to save as much energy as possible, it is natural consequence of the previous two points that the hardware which is at designer disposal is strongly constrained. Thus, the modules of the sensor nodes must be reasonably small low-complexity devices, for this reason embedded hardware is used, often integrating several functions on the same chip. However, a typical sensor node equipment must include at least a sensor device (it may be a mechanical, thermal, optical, chemical, acoustic, or another kind of sensor), a processing unit (microcontroller, DSP, FPGA), a

memory (typically embedded in the processing unit) and a transceiver, plus all the circuitry needed to let them work together.

Security

Sensor networks can be used in a number of domains that handle sensitive information. Due to this, there are many considerations that should be investigated and are related with protecting sensitive information traveling between nodes (which are either sensor nodes or the base station) from being eavesdropped by unauthorized third parties. But develop security in WSNs it's a complex and vast issue, involving several different requirements that have to be catered to; among the most important there are confidentiality, authentication and integrity.

Confidentiality requirement is needed to ensure that sensitive information is well protected and not revealed to unauthorized third parties. The confidentiality objective is required in sensors' environment to protect information traveling between the sensor nodes of the network or between the sensors and the base station from disclosure, since an adversary having the appropriate equipment may eavesdrop on the communication and, doing that, he could overhear critical information such as sensing data and routing information and may cause severe damages.

As in conventional systems, ***authentication*** techniques verify the identity of the participants in a communication, distinguishing in this way legitimate users from intruders. In the case of sensor networks, it is essential for each sensor node and base station to have the ability to verify that the data received was really sent by a trusted sender and not by an adversary that tricked legitimate nodes into accepting false data. If such a case happens and false data are supplied into the network, then the behavior of the network could not be predicted and most of times will not outcome as expected.

Moving on to the ***integrity*** objective, there is the danger that information could be altered when exchanged over insecure networks, so an integrity control must be implemented to ensure that information will not be altered in any unexpected way. Lack of integrity could result in many problems since the conse-

quences of using inaccurate information could be disastrous, for example for the health care sector where lives are endangered.

Therefore, there is urgent need to make sure that information is traveling from one end to the other without being intercepted and modified in the process. Of course, the integrity control can be implemented at different levels in the communication stack and by using different methods.

All of these aspects are very important, but they are just a little part of the vast problem of WSNs' security. However, further details are outside the scope of this thesis, and they will not be given here.

1.1.3 Topology of a WSN

The most popular examples of different network topologies are shown in the following figure:

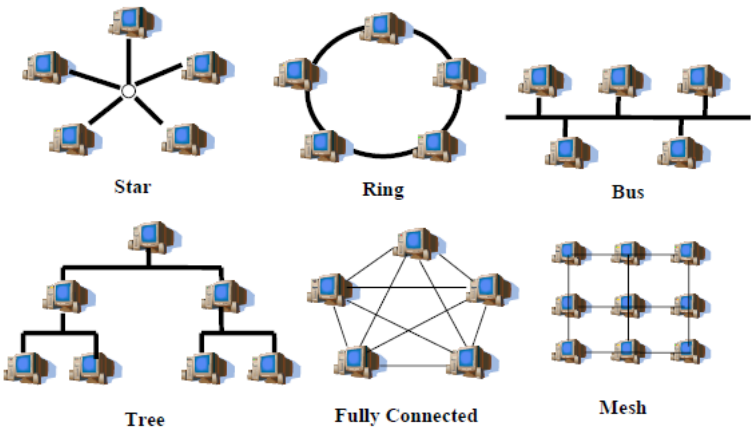

Figure 1.1: *Examples of several different network topologies*

Each of these choices obviously brings its own advantages and disadvantages.

A conventional star network employing a single master and one or more slave devices may satisfy many applications. However, because the transmit power of the network devices is limited by battery life concerns, this network design (or topology) limits the physical area a network may serve to the range of a single device (i.e., the master).

When additional range is needed, network types that support multi-hop routing (e.g., mesh or cluster types) must be employed; the additional memory and computational cost for routing tables or algorithms, in addition to network maintenance overhead, must be supported without excessive cost or power consumption.

Note that the regular structure reflects the communications topology; the actual geographic distribution of the nodes need not be a regular mesh. Since there are generally multiple routing paths between nodes, these nets are robust to failure of individual nodes or links. An advantage of mesh nets is that, although all nodes may be identical and have the same computing and transmission capabilities, certain nodes can be designated as 'group leaders' that take on additional functions. If a group leader is disabled, another node can then take over these duties [22].

However, the organization of the network is also strictly influenced by the strategy of sensor nodes deployment in the physical environment, which may take several forms. Nodes may be deployed at random (e.g., by dropping them from an aircraft) or installed at deliberately chosen spots. Placement may also be a one-time activity or a continuous process, with more nodes being deployed at any time during the use of the network, for example to replace failed nodes or to improve the network coverage. The actual type of deployment affects important properties such as the node density, node locations, regular patterns in node locations, and the expected frequency of network reconfiguration.

A typical strategy that involves all the points discussed so far is "in-network data aggregation".

This is an essential data-processing primitive in sensor networks. Sensor nodes forward data towards the sink (i.e., the base station which will forward the collected data to the querier). As sensor nodes closer to the sink receive data from nodes further away, they aggregate the information into concise digests. This results in significant energy savings over having each node forward its respective readings directly to the sink.

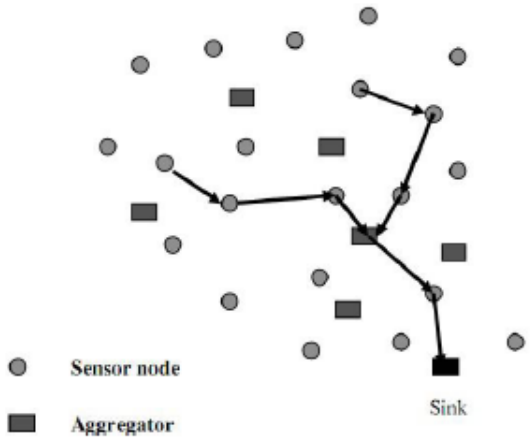

Figure 1.2: *Data aggregation scheme example.*

1.2 Low-power WSN design basics

In the same way that fuel consumption is important in everything from scooters to oil tankers, power consumption is a key parameter in most electronics applications. The most obvious applications for which power consumption is critical are battery-powered applications, such as home thermostats and security systems, in which the battery must last for years. Low power also leads to smaller power supplies, less expensive batteries, and enables products to be powered by signal lines (such as fire alarm wires) lowering the cost of the end-product. As a result, low power consumption has become a key parameter of microcontroller designs, and it is one of the most important constraints in WSNs [24].

1.2.1 Parameters for low-power design

In this section, we intend to briefly discuss about some main parameters we can act on, in order to low down the overall power consumption of our WSN [25].

WSN topology – Topology control is the art of coordinating nodes' decisions regarding their transmitting ranges in order to generate a network with the

desired properties (e.g., connectivity) while reducing node energy consumption and/or increasing network capacity.

This is a powerful mean to change the appearance and properties of a network in a way that significantly improve operational aspects, such as lifetime and energy consumption. For example, in a densely deployed wireless network, a single node has many neighboring nodes with which direct communication would be possible when using sufficiently large transmission power. This is, however, not necessarily beneficial because high transmission power requires a lot of energy.

The number of neighboring nodes determines the number of receivers and total power consumption is reduced for topologies with fewer neighbors. To overcome this problem, topology control can be applied to decrease the neighbor nodes [29]. There are many techniques that can be used to restrict the nodes that are considered as neighbors. This can be done by controlling transmission power, by introducing hierarchies in the network and signaling out some nodes to take over certain coordination tasks, or by simply turning off some nodes for a certain time. So by use of topology control techniques, the topology of the network is deliberately restricted in order to decrease the power consumption [29].

MAC protocol – Medium Access Control (MAC) protocol manages directly the communication hardware (see chapter 4), so it must be designed to reduce the sources of energy waste. The major sources of energy waste in conventional MAC protocols are packet collisions, idle listening, overhearing, and overhead. Collisions can happen if the MAC protocol allows two or more nodes to send packets at the same time. In this case none of transmitted packets can be received correctly so the nodes must retransmit their packets and this causes an increase in energy consumption. In addition to this, latency will be increased as a result of collision. The second source of energy waste is idle listening.

In order to eliminate or reduce collisions, nodes must sense the channel

continuously to obtain scheduling information or wait before sending data until the channel is detected idle. This process is called *idle listening*.

Measurements have shown that energy consumption in idle mode does not differ much from that in Transmit or Receive mode and idle listening consumes 50-100% of the energy required for receiving. So long periods of idle listening may also increase energy consumption and decrease network throughput.

Another source of energy waste is *overhearing*, which occurs when a node picks up packets destined to other nodes. Overhearing increases energy consumption and also degrades the network throughput.

A last source of energy waste in wireless sensor networks is *overhead*. This refers to the situation when too many control packets ate exchanged within the network.

Control packets are used for signaling, scheduling, routing updates, and collision avoidance and their transmission consumes extra energy.

In general, in order to design an energy-efficient MAC protocol, collisions must be avoided as much as possible. Moreover, energy dissipation due to idle channel sensing, overhearing, and overhead should also be reduced to the minimum.

One way to implement this optimization, is to switch a network's node to a sleep state when it becomes "*idle*" (i.e., when they are neither transmitting nor receiving). To maintain communication, each node must wake itself up periodically to listen for a packet intended for it.

Microcontroller choice – Generally, the computing/processing unit is a microcontroller or microprocessor. In most applications, microcontroller is used as a computation device and also performs some simple processing. It is the core of the system which do the decision making task in each node.

In order to choose the ideal microcontroller for the system, the designer must consider the desired performance level. Another important point is

microcontroller's operational modes. Most microcontrollers support various operational modes and the power consumption in each mode is different. The most common modes in microcontrollers are *active, idle, and sleep modes*. For example Texas Instruments MSP430 features a wide range of operation modes: one fully operational mode, which consumes about 1.2mW and four sleep modes. The deepest sleep mode only consumes 0.3 μW but the controller is only woken up by external interrupts in this mode. In the next higher mode, a clock is also still running, which can be used for scheduled wake ups, and still consumes only about 6 μW [27]. Anyway, swapping between one power mode to the other is a power consuming task.

Transceiver choice – The transceiver is the piece of hardware transforms a bit stream from the microcontroller into radio waves: this is usually the largest power consumer and optimizing its power can result in significant improvement at the system level [28].

There's a vast range of transceiver devices on the market, but designers have to be careful: transceivers that appear to have excellent energy characteristics might suffer from other shortcomings like poor frequency stability under temperature variations (leading to partitioning of a network when parts of the node are placed in the shade and others in sunlight), high susceptibility to interference on neighboring frequency channels, or undesirable error characteristics; they could also lack features that other transceivers have, like tunability to multiple frequencies.

So the designer must be aware of the most essential factors which are crucial for the particular application and based on this knowledge he/she can choose the most appropriate transceiver.

Routing protocol – Traditional routing protocols are designed in order to minimize communication's delay, while many of wireless sensor networks' protocols try to minimize the energy required for communication, since nodes in a sensor network are energy-constrained. The characteristics of the environment within which sensor nodes typically operate, coupled with severe

resource and energy limitations, make the routing problem very challenging. The path selection must be such that the lifetime of the network is maximized. There are some general guidelines one can follow to design a routing protocol that leads to energy saving.

Generally speaking, it is better to avoid forwarding packets through regions of the network that are running low on energy resources, thus preserving them for future, possibly critical detection and communication tasks. Nodes can also use resource adaption mechanism and put a threshold for their energy level. If the energy level of a node approaches the threshold, the node can reduce its participation in the protocol operations and reduce or completely eliminate certain activities, such as forwarding control packets or data packets which are not addressed to that node.

Another technique to minimize energy consumption is to avoid links with high communication costs, because congestion always results in an increase in communication cost.

For example, the designer can use a control mechanism such that the sink slows down sources that cause congestion and speeds up sources with lower communication costs. In addition to this, the protocol should ensure that connectivity in a network is maintained as long as possible: one good method is to avoid forwarding packets via the same path because the nodes on that path will run out of energy in a short time and so the network connectivity will be compromised. Instead, the load should be uniformly spread over the network, which leads to a smoother degradation of the network.

1.3 Hardware used

The hardware used is based on the 25mm Tyndall platform, which is a technology developed by the "Tyndall National Institute" in Cork, Ireland. Such hardware platform is a 25mm x 25mm stackable developmental platform, designed to

be modular in nature and to be suitable for a variety of WSN applications. Different layers can be combined in an innovative and robust plug and play fashion and include communications (transceivers), processing (microcontroller or FPGA module) and a variety of sensing interconnections, sensor layers and power supply layers [31].

In our case, we used the "ADF7020/MSP430 layer", which is a transceiver plus processor feature layer containing an ADF7020 transceiver and a TI MSP430 microcontroller.

All the software in the project has been developed by using Texas Instruments' "Code Composer Studio" compiler. The communication interface between the compiler and the MSP430 has been realized by using the Tyndall 25mm programming board (figure 1.3), which is an interface platform for the various node layers.

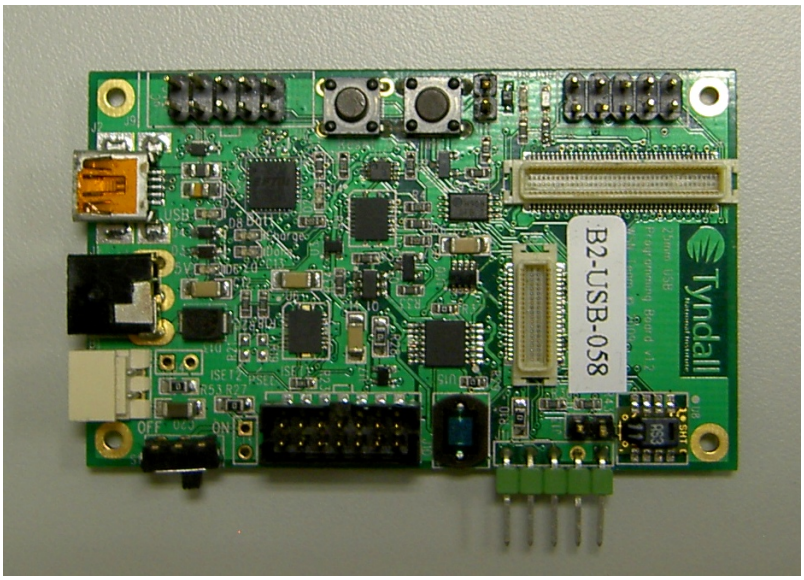

Figure 1.3: *25mm Tyndall programming board.*

The board role is to program the 25mm mote through both USB and JTAG interfaces.

Mote communication is carried out also through the USB interface (there's an USB UART built-in function on-board), so it is possible to show what happens inside the microcontroller directly on a serial port of the PC. To upload the

built code into the CPU, USB interface is used as well, even if a specific JTAG programmer is required to access the microcontroller. It is also possible to use several on-board sensors and a memory that can be interfaced by the various 25mm mote layers and used during the debug phase.

1.3.1 Transceiver/processor layer

As mentioned before, in this kind of motes either the transceiver (ADF7020 by Analog Devices) or the microcontroller (MSP430F5437 by Texas Instruments) are set on the same stack layer. Figure 1.4 below shows a picture of top and bottom part of the Tyndall *ADF7020 - MSP430* layer.

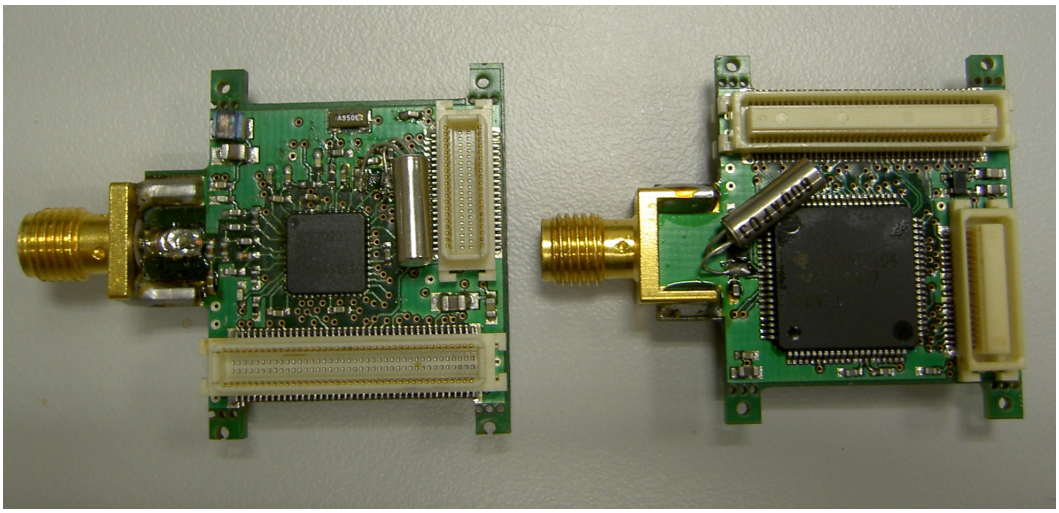

Figure 1.4: *Tyndall mote: ADF7020 chip is on the top (left), and MSP430 is on the bottom (right).*

The aim of this section is to see with greater detail some of the main features of these devices.

MSP430F5437 microcontroller

This device belongs to a Texas Instruments family of ultra low-power microcontrollers, featuring different sets of peripherals targeted for various applications [27]. The architecture, combined with five low-power modes, is optimized to

achieve extended battery life in portable applications, so it fits very well for wireless sensor networks.

The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. Constant generators are "hardwired registers" to store commonly used values. Typically there is only one value per register but the designers of the MSP430 cleverly exploited its four addressing modes to get four values from its dedicated constant generator (called R3/CG2).

In the end, MSP430 has 27 native instructions, and further 24 emulated instructions are defined to make life easier for the programmer. These include common operations such as "clear," which is implemented as an ordinary move with a value of 0 provided by the constant generator; or "an increment", implemented as a traditional add operation with a constant value of 1 [32].

About the memory system of the MSP430 device we used, it has 16kB RAM (beginning at 0200h and contiguous up to its final address), and 64kB of flash memory. RAM is used for all scratch-pad variables, global variables, and the stack. Flash memory, instead, must be written by programmers by using special instructions, and it is divided in 4 non-contiguous different banks. Using of flash memory has to be done with extreme care, since the first bank (where the program is stored) is splitted in two different halves by the interrupt vector registers. Peripherals are also mapped into registers.

Other important features are the 12-bit analog-to-digital (A/D) converter, an hardware multiplier supporting 32-bit operations, DMA, real-time clock module with alarm capabilities, and up to four universal serial communication interfaces, which allow the implementation of different standard buses: IrDA, synchronous/asynchronous SPI and I^2C. Moreover, this microcontroller is equipped with a digitally controlled oscillator (DCO) which allows "CPU wake-up" from low-power modes to active mode (see also section 3.1.1).

A simple block scheme of the F5437 microcontroller is shown in figure 1.5:

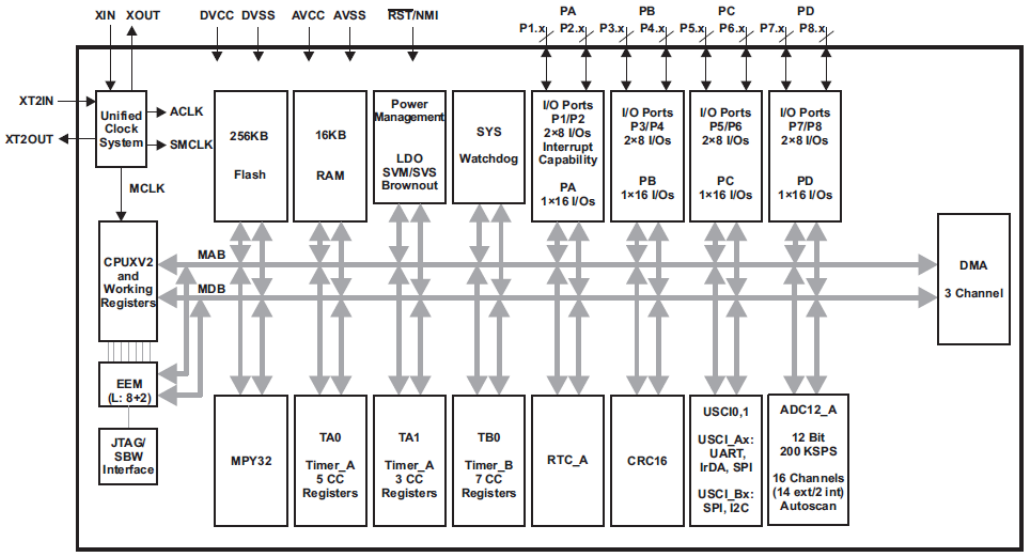

Figure 1.5: *MSP430F5437 Functional Block Diagram*

ADF7020 transceiver

The ADF7020 is a low power, highly integrated transceiver designed for operation in the license-free ISM bands at 433 MHz, 868 MHz, and 915 MHz, supporting different kinds of modulations (FSK, GFSK, ASK, OOK and GASK). A complete transceiver can be built using a small number of external discrete components,making the ADF7020 very suitable for price-sensitive and area-sensitive applications. The transmit section contains a VCO and low-noise fractional-N PLL. The VCO operates at twice the fundamental frequency to reduce spurious emissions and frequency pulling problems [34].

The transmitter output power is programmable in 0.3 dB steps within a range that goes from -16 dBm to +13 dBm. The transceiver RF frequency, channel spacing, and modulation are programmable using a simple 3-wire interface (in our case we used SPI bus for communication). The device operates with a power supply range of 2.3 V to 3.6 V and can be powered down when not in use. A low IF architecture is used in the receiver (200 kHz), minimizing power consumption and avoiding interference problems at low frequencies.

The ADF7020 supports a wide variety of programmable features including Rx linearity, sensitivity and IF bandwidth, allowing the user to trade off receiver

sensitivity and selectivity against current consumption, depending on the application. The receiver also features a patent-pending frequency control (AFC) loop, allowing the PLL to track out the frequency error in the incoming signal.

1.3.2 Tyndall Sensor Layers

Tyndall National Institute has a good experience in designing miniaturized wireless sensors deployed as wearable systems, which can provide a real-time constant patient monitoring by collecting vital parameters like blood pressure and pulse rate. In this thesis, three different kinds of sensors have been used to set up the network: accelerometer, electrocardiography sensor (ECG) and pulse-oximeter (SPO2).

Each sensor layer is equipped with a battery, in order to be attached directly to the transceiver layer and transmit data. Thus, no additional "power layer" is required: this feature is very useful since it renders easier the placement of the sensors on the patient body. It is possible, for example, to have an accelerometer device mounted on the waist (packed in a belt) or on the wrist (packet in a bracelet), which measures the movements of the patient and sends data.

Accelerometer board

This board is equipped with the Analog Devices sensor ADXL330 [34]. The ADXL330 is a small, thin, low power, complete three-axis accelerometer with signal conditioned voltage outputs, all on a single monolithic IC, measuring acceleration with a minimum full-scale range of ±3 g. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The ADXL330 core is a polysilicon surface micro-machined sensor and it contains a signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analogue voltages that are proportional to acceleration: normal voltage values range for zero bias outputs is from 1.2V from 1.8V (typical 1.5V). So the reference voltage of the internal microcontroller's ADC has to be chosen in according to these values.

The sensor is a built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180°out-of-phase square waves.

Acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration, and demodulator output is amplified and brought off-chip through a 32 kΩ resistor.

The ADXL330 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axes sense directions are highly orthogonal with little cross axis sensitivity. Mechanical misalignment of the sensor die to the package is the main source of cross axis sensitivity. Mechanical misalignment can, of course, be calibrated out at the system level.

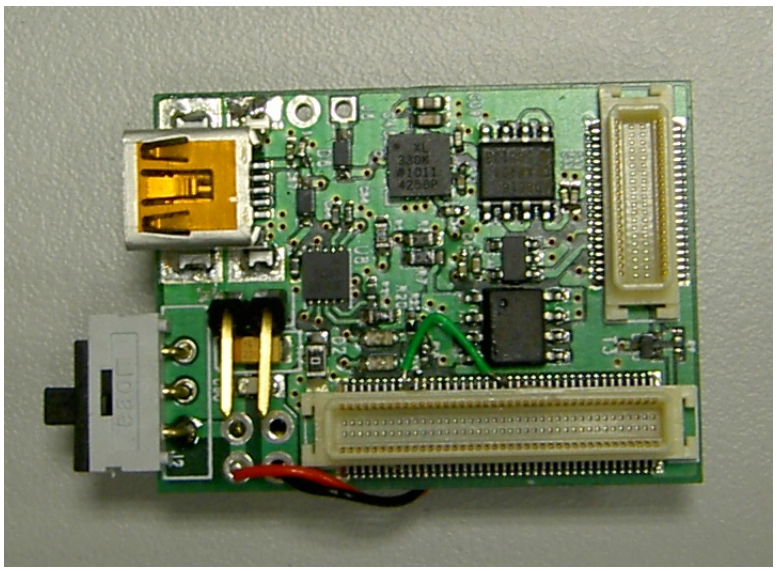

Figure 1.6: *Tyndall accelerometer sensor for MSP430/ADF7020 layer*

Pulse-oximeter (SPO2) board

A pulse oximeter is a medical device that indirectly monitors the oxygen saturation of a patient's blood and changes in blood volume in the skin: it gives the amount of oxygen being carried by the red cell in the blood.

The sensor has a pair of small LEDs placed into the gray rubber connector plugged in the PCB. One LED is red, with wavelength of 660 nm, and the other is infrared, with wavelength of 905, 910, or 940 nm.

The medical reason of this choice is that the absorption at these wavelengths differs significantly between oxyhemoglobin and its deoxygenated form; therefore, the oxy/deoxyhemoglobin ratio can be calculated from the ratio of the absorption of the red and infrared light.

The monitored signal coming from each LED oscillates synchronously with the heart beatbecause the arterial blood vessels expand and contract with each heartbeat. By examining only the varying part of the absorption spectrum, a monitor can ignore other tissues or nail polish, and discern only the absorption caused by arterial blood.

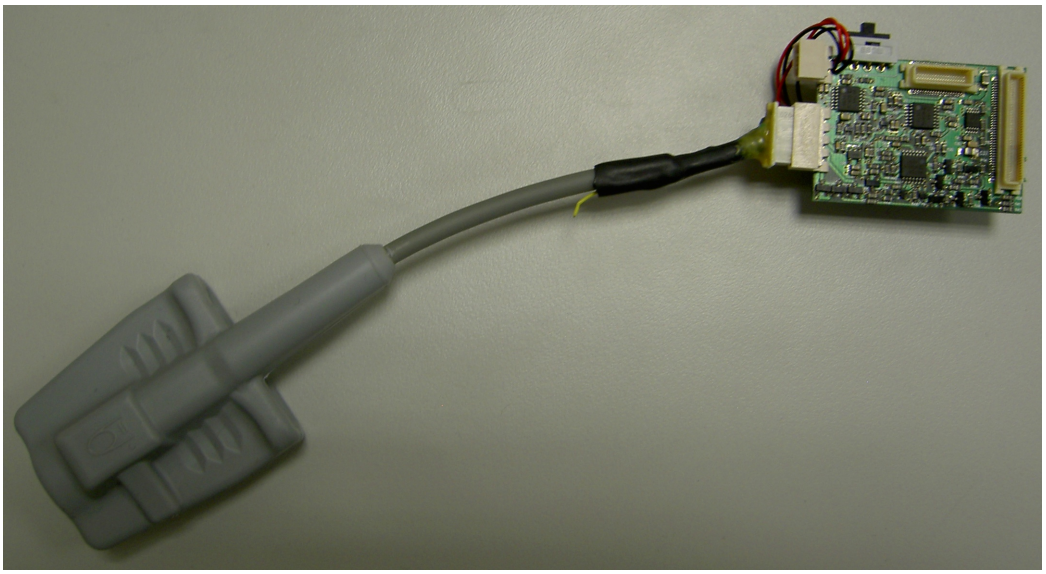

Figure 1.7: *Tyndall SPO2 sensor for MSP430/ADF7020 layer*

Chapter 2

Choice of respirations rate detection algorithm

2.1 Data description

2.1.1 PPG signals and pulse oxymetry

The pulse oxymetry is a non-invasive method that allows monitoring the patient's hemoglobin oxygenation. The medical device that indirectly monitors the oxygen saturation of a patient's blood (as opposed to measuring oxygen saturation directly through a blood sample) is called pulse oximeter, and it measures changes in blood volume in the skin, producing a photoplethysmograph; most monitors, moreover, also display the heart rate. The original oximeter was made by Milliken in the 1940s. The precursor to today's modern pulse oximeters was developed in 1972, by Aoyagi at Nihon Kohden using the ratio of red to infrared light absorption of pulsating components at the measuring site. It was commercialized by Biox in 1981. The device did not see wide adoption in the United States until the late 1980s (source: www.wikipedia.org).

Principles of Pulse Oximetry Technology. The principle of pulse oximetry is based on the red and infrared light absorption characteristics of oxygenated and deoxygenated hemoglobin: oxygenated hemoglobin absorbs more infrared light and allows more red light to pass through; on the other side, deoxygenated

(or reduced) hemoglobin absorbs more red light and allows the infrared component to pass through. This physical effect is based on a frequency (or wavelength) difference between the red light and infrared light: the red one has a wavelength band extending from 600 to 750nm, the infrared one extends from 850 to 1000nm.

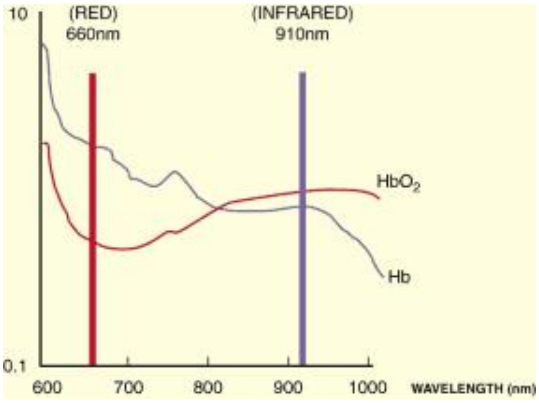

Figure 2.1: *Red and infra-red frequencies*

Pulse oximetry requires a reasonably translucent site with good blood flow; typically, a sensor is placed on a thin part of the patient's body, usually a fingertip or earlobe, or in the case of an infant, across a foot. A light containing both red and infrared wavelengths is passed from one side to the other where, at the opposite side of the emitter, there is a photodetector that receives the light that passes through the measuring site.

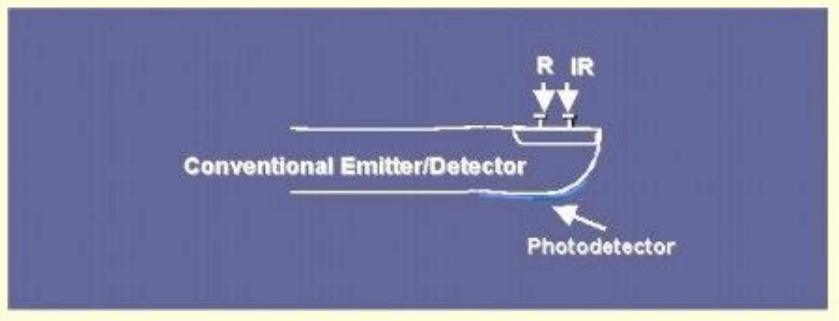

Figure 2.2: *Example of application*

After the transmitted red (R) and infrared (IR) signals pass through the measuring site, changing in absorbency between the red and infrared light caused

by the difference in color between oxygen-bound (bright red) and oxygen-unbound (dark red or blue, in severe cases) is measured, allowing determination of the absorbencies due to the pulsing arterial blood alone, excluding venous blood, skin, bone, muscle, fat, etc. The R/IR ratio is calculated and is compared to a "look-up" table (made up of empirical formulas) that converts the ratio to an SpO2 value. Most manufacturers have their own look-up tables based on calibration curves derived from healthy subjects at various SpO2 levels. Typically a R/IR ratio of 0.5 corresponds to approximately 100% SpO2, a ratio of 1.0 to approximately 82% SpO2, while a ratio of 2.0 equates to 0% SpO2.

This method based on sending the light through the measuring site is called "**transmission**" and is the most common solution for recording a PPG signal. However, it is possible to obtain a PPG signal using a second method (called "**reflectance**"), for sending the light through the tissues of the body. In this solution, as shown in the figure below, the emitter and photodetector are next to each other on top the measuring site and the light bounces from the emitter to the detector across the site. Apart this, the measure of oxygenation is made in the same way and is based on the same physical principle.

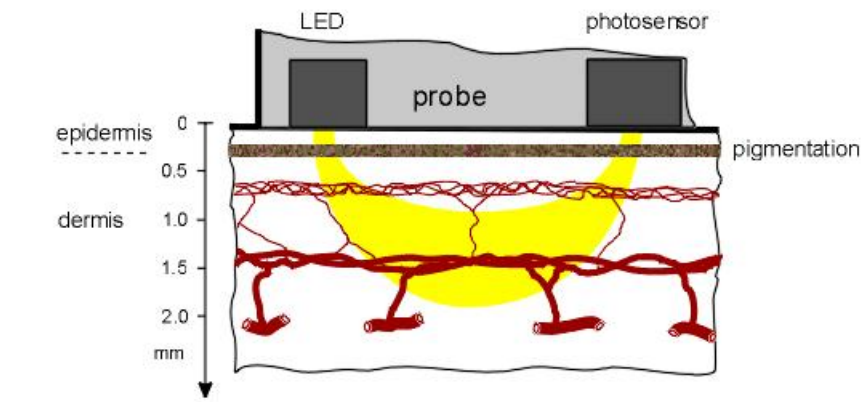

Figure 2.3: *Reflection mode*

This method however, presents some higher sensibility to disturbs if compared with the transmission one. In particular, we can observe higher sensitivity to motion artifacts and to noises (electrical, light interference, etc.).

For these reasons, the transmissive technology is often preferred to the reflective one and more used in practice.

A common feature for a PPG signal is a remarkable DC component: indeed, regardless of the measuring site, there are constant light absorbers like skin, tissue and venous blood, that are always present. However, with each heart beat the heart contracts and there is a surge of arterial blood, which momentarily increases arterial blood volume across the measuring site; this results in more light absorption during the surge. If light signals received at the photodetector are looked at 'as a waveform', there should be peaks with each heartbeat and troughs between heartbeats. If the light absorption at the trough (which should include all the constant absorbers) is subtracted from the light absorption at the peak then, in theory, the resultants are the absorption characteristics due to added volume of blood only, which is arterial. Since peaks occur with each heartbeat or pulse, the term "pulse oximetry" was coined. This solved many problems inherent to oximetry measurements in the past and is the method used today in conventional pulse oximetry.

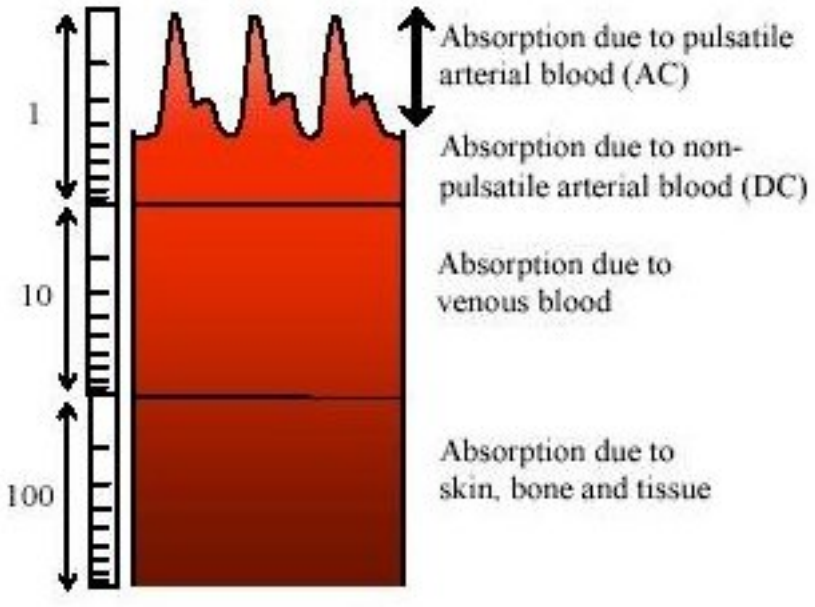

Figure 2.4: *AC and DC components for a ppg signal*

In conventional pulse oximetry, noises like motion artifacts, low perfusion (weak signal), and electrical noise, can alterate dramatically the signal nature, making difficult to depend only on the PPG, when taking medical decisions. Arterial blood gas tests have been, and continue to be, commonly used to supplement or validate SpO2 readings. The "Next Generation" pulse oximetry technology, has demonstrated significant improvement in the ability to read through motion and low perfusion, making pulse oximetry more reliable for medical decisions. Figure (2.5) shows some examples of PPG signals distorted by different noises; as we can see, the signal becomes unrelaible for extracting information.

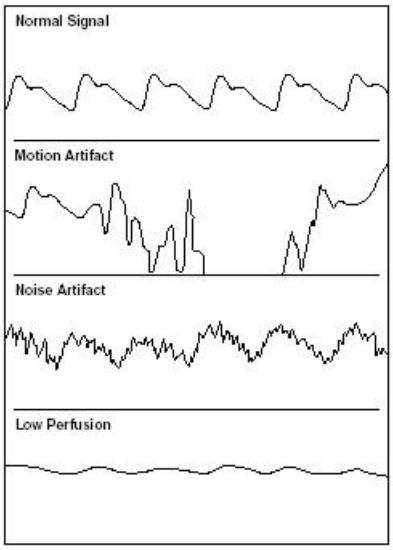

Figure 2.5: *Example of noises for a PPG signal*

2.1.2 Accelerometer signal

A tri-axial accelerometer is a device that measures the acceleration in three orthogonal directions (sensing axes). An accelerometer can be used to sense vibrations (e.g. the vibration of a machine), orientation (e.g. in human activities monitoring) and inertia. There are many kind of devices commercially available, each based on a different technological solution, but all rely on the same theoretical base.

The theory behind acceleromyography is based on Newton's second law of motion: when mass is constant, acceleration is directly proportional to force:

$$F = m \cdot a$$

The most common kinds of accelerometer used in embedded biomedical applications are:

- Piezoelectric: a device that utilizes the piezoelectric effect of certain materials to measure dynamic changes in mechanical variables such as acceleration, but also vibration and mechanical shock.
- Capacitive: capacitive accelerometers sense a change in electrical capacitance, with respect to acceleration, to vary the output of an energized circuit. The sensing element consists basically of two parallel plate capacitors, acting in a differential mode.

Modern devices can measure acceleration in all the three directions of the space (so called tri-axial accelerometers); for every axis, it is possible to obtain a signal like the one shown in the following figure:

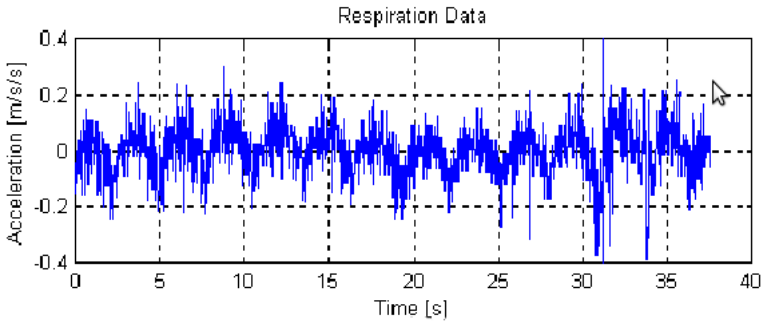

Figure 2.6: *Example of one axis accelerometer signal*

In order to estimate the respiratory rate, it is possible to observe that accelerometer's sensitive axes are contained in the sagittal plane, with an axis along the anteroposterior direction, and a second one along the vertical direction (Fig. 2.7). The figure below illustrates the accelerometer measurements when the person is in either vertical or horizontal position.

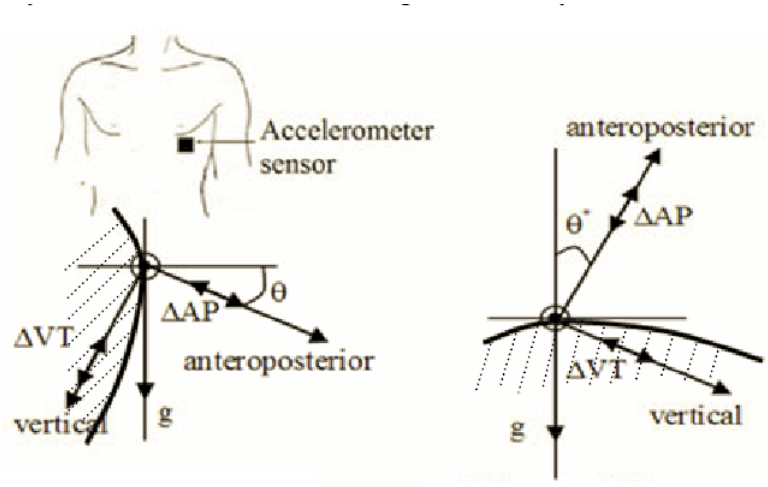

Figure 2.7: *Sensor module location: vertical (left) and horizontal (right)*

The normal breathing activity causes a periodical movement of thorax and thus it changes the chest-accelerometer's inclination. It is possible to detect this movement (or inclination, see paragraph 2.2.3) in order to estimate the RR. Moreover, as the breathing signal is principally visible along the direction perpendicular to the gravity, which is the most sensitive direction for measuring the thorax movement, it is quite common practice to use the sum of both accelerometer axes to obtain data independent of the subject's posture.

Remarkably, the employment of an accelerometer also allows to observe the heart vibration, that is "visible" along the anteroposterior axis and in both postures (vertical and horizontal).

2.2 State of the art for algorithms

2.2.1 Algorithms for estimating RR using a SpO2 sensor

Pulse oximetry is frequently used in clinical situations for non-invasive measurement of heart rate and arterial oxygen saturation. It has been suggested that signal processing techniques can be used to extract the respiratory rate from the photoplethysmogram (PPG), which is the pulsating waveform produced by a

pulse oximeter at one of its two wavelengths (red and infra-red). If this is possible, it would allow non-invasive measurements of breathing rate using a device (the pulse oximeter), that is already used in many clinical situations, and is known to cause a minimum of distress or inconvenience to the patient.

A number of methods for deriving the breathing rate from the PPG have been suggested in the literature: in this paragraph, we are going to evaluate pros and cons for the most widely used techniques in this field. A landmark work for this analysis has been the paper presented by S. Fleming et al.[1] in 2007.

1) Digital filtering approaches

The simplest approach found in literature is based on digital filtering techniques, as proposed by Nilsson et al [2],[3],[4]. This method exploits the possibility of distinguishing two different fluctuations in the PPG waveform: one at higher frequency (PPGc), corresponding to the heart rate waveform, and a second one, that reflects a variation synchronous with respiration (PPGr).

By properly filtering the data, it is possible to extract the respiratory rate harmonic from the filtered signal. Nilsson suggested the use of a 3^{rd} order Butterworth band-pass filter with a pass-band from 0.1Hz to 0.3Hz (6 to 18 breaths per minute), with a "visual" analysis of the results, but it is also possible to obtain the RR using a spectrum analysis (e.g. detecting the spectrum peak).

According to the Fleming's analysis, the Nilsson's method was found to perform well, with an average error of less than 0.5 breaths per minute when compared to the reference rate. A remarkable advantage of this technique is its simplicity so, it seems to be particularly suitable for real time applications.

Despite this, the main drawback is that the accuracy of this technique degrades with motion artifacts, which are especially prevalent in the PPG signal during exercise. These kind of noises can lead to completely wrong results in the RR estimation, thus, the application of this kind of solutions has to be restricted to signal recording sections where the patient is laying down or seated.

Filtering is also used in the method suggested by Nakajima et al.[5]. In this work, the PPG is initially filtered with analogue low- and high-pass filters with

cut-off frequencies at 0.1 and 5Hz. The breathing signal is then obtained using three different low-pass filters, with cut-off frequencies at 0.3, 0.4 and 0.55Hz. The choice of the filter is determined by the heart rate (or pulse rate), thus, a higher cut-off frequency is used at a higher heart rate, and hysteresis is employed to reduce the rate at which filters are switched.

According to the results found in literature, despite the increased complexity of this algorithm, it does not seems to perform better than the other digital filtering techniques considered. In the Fleming's study [1], it has been found an average error of over 3 breaths per minute when compared to the reference rate.

Moreover, this solution presents the same accuracy problems due to motion artifacts in the original waveform so, in order to use this algorithm, it is necessary to apply the same "restriction" for the application field.

2) Auto-regressive model

In statistics and signal processing, an autoregressive (AR) model is a type of random process which is often used to model and predict various types of natural and social phenomena.

This technique has been applied to a number of other physiological signals, including the EEG and the intrapartum cardiogram, but it has been applied to the problem of extracting breathing rate information from the PPG waveform only recently[1],[6].

Basically, AR modelling can be formulated as a linear prediction problem, where the current value $x(n)$ can be estimated as a linearly weighted sum of the preceding P values. The parameter P is the model order, which is usually much smaller than the length of the data sequence (N).

$$x(n) = -\sum_{k=1}^{P} a_k \cdot x(n-k) + e(n) \tag{2.1}$$

The signal $e(n)$ is white noise, which is assumed to be normally distributed with zero mean and variance σ^2, and is used to model the error committed on the prediction.

Thus, an autoregressive model can be viewed as the output of an all-pole infinite impulse response filter (IIR) where the input signal is white noise.

Working in the digital domain, every sample of the output $x(n)$ is therefore a linear regression on the signal itself using P previous values; these ones are weighted by the coefficients a_k, also called *parameters* of the model.

Some constraints are necessary on the values of these parameters, in order to be sure that the model remains wide-sense stationary (WSS), i.e. only the process mean and autocorrelation do not change when shifted in time (in a strongly stationary process instead, this is true for all the statistic functions). As result of this condition, we have: $m_X(t) = m_X$ for the mean, and $R(t_1, t_2) = R(t_2 - t_1)$ for the autocorrelation.

In general, for a p-order AR model to be wide-sense stationary it is required that the roots of the polynomial:

$$z^p - \sum_{k=1}^{p} a_k \cdot z^{p-k}$$

lie within the unit circle. In other words, each root z_k must satisfy the inequality $|z_k| < 1$.

The problem of AR modelling however, can also be visualized in terms of a system with input $e(n)$, and output $x(n)$. In this case the transfer function $H(z)$ can be written as shown below:

$$H(z) = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}} = \frac{z^p}{(z - z_1)(z - z_2) \cdots (z - z_p)} \tag{2.2}$$

As shown in (2.2), the denominator of H(z) can be factorized into p terms. Each of these terms defines a root z_i of the denominator of $H(z)$, corresponding to a pole of $H(z)$; as stated before, the AR model is an all-pole model since $H(z)$ has no finite zeros. The poles occur in complex-conjugate pairs, and define spectral peaks in the power spectrum of the signal, with higher magnitude poles corresponding to higher magnitude peaks. The resonant frequency of each spectral peak is given by the phase angle of the corresponding pole.

Such angle is defined in (2.3), which shows that θ is also dependent on the sampling interval Δt, at frequency f.

$$\theta = 2\pi f \Delta t \tag{2.3}$$

The PPG signal is typically sampled at rates between 50 and 250 Hz to ensure that the shape and heart rate information are preserved. At such high sample rates, the phase angles corresponding to breathing frequencies are very small, which is likely to lead to inaccuracy in identifying the frequency of the breathing pole, or possibly even to the absence of a breathing pole in the AR model. It is therefore necessary to downsample the signal to increase the angular resolution of the low-frequency information. This also ensures that the cardiac-synchronous pulsatile component of the PPG is no longer dominant, as otherwise many or all of the poles will be used to model this signal, rather than the wanted breathing signal. To improve the stability of the AR model, it is also necessary to remove any DC offset from the downsampled PPG.

In the algorithm presented by Flaming and Tarassenko, the PPG signal is therefore downsampled and detrended prior to AR modelling. A decimation algorithm, which filters the signal prior to resampling, is used to reduce the effect of aliasing in the downsampled signal.

A range of angles is defined by the expected breathing frequencies for a normal subject, and the poles with phase angles within this range are identified as possible breathing poles. For models with p>3, multiple poles may be identified; as the range of possible breathing frequencies is quite large, most other signals have to be removed by the filtering and downsampling steps. The choice of pole is made using the magnitude of the poles, as poles corresponding to breathing should have a high magnitude.

The pole with the highest magnitude in the sector of interest is identified, and a threshold of 95% of the highest magnitude is used to determine candidate poles in the breathing range. Among these candidates, the pole with the smallest angle (or lowest frequency) is identified as the breathing pole. This requirement is introduced because poles at a multiple of the breathing frequency occasionally

occur with a slightly higher magnitude than that of the true breathing pole, leading to incorrect results.

3) Wavelet decomposition

This "family" of solutions has been studied by Addison et al [7],[8],[9],[10]. Wavelet transforms are a family of relatively new signal processing strategies that enable time-frequency unfolding of time-domain signals. These techniques have particular advantages in comparison for example, to Fourier-based techniques, for focusing upon, and extraction of, pertinent features within ambiguous data sets. The wavelet transform of a signal $x(t)$ is defined as:

$$T(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \cdot \psi^* \left(\frac{t-b}{a} \right) \, dt \tag{2.4}$$

where $\psi^*(t)$ is the complex conjugate of the wavelet function $\psi(t)$, a is the dilation parameter of the wavelet and b is the location parameter of the wavelet. The time-scale representation given by Equation (2.4) can be converted to a time-frequency representation, where the characteristic frequency associated with the wavelet is inversely proportional to the scale a. By interrogating the wavelet transform of the PPG signal, we can extract breathing information.

The energy density function of the wavelet transform, or *scalogram*, is defined as:

$$|T(a,b)|^2$$

where "||" is the modulus operator. One common rescaling of the scalogram is defined as:

$$\frac{|T(a,b)|^2}{a}$$

and is useful for defining ridges in wavelet space when, for example, the Morlet wavelet is used. In the paper presented by Addison[7], ridges are defined as the locus of points of local maxima in the plane; however, any reasonable definition of a ridge may be employed in the method.

For practical implementation requiring fast numerical computation, the wavelet transform may be expressed in Fourier space and the FFT algorithm employed.

However, for a real time application the temporal domain convolution expressed by equation (2.4) may be more appropriate.

In a wavelet-ridge based method, the pertinent repeating feature in the signal gives rise to a time-frequency band in wavelet space or a rescaled wavelet space; for a periodic signal, this band remains at a constant frequency level in the time frequency plane.

For many real signals, especially biological signals, the band may be non-stationary, varying in characteristic frequency and/or scale over time. Figure (2.8) shows a schematic of a wavelet transform of a signal containing two pertinent components leading to two bands in the transform space; these bands are labeled band A and band B on the 3-D schematic of the wavelet surface. We define the band ridge as the locus of the peak values of these bands with respect to frequency. We can assume for example, that band B contains the signal information of interest and we call it the "primary band". In addition, we assume that the system from which the signal originates, and from which the transform is subsequently derived, exhibits some form of coupling between the signal components in band A and band B.

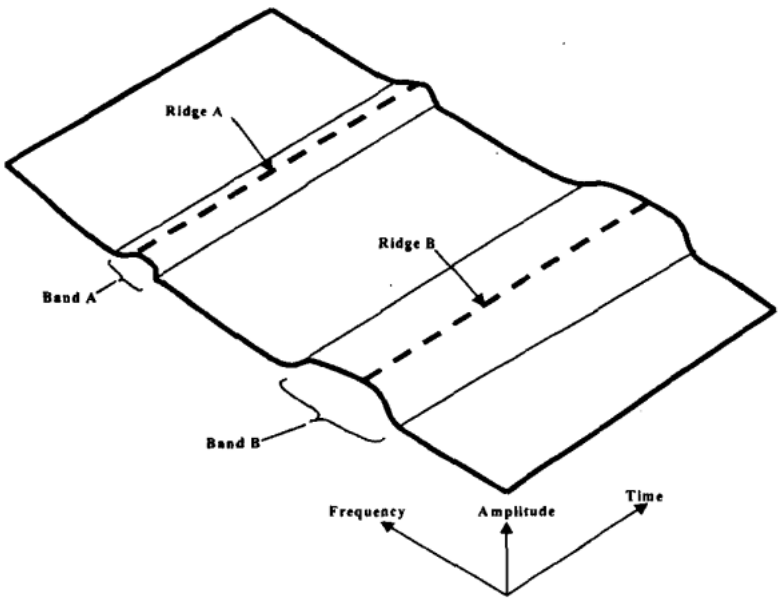

Figure 2.8: *3-D Schematic of a wavelet transform surface containing two bands*

When noise, or other disturbances, are present in the signal with similar spectral characteristics of the features of band B, the information within band B can become obscured. In this case, the ridge of band A can be followed in wavelet space and extracted, either as an amplitude time-signal or a frequency-time signal, which we call the "ridge amplitude perturbation (RAP) signal", and, the "ridge frequency perturbation (RFP) signal", respectively. Subsequently, a further wavelet decomposition of these newly derived signals, is performed. With this secondary wavelet decomposition, we exploit the coupling between the two bands to extract the information in band B, which is corrupted by noise.

The method described up to now, which is called secondary wavelet feature decoupling (SWFD), allows information concerning the nature of the signal components, causing the primary band B (figure 2.8) to be extracted when band B itself is obscured by the presence of noise or other erroneous signal features.

Hence, up to three breathing signals may be obtained from a single PPG: the original signal, the RAP, and the RFP. In order to extract the breathing frequency, all the three scalograms must be compared, and the optimal one for the extraction of the information required must be determined. In practice, the cleaner of the two scalograms is chosen, looking at the one which significantly does not contain interference from artifacts, noises, etc.

This algorithm can be fully automated[10], in order to develop an embedded application. In this case, respiratory information is extracted from all the transform spaces, which are inspected and ranked in terms of confidence prior to the selection of respiratory components.

According to the results found in literature [1, 12], this method appears to be quite accurate in the respiratory rate estimation, but at the price of an elevated computational difficulty, and high complexity. Moreover, the accuracy of this method seems to became less accurate with the increase in actual breathing rate.

4) Time-frequency approaches

Another family of solutions is represented by all methods that use a joint time and frequency data analysis, in order to extract the signal information.

These algorithm can be very accurate but, according to the literature, very complex at the same time.

In our research, we have found several different algorithms that rely on this kind of analysis. In all of these methods, the aim is to achieve a frequency-over-time representation for the data, regardless of the signal processing carried out on the original signal.

A first example of this kind of technique can be the work presented by Shelley and Awad [11]. In this work they used a short Fourier Transform with a Hann window, in order to analyze the pulse oximeter waveform:

$$X[k, n_0] = \sum_{m=n_0-N+1}^{n_0} (w[m - n_0]x[m])e^{-j(2\pi N)km} \tag{2.5}$$

$$k = 0, 1, 2, \cdots, N - 1$$

$$w(n) = 0.5 - 0.5cos(2\pi n/N - 1)$$

The algorithm presented is composed by the following steps:

1. A windowing function is used on the data in a digital buffer, in order to minimize the effect of the sample set's finite range;

2. A Fourier analysis is performed on the data set in the digital buffer. The data is then expanded in a logarithmic fashion to compensate the, otherwise overwhelming, heart-rate signal strength.

3. The result is transferred to a display buffer, and then, new data are accepted from the pulse oximeter on a first-in, first-out basis. The amount of new data added, is determined by respiratory rate measured up to that point.

4. The resulting data are plotted with Y axis - frequency and X axis- time. A number of different techniques can be used to display the results (false color, gray scale, "waterfall" or as a surface plot).

At this point, by analyzing the joint frequency-time plot, it is possible to extract information like the respiratory rate as well as the heart one.

The main drawback of this technique is that the presence of motion artifacts is proved to be the primary cause of failure for this algorithm. In addition, the presence of random heart beats (i.e. atrial-fibrillation, premature arterial contraction, etc.) disrupts the regularity of underlying waveform created by the pulse, causing inaccuracy of results.

Other algorithms that belong to this category rely on different signal processing techniques in order to obtain a time-frequency spectrum (TFS). According to the literature, the most precise methods are those that implement a variable frequency complex demodulation (VFCDM) for estimation of TFS [12],[13].

The VFCDM method involves a two-step procedure: the first step is to use the complex demodulation (CDM), or what we termed the fixed-frequency CDM (FFCDM) to obtain an estimate of the TFS; the second step is to select only the dominant frequencies of interest, for further refinement of the time–frequency resolution, using the VFCDM approach. In the first step of the VFCDM method, a bank of LPFs is used to decompose the signal into a suite of band-limited signals. The analytic signals, that are obtained from these by using the Hilbert transform, then provide estimates of instantaneous amplitude, frequency, and phase within each frequency band. Therefore, by the combination of the CDM and Hilbert transform, a high TF resolution spectrum and accurate amplitude information can be obtained.

Once the TFS is obtained via the VFCDM method as described before, respiratory rates are determined by extracting the frequency component that has the largest amplitude for each time point at the heart rate frequency band, since this component reflects the FM. This is justified since the FM is a form of fluctuation that is reflected in the sub-frequency band of a carrier wave, which in our case is the heart rate. To determine frequencies (e.g., respiratory rate) associated with these oscillations, the power spectrum of the FM sequence is calculated, and the frequency at which the highest peak occurs is the desired respiratory rate. A time-varying spectral method can be used in lieu of the power spectrum, if the FM time series is non-stationary. The same procedure is also used for extraction of respiratory rate using the AM.

In other words, this method is based on the use of VFCDM to extract FM signals, followed by the power spectrum to extract the respiratory rate accurately.

According to the Chon work [12], this algorithm is the most accurate solution for the estimation of RR rate from a PPG signal, with a computational time faster than the wavelet based algorithm, but slower than the AR model method. Nevertheless, it should be noted that, as indicated in the Chon's paper, the performance of the VFCDM method has been tested against metronome synchronous breathings and not spontaneous ones, with a frequency range extending from 0.2Hz to 0.6Hz (12 to 30 breaths per minute), and only on healthy subjects.

The main drawback that has been observed for the VFCDM technique (as well as for the wavelet one), in both supine and upright positions, is that the detection of respiratory frequency became less accurate with the increase in actual breathing rate. However, for high breathing rates (i.e. higher than 30 breaths per min), the method would at least serve to give a clear indication of deviation from normal frequency, even if it is not able to estimate the frequency itself with very high accuracy.

2.2.2 Features of the measuring sites

In the previous section, we presented some among the most common approaches found in literature to extract the respiratory rate from a PPG signal. Obviously, a photoplethysmographic waveform may be recorded from a few different sites throughout the human body; as all these sites present different features, some of them can be more suitable than others in order to estimate the ventilatory effect on the photoplethysmogram.

However, the aim of our work is to employ a signal recorded from a finger pulse oximeter: in this paragraph we are going to discuss how good is the site chosen for this project in order to estimate the respiratory information.

Usually, SpO_2 sensors can be applied in five parts of the body: finger, forearm, forehead, shoulder and wrist; figure (2.9) shows an example of application.

This analysis is based on two papers found in literature, one by Nilsson[4] and one by Shelley et al[14]. In both works, ventilation effect has been estimated

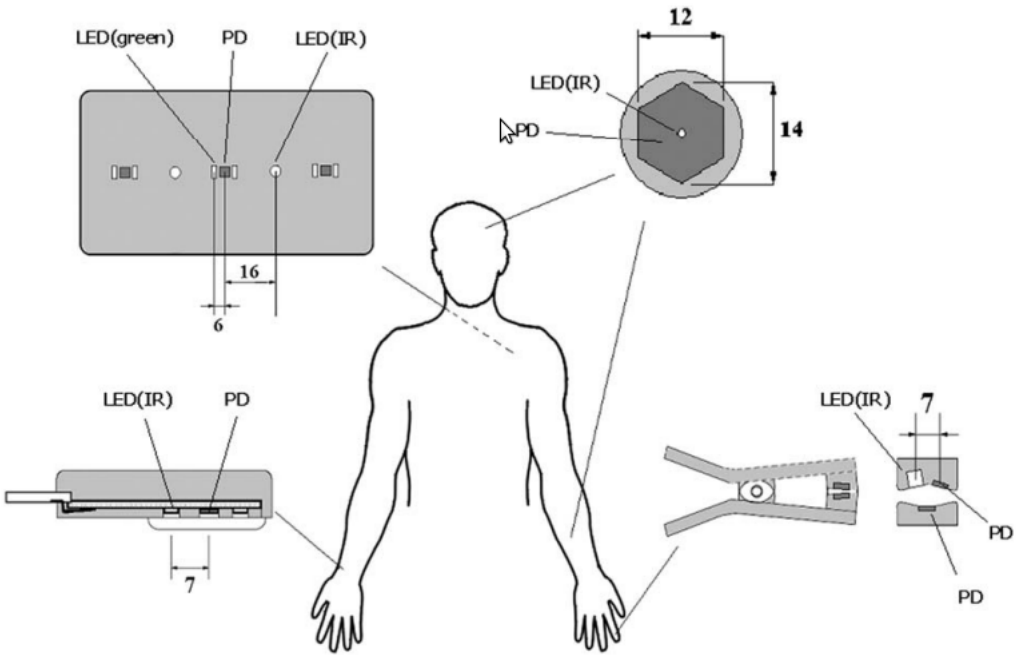

Figure 2.9: *The five photoplethysmographic sensors commonly used*

using different sites for signal recoding, and the results presented seem to match among them.

More precisely, Shelley used three different spots to record the PPG signal: ear, forehead and finger. The respiratory frequency was determined by identifying the peak frequency of the air pressure waveform, and the influence of respiration on each plethysmographic signals was determined by measuring the amplitude of the power spectrum at the respiratory frequency.

On average, the effect of ventilation was expressed 18 times stronger in the ear plethysmograph, when compared with the finger PPG with positive pressure ventilation, and 12 times stronger with spontaneous ventilation. Two factors likely contributed to this finding: first, the shorter distance between head and chest (compared with the distance between finger and chest) means that there is less distance for attenuation of the ventilatory signal within the vasculature.

This attenuation of the signal could come from the dampening that occurs because of the natural elasticity of the vessels, or from the influence of valves within

the venous system. Second, the vasculature of the head is relatively insensitive to local sympathetically mediated vasoconstriction, that may mask respiration-generated oscillations. Figure (2.10) reports an example of differences in power spectrums between different devices (Shelley, [14]).

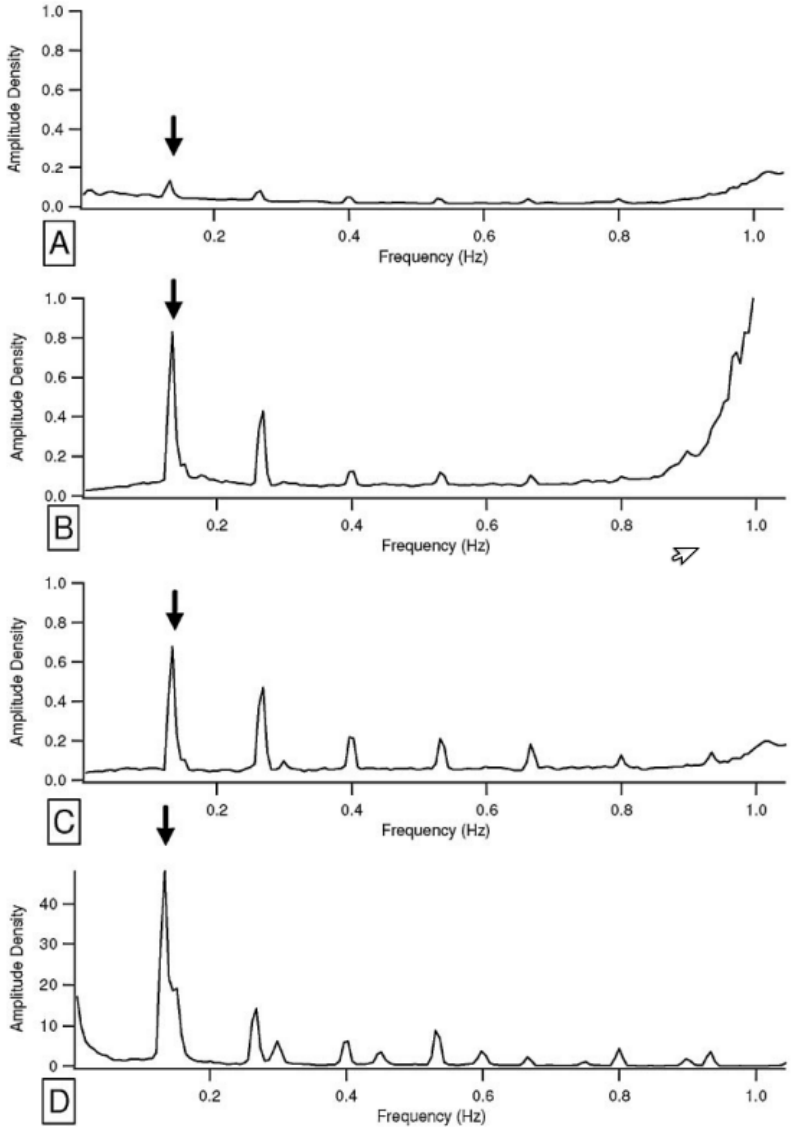

Figure 2.10: *The frequency spectrum of the PPG waveforms A: finger, B: ear, C: forehead and D: airway pressure*

Similar results have been found by Nilsson and his team in [2, 3, 4], but this study exploited five different sensors (figure 2.9).

Nilsson found that respiration- and cardiac-induced variations in the PPG signals could be simultaneously detected at all skin sites investigated. However, analysis showed significant differences for both respiration and pulse detection because, in general, sites having good respiratory variation had poorer cardiac variation, and vice versa. Detection of respiration using PPG was generally more difficult than detection of pulse; this was manifested in the study by lower spectral power content and coherence for respiration than for pulse. Nilsson's results gave no obvious location where best to simultaneously detect variation in PPG signal synchronous with breathing and heartbeat.

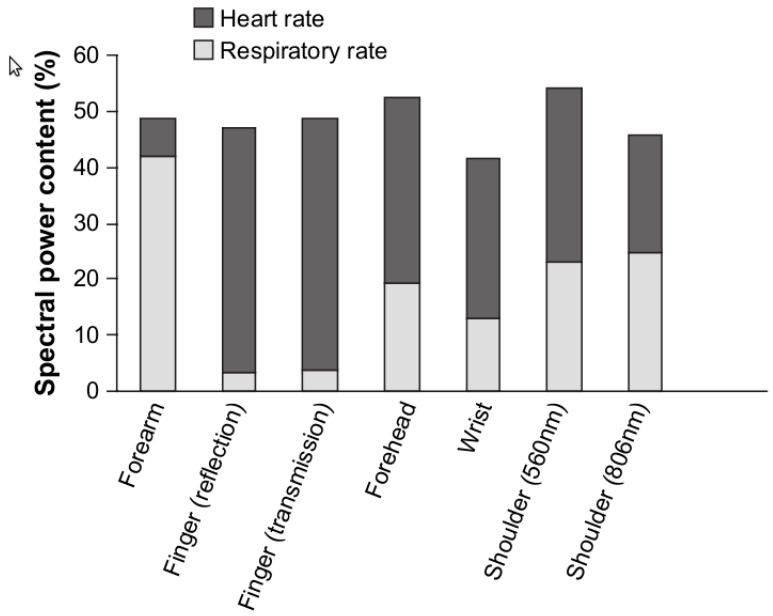

Figure 2.11: *The respiration and cardiac spectral power content derived from the different sensors*

Nevertheless, for our purpose the graph in figure (2.11) is very interesting, since it shows clearly that the finger is not the optimal site for the estimation of the respiratory rate from the plethysmographic waveform. Despite this, since the finger pulse oximeter is the most commonly used patient monitor, both in

and out of the operating room, it has been very important to investigate signal processing techniques capable of evaluating the ventilation effect, in a reliable way, even from a finger- recorded PPG.

2.2.3 RR extraction from accelerometer signal

In literature, it is possible to find plenty of solutions to the problem of extracting the respiratory rate from an accelerometer but, obviously, not all of these have the same accuracy and computational cost.

The simplest solutions are those which rely on digital filtering and spectrum analysis. Due to the nature of the accelerometer signal, the respiratory signal quality depends not only on respiratory effort, but also on body posture and sensor orientation. These are problems that have to be considered during the data collection, in order to obtain correct results.

1) Digital filtering solutions

The simplest technique proposed to extract the respiratory rate through an accelerometer, is to apply a mechanically stretching belt, worn on the patient's chest. This belt could be equipped with different kinds of sensor (usually capacitive MEMS or piezo-electric), in order to measure chest movements during the breathing. In the paper presented by T. Reinvuo, M. Hannula et al. [15], a chest belt with a capacitive accelerometer on board is used (more precisely, two single-axis devices are used in order to compare results).

Collected data were processed with a Butterworth filter (low pass 0.2 Hz, second order) and then, the spectrum of the filtered signal was calculated with a FFT algorithm. In this work, respiratory rates were obtained from the maximum peaks of the obtained signals spectra.

According to the authors, this study has shown that this method is feasible and provides reliable results (80-100% in the solar plexus) thus, it is capable to estimate respiratory rate with good accuracy. Moreover, the computational cost for this algorithm is minimum, making this approach particularly desirable for an

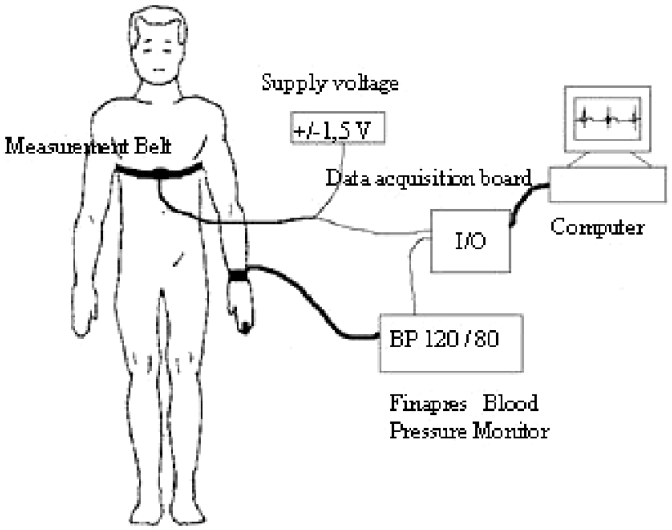

Figure 2.12: *Chest belt application example*

embedded real-time application. Furthermore, the used chest belt arrangement is lightweight, cheap and easy to use, making it suitable for both clinical and home care use.

Due to its sensitivity to movement artifacts however, this approach could be applicable only to immobile or moderately moving subjects. Indeed, presence of motion artifacts in the recorded signal, could lead to erroneous results in the respiratory frequency identification.

A similar approach has been used by Morillo et al. [16]. For this work, a different kind of accelerometer (piezoelectric) has been used, and it has been located between the thyroid cartilage and the superior third of the breastbone (figure 2.13), in order to maximize the average levels of signal power, in different frequency bands (heart, breathing and snore).

To extract the respiratory component, the data recorded from the accelerometer have been firstly downsampled (accelerometers for biomedical applications are often sampled at 50 - 60 Hz) to a frequency closer to the respiratory one. After this, the downsampled signal was low pass filtered with a cut-off frequency $f_u = 0.68$Hz, normalized by the sampling rate. This processing allows to obtain respiratory correlated signals, without need of windowing.

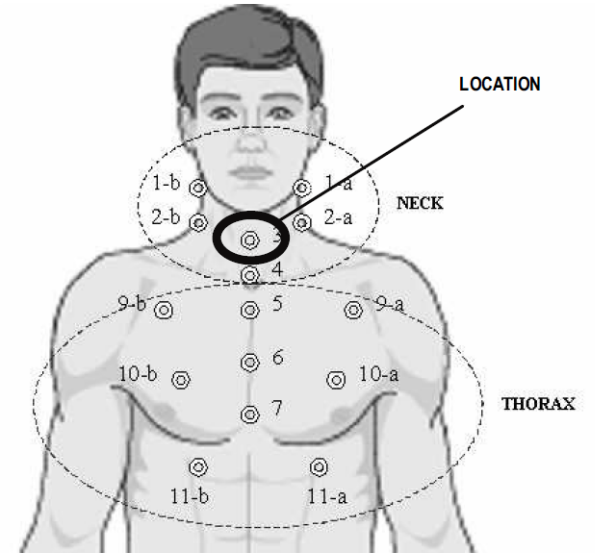

Figure 2.13: *Alternative locations for the accelerometer*

As in the method presented before, Power Spectral Density (PSD) of respiratory component was evaluated, and the respiratory rate has been obtained directly from PSD maximal value.

The algorithm presented in [16] provides fundamental information of the different cardio respiratory variables, useful for the diagnosis of the different types of respiratory abnormal phenomena (apneas, hypopneas and respiratory efforts, heart disorders and respiratory rates), during the sleep or in decubitus position, with good accuracy.

As in the solution presented in [15], this method shows a light computational cost due its simplicity, but it is likely not able to maintain its accuracy when motion artefacts are presents in the signal recorded from the accelerometer, as no remedies against these noises are presented.

2) Adaptative filtering approach

As we have seen up to now, in conventional methods, the respiratory waveform is detected using a fixed-frequency band-pass filter to enhance the measured signal, where the frequency band is usually within the range [0, 1]Hz [15, 16].

In case of deep breathing, it has been observed that the signal could be buried with noise, after such a constant band-pass filter operation.

In contrast with previous approaches, Phan et al. [17]have derived an algorithm that adaptively changes the filter frequency band to optimize the SNR. In order to achieve this goal, they used the sum of both accelerometer axes (in the sagittal plane), in order to be independent of the subject's posture. This algorithm computes the respiration rate as follows:

1. Divide the signal into 1-minute segments (with 10-s overlap);

2. Compute the accelerometer signal spectrum, and detect the dominant frequency f_0 in the range [0.1, 1]Hz. This selection is based on the observation that the number of respiration cycles per minute is between 6 and 60.

3. Band-pass filter the accelerometer signal around f_0 with a Butterworth filter $[f_1,\ f_2]$, 4th order. The choice of the bandwidth is conditioned by the following rules: $f1 = max(0.1, f_0 - 0.4); f_0 = f_0 + 0.4$.

According to results presented by the authors, this algorithm can estimate the RR with accuracy, and could be also used to diagnose or identify, automatically, arrhythmia or respiratory malfunctions. The price to pay in this case, is a higher computational cost and an increased complexity, in comparison with "traditional" approaches. Besides, this method is still sensible to the patient's movements that can occur during recordings. A possible solution to this drawback is the employment of an additional accelerometer that could be applied to the abdomen: the combination of the two devices could be useful to detect body movements, and to differentiate them from those caused by respiratory activity, but at the cost of a further hardware.

3) Techniques based on movements detecting

Another technique relies on the use of a tri-axial device, to detect inclination changes measured regardless of orientation. In a recent paper [18], axis fusion of

the signals from a three axis accelerometer was adopted showing three possible algorithms.

The device was used as an inclinometer to reflect the abdomen or chest movement, caused by respiration. This is based on the fact that the inertial acceleration magnitude is relatively small if compared to the changes of gravitational components. Since the accelerometer is used as an inclinometer, it should be placed on the area where the sensor orientation changes during the respiratory movement. The most important muscle involved in respiration is the diaphragm therefore, the accelerometer is placed roughly at the position of the diaphragm muscle (at the lower end of the sternum).

The raw signal is first collected by the data acquisition module, and fed into the pre-processing block. Preprocessing includes signal conditioning, segmentation, and band-pass filtering. Artifacts such as speech, motion and eating, are detected and removed by pattern classification methods after preprocessing. An essential step before respiratory rate extraction is axes fusion, which aims at reconstructing the original respiration-induced movement signal from the tri-axial signals, given a posture and sensor orientation. In the final step, the respiratory rate is simply calculated by either peak detection in time-domain, or taking the fundamental frequency of the reconstructed signal in frequency domain, when the reconstructed signal is artifacts free.

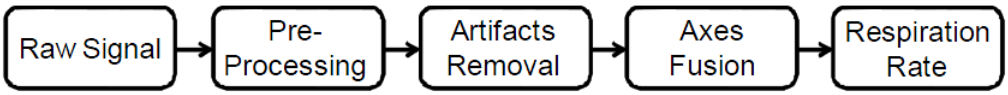

Figure 2.14: *Block diagram of the algorithm*

In this paper ([18]), in order to achieve the axes fusion block, three methods have been investigated:

- Analytical Approach (referred to as Full Angle)
- Principal Components Analysis (PCA)
- Hybrid PCA, the combination of methods 1 and 2.

The focus was on obtaining the highest SNR levels in a combined one-dimensional output signal, and the most effective methods were those based on PCA.

This approach is appropriate if the aim is solely to recover a respiratory rate, but the resulting signals are no longer directly related to a physical quantity. Moreover, it presents the same drawback about the body movements that can occur during the recording session. A good feature of this algorithm, anyway, is that the signal processing phase consists basically of a pass-band filter (axes fusion apart), thus it can be implemented with a computational cost lower than for the previous solution.

The most sophisticated algorithm found in literature has been presented by Bates and al. [19]. In this method, the problem of practical continuous monitoring -in which patient movements disrupt the measurements and the axis of interest changes- is considered, in order to obtain an algorithm robust even against this common limitation. This approach is based on reconstruction of the angular motion induced by breathing, in periods when the patient is otherwise static. When the patient is moving, any accelerometer-based method is probably unfeasible, since the magnitude of the movement-induced signal vastly exceeds that due to breathing.

If the patient is static, the measured and normalized acceleration vector will be close to the acceleration due to gravity g, since the linear accelerations due to breathing are small. As the accelerometer rotates, the gravity vector will rotate in the co-ordinate frame of the device. The axis of this rotation may be arbitrarily oriented in the device frame, and may change due to differences in the way the patient breathes at different times. It can also change with the orientation of the patient. In order to continuously monitor angular motion on the major axis of rotation, a way for finding and tracking this axis as it changes is needed.

Thus, the most remarkable feature of this algorithm is the ability to estimate respiratory rate only when it is sure that the patient is steady and not moving. In order to achieve this goal, a quite complex method for tracking the rotation axis is implemented (not reported here), then the RR is estimated only during

the "clean" time slots recorded. Despite the very good accuracy, this approach does not seem to be particularly suitable for a real time application due to both its complexity and its high computational cost, if compared with the previous techniques.

Chapter 3

Implementation and validation of the algorithms

3.1 Algorithms testing and validation

In this chapter, we are going to discuss the solutions adopted for the two sensors, in order to obtain a low power application capable to estimate the respiratory rate. Two minutes (or longer) data sections, have been recorded for both sensors using a commercially available finger pulse oximeter, and a chest tri-axial accelerometer. Smaller amount of data extracted from these records have been imported in Matlab for algorithms testing and validation.

After this preliminary phase, a first version of C code has been written on a PC using Visual Studio (Microsoft) and tested on the same data for a comparison with the Matlab's results. After all these tests, the code has been ported to the microcontroller MSP430 (Texas Instruments) for the time and power consumption measurements.

3.1.1 Programming techniques

In order to efficiently program the MSP430 microcontroller, some programming-style tips can be taken in order to optimize the trade-off between code execution time and power consumption.

The first thing which should be kept in mind during the development of the code, is the proper setting of the microcontroller clock frequency. About that, there are several ways to proceed: in fact, MSP430 provides 5 low power modes (LPMx) at our disposal.

One way to operate could be to keep the microcontroller in a sleep mode that fits our proposal, then to raise CPU speed at the maximum value when the microcontroller is requested to be active, in order to compute the task as quick as possible, and finally go back to sleep mode once the computation has finished. Thus, one of the first goal of a power-efficient application is to maximize the time spent in low-power modes [27].

Of course, instead of using the maximum CPU frequency, we can use the minimum value that allows us to perform the given task before the time deadline has come (in our case we have to finish the packet transmission within the time slot's duration). About this, MSP430 microcontrollers have a feature named "Dynamic Voltage Scaling (DVS)", that allows programmers to dynamically adapt the processor's supply voltage and operating frequency, in order to just meet the instantaneous processing requirement. Obviously, running at lower speed means to have lower clock rates, and thus lower microcontroller power consumption.

This technique is particularly beneficial for CMOS circuits, as we know that here the power consumption P (and hence the overall energy consumption) is related to CPU frequency and supply voltage by the following relation:

$$P \alpha f V_{DD}^2$$

Anyway, here comes an other problem: in fact one of the main ways to have an energy-efficient application, is to minimize *active program duty cycle*. Trying to achieve this goal, leads inevitably to a trade-off: CPU active time is a direct

function of how many cycles an instruction needs to be executed. By slowing the microcontroller main clock (in general they have more than one clock system), we can reduce instantaneous power, but every operation will require more time to be executed, and this increases active duty cycle. In this way power saving is almost nullified.

One of the best practices, is to use a high main clock frequency, together with DVS technique, when performance is required, and try to move to the peripherals as many functions as possible. The rationales to do this are, in the first instance, that peripherals can be supplied with a clock slower than the main one. Microcontroller CPU is by far the most power hungry circuit on the device, so it is good to "use" it only when necessary. If we are not addressing data or computing but, for example, only listening a channel, we can keep the CPU switched off, and wake it up when needed by using an interrupt service routine. This is the reason why is so important to have a good interrupt-driven code.

A typical example of this is the *UART* reading. It is a pointless waste of energy to keep CPU active only waiting a character from the serial channel: it is better to put microcontroller CPU in sleep mode, and wake it up when a character from serial bus is received (the listening is accomplished by the *UART* module). This trivial example shows that the overall CPU load is reduced about 1000 times!

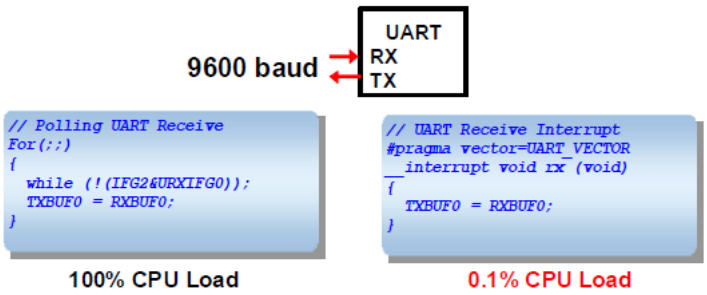

Figure 3.1: *Example of interrupt optimized code*

Anyway, apart from the compiler optimizations, there are some others explicit tips to write code on microcontrollers that can help to achieve an energy-efficient code. For example:

- *Use local variable as much as possible* – Local variables in fact use microcontroller registers, while global variables use RAM, which requires more time for reading/writing operations.

- *Use bit masks instead of bit fields* – (for unsigned int and unsigned char). In the past there was a sort of vicious circle for which programmers used masks because the bitfield code was slow, and complier writers didn't bother much to optimize bitfields because they were rarely used. In general, anyway, bit field can be implemented in a different way by the compliers, so the code is "less portable".

- *Use unsigned data types where possible* – many microcontrollers in general have a lack of operational codes for signed integers. Thus, since a direct instruction set support is not available, the use of a signed integer forces the compiler to use a library function or macro to perform the requisite operation, and clearly this is not very efficient. Moreover, when faced with performing a division by a power of 2 on a signed integer, the compiler has no choice other than to invoke a signed division routine rather than a simple shift operation.

- *Use pointers to access structures and unions* – actually most of the speed is gained only in loops. When you use an array, you would use a counter which is incremented. To calculate the position, the system multiplies this counter with the size of the array element, then adds the address of the first element to get the address. With pointers, all you need to do to go to the next element is to increase the current pointer with the size of the element to get the next one.

- Use *"static const"* class to avoid run-time copying of structures, unions, and arrays.

- *Count down for-loops* – this because comparing the loop index to zero is more efficient than comparing it to some other number.

3.1.2 Pulse oximeter

In order to develop a low power application, we focused our attention on the simplest and most efficient approaches found in the literature. For the SpO2 sensor, such methods are represented by the digital filtering solutions and the AR modeling approach.

Indeed, despite time-frequency spectra techniques, such as the complex wavelet transform (CWT) or the variable frequency complex demodulation (VFCDM), can achieve greater accuracy, they are not suitable for a wireless low power applications. The studies by Leonard, Addison, Watson et al. [7, 8, 9, 10] for example, have shown relatively good results. However, the CWT is impractical because the extraction of respiratory rate is done in some cases with the use of frequency modulation (FM), while in other cases with the amplitude modulation (AM) of heart rate. This requires additional adaptive decision-making schemes to determine whether to use either FM or AM of the heart rate signal, making this kind of approach not suitable for a low power system.

Even the most accurate solution for the estimation of RR, the VFCDM, presents the drawback of being complex even if it is faster than the CWT. Moreover, the same authors reported that its accuracy decreases with the increase in actual breathing rate. For these reasons, it does not appear to be suitable for our application.

Before being processed in Matlab, the recorded data have been firstly divided in smaller 54-seconds long data sections. The SpO2 data have a frequency range which goes from 0.15Hz to 0.65Hz (9 to 39 breaths per minute): a metronome has been used in order to ensure that the whole range of interest was covered.

Since our aim was to minimize the power consumption and computational speed, we firstly have tried a digital filtering approach. The implemented solution, is similar to the method proposed by Nilsson et al. [2, 3, 4] (digital filtering + power spectrum analysis) but, for this work, we used smaller data sections, and we searched the breathing frequencies in a larger range of interest, in comparison with Nilsson's work.

This is the implemented procedure for extracting the RR from the SpO2 sensor:

1. BP filtering (Butterworth 8^{th} order, pass-band extending from 0.15Hz to 0.65Hz
2. Downsampling the filtered signal from 60Hz to 2Hz
3. FFT of the filtered signal and RR estimation

In this case, downsampling has been implemented in order to reduce the overwhelming influence of the heart rate component on the PPG signal. The respiratory rate has been estimated by simply observing the frequency corresponding to the spectrum peak, within the range of interest.

In comparison with the filtering solutions found in literature, it has been necessary to use an higher filter order. This could be probably explained by the difference in the range of breathing frequencies accepted, and the smaller amount of data used.

With this method, it has been possible to determine breathing frequencies with good accuracy, as undesired subharmonics of the HR has been effectively filtered. Table 3.1 reports results for some extracted 54-sec data sections:

Rec	Freq	RR	Matlab results		Error
	(Hz)	(br/min)	(Hz)	(br/min)	(br/min)
1	0.15	9	0.1406	8.436	0.564
2	0.20	12	0.2031	12.186	0.186
3	0.25	15	0.2500	15.000	0.000
4	0.30	18	0.2969	17.814	0.186
5	0.35	21	0.3594	21.564	0.564
6	0.40	24	0.4063	24.378	0.378
7	0.45	27	0.4531	27.186	0.186
8	0.50	30	0.5000	30.000	0.000
9	0.55	33	0.5469	32.814	0.186
10	0.60	36	0.6094	36.564	0.564
11	0.65	39	0.6563	39.378	0.378

Table 3.1: *Digital filtering approach results*

The following figures report an example of signal processing for a single 54-seconds data section (record number 7, sec 96-150):

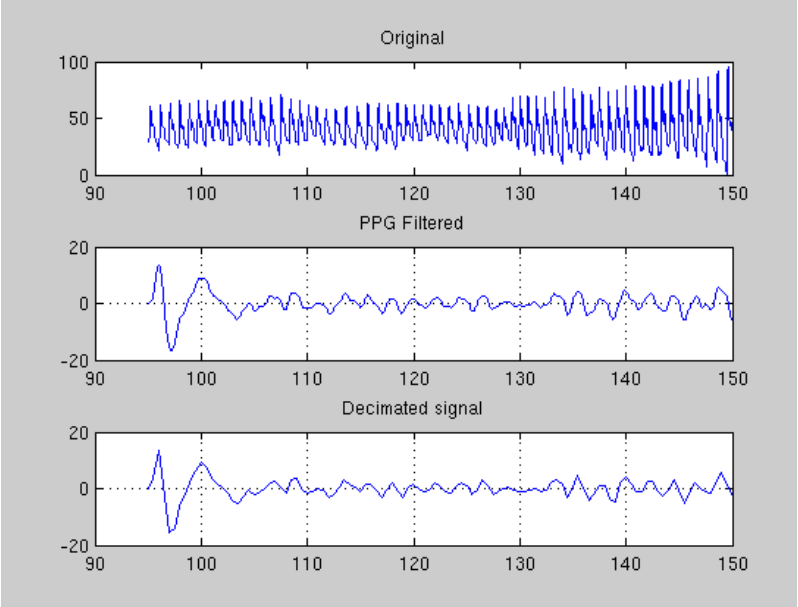

Figure 3.2: *PPG signal row, filtered and downsampled*

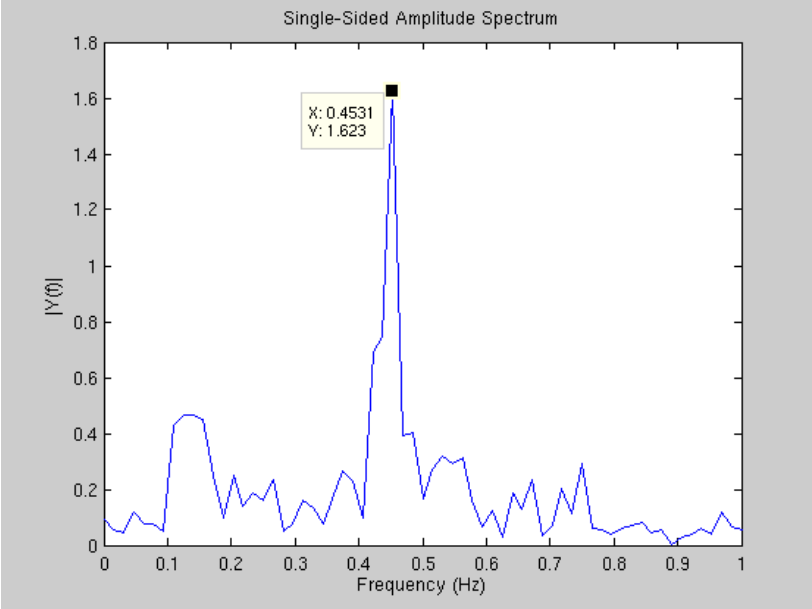

Figure 3.3: *Spectrum of the filtered signal*

Despite these results, the AR approach has been implemented for a comparison as well. The AR-model algorithm has been tested following the same conditions presented by Tarassenko et al. in the related work [1], and it has been carried out using the procedure below:

1. Low-pass filter the PPG row signal: FIR with Kaiser windowing function, transition band extending from 0.4–0.8Hz. The pass-band ripple was designed to be 5%, and the stop-band attenuation was chosen to be 30 dB.
2. Downsampling the filtered signal from 60Hz to 2Hz
3. AR modelling using the filtered and downsampled signal (using Yule-Walker approach for the parameter estimation)
4. AR poles (modules and phases) estimation
5. Determination of breathing frequency choosing the lowest frequency among the poles with higher magnitude (a threshold of 95% of the highest magnitude has been used).

Nevertheless, different filtering techniques and many model orders-downsampling frequencies combinations, has been tried in order to achieve good accuracy. According to the results presented in the paper, the best combination has been found with 9^{th} model order, and downsampling frequency $f_{ds} = 2$Hz.

Results obtained using the same LP filter (even with different parameters) suggested in the paper, were not good because it has not been possible to filter properly the strong DC component in the PPG signal.

Even using the same filter adopted for the digital filtering approach, for some data sections the DC component was still overwhelming, leading to wrong results and big errors in RR estimation (Table 3.2).

Despite the increased complexity, this method does not seem to perform better than the solution presented before. As results show, this approach is too sensible to the powerful DC component typical in a PPG signal, leading to incorrect respiratory rates values (Table 3.2).

Rec	Freq (Hz)	RR (br/min)	Matlab results (Hz)	Matlab results (br/min)	Error (br/min)
1	0.15	9	0.153	9.157	0.157
2	0.20	12	0.196	11.732	0.286
3	0.25	15	0.254	15.225	0.225
4	0.30	18	0.161	9.664	8.336
5	0.35	21	0.159	9.514	11,486
6	0.40	24	0.137	8.219	15,783
7	0.45	27	0.142	8.543	18,457
8	0.50	30	0,157	9.451	20,549
9	0.55	33	0.559	33.579	0.579
10	0.60	36	0.158	9.517	26,483
11	0.65	39	0.639	38.352	0,648

Table 3.2: *AR modelling results*

After this preliminary test phase, we decided to implement the digital filtering approach for our application. The Matlab code has been translated in a first C code version for PC, and the program has been tested further, employing the recorded data used before.

3.1.3 Accelerometer

As we have seen in chapter 2, conventional methods for extracting the respiratory waveform from a chest accelerometer use a fixed-frequency band-pass filter to enhance the measured signal, where the frequency band is usually within the range [0, 1]Hz.

Reinvuo et al.[15], used a chest belt with a capacitive accelerometer on board; collected data were low-pass filtered (Butterworth 2^{nd} ord., f_c=0.2 Hz) and then, respiratory rates were calculated by observing the maximum peak of the filtered signal spectrum. A similar approach has been used by Morillo et al. [16], where data recorded from the accelerometer have been firstly downsampled, then low pass filtered with a cut-off frequency $f_u = 0.68$Hz and normalized by the sampling

rate. For the calculation of the RR, Power Spectral Density (PSD) of respiratory component was evaluated, and the rate has been obtained directly from its maximal value.

These kind of solutions present the remarkable feature of being simple and, at the same time, to require low computational cost. The main drawback, however, is that they are not robust against eventual motion artifacts which are the first cause of inaccuracy.

Recently, more sophisticated techniques have been introduced against this kind of problems: a system of adaptive filters proposed by Phan et al. [17], the use of a tri-axial accelerometer as an inclinometer to reflect the abdomen or chest movement caused by respiration (Jin et al.,[18]), and even a movement detection method, to classify periods in which the patient is static and breathing signals can be observed accurately (Bates et al., [19]).

All these solutions, have good accuracy and greater resistance against motion artefacts. Despite this, the require more computational effort in order to achieve these conditions thus, as we are trying to reduce power consumption and computational speed, we have adopted a conventional approach for our application.

In particular, a procedure similar to the SpO2 one has been used also for the chest accelerometer:

1. FIR low pass filtering, f_c=1Hz
2. Downsampling from 60Hz to 2Hz
3. BP filtering (Butterworth 8^{th} order, pass-band extending from 0.1Hz to 1.0Hz
4. FFT of the filtered signal and RR estimation

As in the previous algorithm, RR has been estimated simply observing the frequency corresponding to the spectrum peak in the range of interest.

Due to the nature of the accelerometer signal, the extraction of RR in this case is less tricky, so we could afford to use smaller data sections, only 45-seconds

long; and collect data with a wider range of breathing frequencies: for this sensor the range of interest ranges from 0.15Hz to 1.0Hz (9 to 60 breaths per minute).

An important issue that has to be considered using an accelerometer, is the patient's posture kept during the recording. Usually, in order to achieve a posture-independent respiration monitoring system, it is necessary to perform a kind of "axes fusion", that returns a signal composed by the three different axes signals, and that allows to extract RR regardless the user's position. Axes fusion can be achieved in several different ways, from the simple sum of vectors, to more complicated methods(such as full angle, PCA, etc.) [17, 18, 19].

For this work we did not perform any axes fusion method, because we assumed to maintain the sited position during the data recording. As the breathing signal is principally visible along the direction perpendicular to the gravity, the only axis employed for this algorithm was the z one, since it is the most sensitive direction for measuring the thorax movements. In a future development, an axises fusion method could be implemented in order to improve the application.

Despite these limitations, it has been possible to estimate the respiratory rate with good accuracy, as results show. Table 3.3 reports results for some 45-sec data sections extracted from data recorded previously:

Rec	RR	Matlab results		Error
	(br/min)	(Hz)	(br/min)	(br/min)
1	14	0.2344	14.064	0.064
2	13	0.2188	13.128	0.128
3	33	0.5469	32.814	0.186
4	25	0.4219	25.314	0.314
5	58	0.9688	58.128	0.128
6	8	0.125	7.500	0.500
7	28	0.4688	28.128	0.128
8	24	0.4063	24.378	0.378

Table 3.3: *Accelerometer results*

Indeed, although it would be possible to estimate the heart rate using a chest accelerometer [17], the extraction of RR with this sensor has been easier and it has been necessary to filter the signal mostly to remove noise, rather than suppressing unwanted HR subharmonics. The following figures report an example of signal processing for a single 45-seconds data section (expected frequency = 0.55Hz):

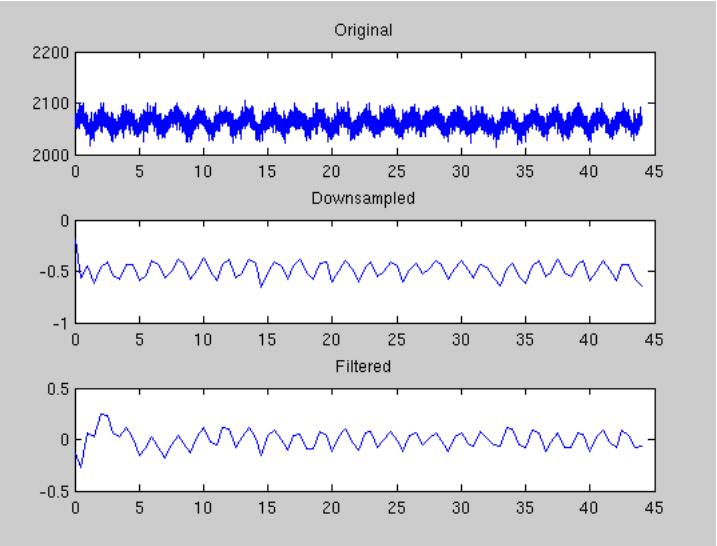

Figure 3.4: *Accelerometer signal row, downsampled and filtered*

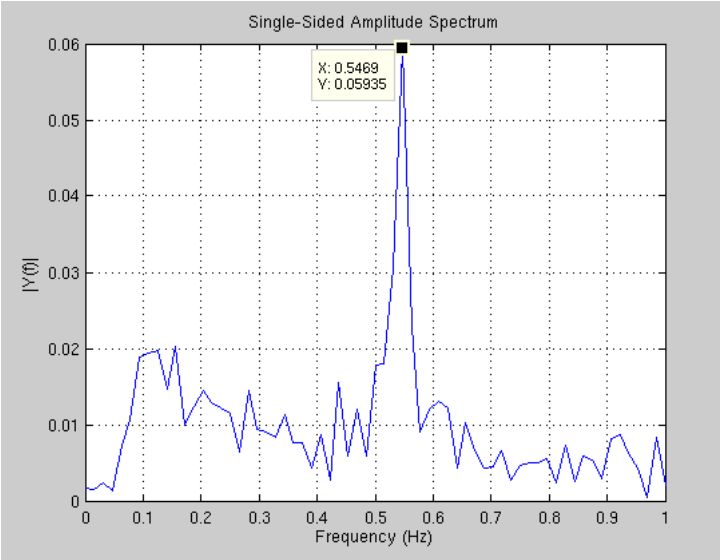

Figure 3.5: *Spectrum of the filtered signal*

3.2 MSP430 C implementation and evaluation

For the embedded implementation of the algorithms under discussion, we used the microcontroller model F5437, of the MSP430 family. This device provides to the user 16 kB of RAM and 256 kB of flash memory, and very few intrinsic functions for signal processing. Thus, for our work, we have fully developed the C code at first on a PC and then, we have ported it into the microcontroller memory. As the two sensors require different amount of data and different solutions, the obtained results show that the pulse-oximeter version has been, by far, the most critical.

3.2.1 Pulse oximeter code

The code has been implemented following the procedure of digital filtering introduced in section 3.1.2. The most important routines that compose the code are: the initial IIR filter, divided in four second-order stages (since it is an 8^{th} order Butterworth filter), where the coefficients has been imported from the Matlab "*Fdatool*"; and the FFT routine, that have been used for the amplitude spectrum modules estimation. We applied FFT routine to the filtered and downsampled signal, so there was no imaginary part as input.

The output is another integer-type vector, made by the 128 FFT modules obtained as sum of FFT real and imaginary parts, squared. No square root has been applied to the modules in order to make the code faster, but an opportune data normalization has been necessary, so as to avoid overflow problems.

The downsampling as been performed between the two routines calls, and it has been implemented simply with a "for" cycle. No anti-aliasing filter has been applied, because the data were already in the correct frequency range after the IIR filtering.

As discussed in the previous section, in order to achieve good accuracy, we had to process 54 seconds of sampled data at every calculation. Data coming from

the sensor can be safely stored in a 16 bit (per sample) register. However, given that to perform the computation, all the information have to be quantized and normalized; the size of the elements in the actual data buffer used for calculations was 32 bit. This size for the values was necessary because, with 16 bits data, we could not reach the desired precision.

By some tests carried out with lower precision, in fact, we observed that the estimation became no more reliable, with completely wrong results. This difference is likely due to the filter coefficients quantization error: with such a low precision, we are probably moving poles in the filter frequency response, obtaining different results from the expected.

So, since the data sampling rate of the sensor is 60 Hz and we are working with 32 bits data version, we must store 3240 32bit signed integer values in RAM. Thus, at the microcontroller initialization moment, almost 13 kB are allocated, and the remaining space in memory is used to allocate other working variables like FFT and IIR arrays. In order to save space in memory, at every filter stage, the main data array is overwritten with new elements: every time we perform a new filter stage, the data vector is overwritten with the filter stage output.

The overwriting of the sampled data, together with the full RAM memory, can represent a problem in order to make our implementation real-time. In this case, we need to refresh the computed RR value after a certain amount of time, overwriting the oldest n samples, and replacing them with the new ones coming from the sensor. The actual implementation on the used hardware does not leave enough space in memory to save the sampled sensor data for the RR estimation of the next time slot, because after the first stage, the original sensor data are overwritten. Therefore, a processor with a slightly bigger memory is required in order to develop a proper real-time application.

All the multiplications in the code are computed with a 32 bit hardware multiplier, and the results are stored in a 32 bit registers again. Despite the outcome of the multiplications could exceed that size, we can avoid overflow or data loss by setting properly the quantization and normalization factors. Using 16 bit multiplication, we could not achieve enough precision.

The following table shows code results obtained for some 54-seconds sections, extracted from different 2-min data recordings.

Rec	Freq (Hz)	RR (br/min)	Code Res (br/min)	Error (br/min)
1	0.15	9	9.375	0.375
2	0.20	12	12.187	0.187
3	0.25	15	15.000	0.000
4	0.30	18	17.812	0.188
5	0.35	21	20.625	0.375
6	0.40	24	23.436	0.564
7	0.45	27	26.250	0.750
8	0.50	30	30.000	0.000
9	0.55	33	32.813	0.187
10	0.60	36	35.625	0.375
11	0.65	39	39.375	0.375

Table 3.4: SpO2 code results

As we can see, the reported results are quite accurate; in this case, the maximum error is $0.750 br/min$, with a mean difference of $0.307 br/min$.

The first test we have carried out on the code is the timing analysis. As known, this model of MSP430 microcontroller allows the user to set the internal digital controlled oscillator (DCO) module, in order to get different clock frequency, up to 25 MHz. The goal of the carried out analysis was simply to figure out how much time this algorithm requires to estimate RR with a single data section of 54 seconds, on this microcontroller.

For our test, we have chosen a time frame (TF) of 8 seconds, where here TF is the maximum time interval considered when running the algorithm. This value comes from the fact that it can contain every algorithm running within the selected frequency range; and it is also a likely time for the refreshing rate of the algorithm, in case of a continue real-time respiration rate computing. We tested the algorithm using different frequencies in the range from 6 MHz to 22.6 MHz

(in our specific case, higher values were not reachable due to hardware limitations). The aim of this timing analysis is to understand the law that describes the reduction of the execution time, with the increase of the frequency.

This is the law we obtained from our analysis:

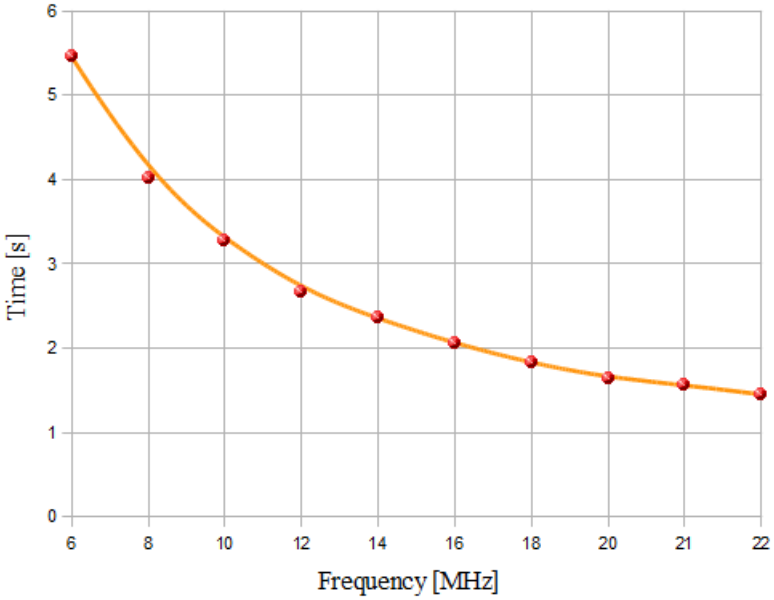

Figure 3.6: *Execution time versus clock frequency for the SpO2 code*

As we can see from the graph, despite this is a quite simple algorithm, it can require even a few seconds to compute a respiratory rate on a single data block. Unfortunately, in order to have good accuracy in results, it has not been possible to downsample the data before the IIR filtering. As a result of this, the code performs a high number of iterations, since it has to operate on the whole amount of data (3240 samples), rather than the reduced number of samples (108) obtained after the downsampling. This difference has emerged already in the Matlab-testing preliminary phase, where we obtained completely wrong results.

3.2.2 Accelerometer code

For this code we followed the procedure of digital filtering introduced in section 3.1.3. Also in this program we used the IIR and FFT routines written for the previous sensor. The implemented IIR was again band-pass type, Butterworth

8^{th} order, where the main difference was only the pass-band (0.1Hz - 1.0Hz), coefficients have been imported from *Fdatool*. About the FFT routine, it has been applied again to the filtered and downsampled signal, thus, as before, there is no imaginary part as input, and the output is another integer-type vector made by the 128 FFT modules. Even as before, no square root has been applied to the modules in order to make the code faster, but an opportune data normalization has been necessary.

The main difference between the two codes, is that in this case it has been possible to obtain good results even implementing the downsampling of the row signal, before the band-pass filtering. More in detail, the data from the sensor have been firstly low-pass filtered, with a FIR (anti-aliasing) filter as prescribed in section 3.1.3; then downsampled, and band-pass filtered (with the IIR). Spectrum analysis has been then carried, out as done for the previous code. This means that the IIR filter has been applied to a vector of only 108 signed integer values, dramatically reducing the computational time and the power consumption.

As we said in section 3.1.3, moreover, in order to achieve good accuracy with this sensor, it has been sufficient to process 45 seconds of sampled data for every calculation, leading to a much smaller memory consumption.

Indeed, despite data collected from the accelerometer can be safely stored in a *int16_t* type (16 bit) register, the elements size in the actual data buffer used for calculations is 32 bit, since we needed to quantize and normalize our data, in order to carry out calculations without overflow errors. With 16 bits data we could not reach the desired precision.

As before, data sampling rate was 60 Hz and we used 32 bits data version, but, since this time we needed only 45 seconds of data to extract the RR, we had to store only 2700 32bit signed integer values in RAM, sharply reducing the amount of memory required at the microcontroller initialization instant. The remaining space in memory is used to allocate other working variables like FFT and IIR arrays, as done for the SpO2 code.

The most remarkable result of the reduced memory consumption, together with the fact that in this code the IIR filter can safely operate using a much

smaller amount of data, is that, in this case, it has not been necessary to overwrite the data imported from the sensor, in the microcontroller memory, during the different filter stages. This has been possible because, in the RAM memory, there was enough space to store and refresh the samples from the sensor.

This is a very important feature in order to make this application real-time, because there is the possibility to safely "refresh" data from the sensor, simply by sampling a time frame of data containing a new portion of the accelerometer signal. The oldest time frame, instead, can be at this point safely deleted (or overwritten), because it is no longer used for the RR estimation.

As before, all the multiplications in the code are computed with a 32 bit hardware multiplier, and results are stored in a 32 bit register again. In order to avoid overflow or data loss errors, we needed to properly set the quantization and normalization factors.

Table 3.5 shows code results obtained for some 45-seconds sections extracted from different 2-min data recordings.

Rec	RR (br/min)	Code Results (br/min)	Error (br/min)
1	14	14.062	0.062
2	13	13.125	0.125
3	33	32.815	0.185
4	25	25.313	0.313
5	58	58.125	0.125
6	8	7.500	0.500
7	28	28.125	0.125
8	24	24.375	0.375

Table 3.5: *Accelerometer code results*

The results reported in the table 3.5 are quite accurate; in this case, the maximum error is $0.500 br/min$, with a mean difference of $0.226 br/min$. Obviously, we have tested the algorithm on much more 45-seconds data sections; the errors

reported are estimated only from a single 45-seconds data section per record, and they do not represent the exact value of error for the whole amount of data. However, they can be useful to have an idea of the order of magnitude for the expected error in RR estimation, using this algorithm.

We carried out a timing analysis also for this algorithm, and we compared the obtained law with the SpO2 one. As expected, this code is much faster than the one written for the pulse oximeter, with an execution time that can be even more than ten times shorter.

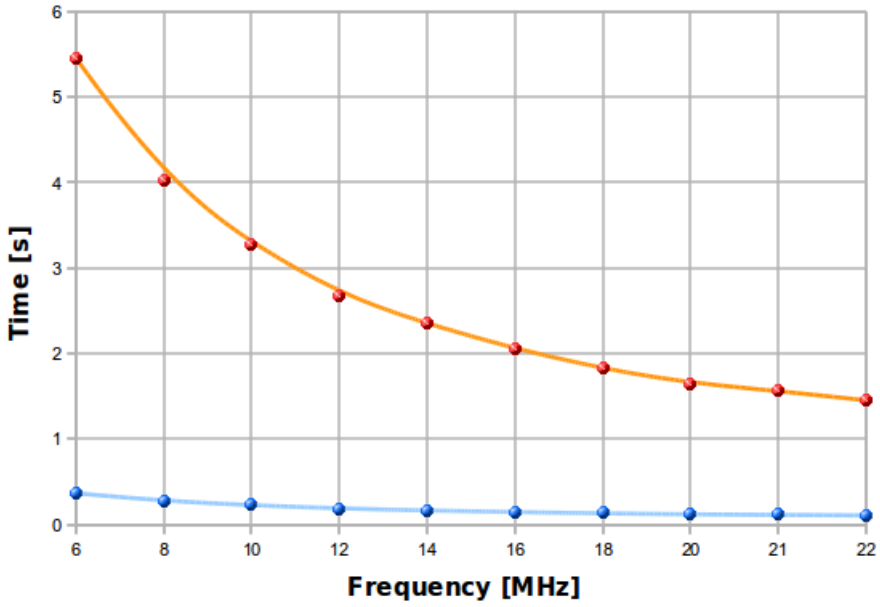

Figure 3.7: *Comparison between the codes execution times*

Chapter 4

The wireless MAC protocol for WBAN

In the networking field, the *Medium Access Control* (MAC), is a sublayer of the Data Link Layer, specified in the seven-layer OSI model (see figure 4.1).

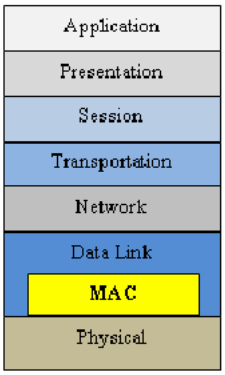

Figure 4.1: *OSI stack model*

It provides addressing and channel access control mechanisms that allows several terminals or network nodes to communicate within a multi-point network, typically a local area network (LAN) or, as in our case,a wireless sensor network.

MAC protocols can be roughly classified into three broad categories:

- *Fixed assignment class* - they have schemes like TDMA, CDMA and FDMA. The main criteria of these protocols is to divide the channel into smaller "pieces" of time or frequency, and to allocate each piece to a node for exclusive use. These methods suffer of a lack of flexibility in allocating

resources and thus they have problems with configuration changes. This facts make them unsuitable for dynamic and bursty wireless packet data networks.

- *Random assignment class* - they are a flexible class of methods, commonly employed in wireless LANs. Collisions are allowed and it is up to the protocol to "recover" transmission data from collisions.

- *Demand assignment class* - with schemes like Token Ring, GAMA and PRMA, this class of protocols attempts to combine the nice features of both the fixed assignment and random assignment classes, but special effort is needed to implement them in the wireless case.

The aim of this section is to provide to the reader the basic information about different types of MAC protocols, dealing with their characteristics and the main features that they must implement in order to satisfy the wireless body area network requirements.

4.1 WBANs MAC protocols basics

Body Area Networks (BAN) consist in a number of sensors that are either connected (somehow) with a person's body or are small enough to be implanted [37]. These sensors usually need to transmit data at a relatively wide range of rates, typically from 1kb/s to 1Mb/s, and they also need to consume as little power as possible, since power supply is very limited.

Current technologies meet the speed levels required for a BAN, but still do not meet the power requirement of less than 10mW according to [36].

There is no standard developed exclusively for Wireless body area network (WBAN). However, other wireless networks standards, such as IEEE 802.15.1 (*Bluetooth*) and 802.15.4 (*ZigBee*) can be employed in WBAN applications even though they are generally used for longer transmission ranges.

The main schemes for MAC protocols for sensor networks are CSMA/CA (carrier sense multiple access with collision avoidance) and TDMA (time division multiple access). FDMA (frequency division multiple access) requires a more complex hardware, while CDMA (code division multiple access) has high computational demands,so they are not normally used in WBANs.

In the following section, we intend to briefly show the main features of these two families of protocols and make a comparison about their application in WBANs.

4.1.1 A comparison of TDMA and CSMA in the context of WSN

Carrier Sense Multiple Access (CSMA) is a probabilistic MAC protocol, in which each node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus, or a portion of the electromagnetic spectrum.

"*Carrier Sense*" describes the fact that a transmitter uses channel measurements to detects carrier wave before trying to transmit. This means that the transmitter node tries to detect the presence of a signal from another station on the channel, before attempting to transmit. If a carrier is sensed, the station waits for the transmission in progress to finish, before initiating its own transmission.

"*Multiple Access*" describes the fact that multiple stations are allowed to transmit on the medium. Transmissions by one node are generally received by all other stations connected to the medium.

The basic CSMA scheme described above can be actually implemented in two different ways:

1. *CSMA with Collision Detection* ***(CSMA/CD)*** - is used to improve CSMA performance by terminating transmission as soon as a collision is detected, and reducing the probability of a second collision on retry.

2. *CSMA with Collision Avoidance* ***(CSMA/CA)*** - it improves the performance of CSMA by attempting to be less "greedy" on the channel. If the

channel is sensed busy before transmission, the transmission is deferred for a "random" interval. This reduces the probability of collisions on the channel.

The classic CSMA/CA protocol is the *IEEE 802.11* standard. Although it prevents collisions, the channel utilization decreases when the number of nodes increases, as bandwidth is wasted by taking repeated back-off to avoid collisions. In the context of WSNs with on-demand and periodical data gathering these characteristics of CSMA protocols are not efficient [38]. Constant probing of the medium in the long inactive periods can quickly drain nodes' batteries. Also, when the network is active, considering the large number of nodes and high density in WSNs, the CSMA protocols again prove to be inefficient due to reduced channel utilization.

TDMA is a type of *Time-division multiplexing*, with the special feature that instead of having one transmitter connected to one receiver, there are multiple transmitters.

TDMA protocols allow several users to share the same frequency channel by dividing the signal into different time slots. The users transmit in close succession, one after the other, each using his own time slot. This allows multiple stations to share the same transmission medium (e.g. radio frequency channel) while using only a part of its channel capacity.

It is very important that the slots are assigned in a way that prevents transmission interferences, or conflicts. The latters can be of two main types:

1. Primary conflicts caused by two adjacent nodes sending data at the same time.

2. Secondary conflicts caused by two non-adjacent nodes sending data to a single, third receiver. This is known as the "*hidden node*" problem and is handled in CSMA/CA by introducing request to send (RTS) messages.

Hence, the transmission is discontinuous, due to the fact that every node has a cyclic access to the common radio resources during the allocated time-slot. Figure 4.2 presents a TDMA system example with 4 time-slots in the TDMA frame.

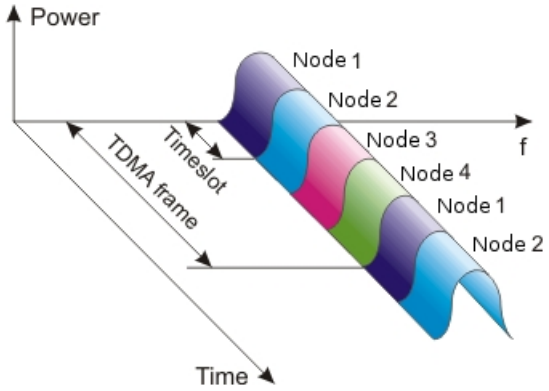

Figure 4.2: *Example of TDMA scheme.*

TDMA protocols will only schedule the activity of the network, specifying in which period every node must be active. In the idle times between each data gathering session, nodes can turn off the radio interface and switch to a sleep state. Thus, the main and most important advantage of TDMA is the low-power consumption. Also, TDMA achieves better channel utilization as the number of nodes increases.

In table 4.1, a qualitative comparison between TDMA and CSMA protocols is given, concerning their utilization in WBAN medium access control [37].

	TDMA	**CSMA/CA**
Power consumption	Low	High
Bandwidth utilization	Maximum	Low
Preferred traffic level	High	Low
Dynamic (network change)	Poor	Good
Effect of packet failure	Latency	Low
Synchronization	Crucial	-

Table 4.1: *Comparison between TDMA and CSMA/CA protocols*

TDMA based protocols outperform CSMA based ones in all areas except the protocol adaptability to changes in network topology.

There are WSN applications where TDMA is not appropriate and where topology, mobility or expected data-rate play a major role in selecting the MAC protocol. TDMA does not adapt well to network changes which are quite frequent

in sensor networks due to node failures.

Also, nodes have to be synchronized in such a way that the clock drift between nodes be negligible compared to the slot size. Synchronization problems can lead to high latency and power consumption, caused by interference from overlapping schedules. Thus, TDMA based protocols need a very good synchronization scheme. Such schemes are not easy to implement in a dynamic network, but since WBANs have fixed distances between sensors and fixed sensor functions, this problem does not arise.

While CSMA protocols may be suited for *event-driven WSN applications with dynamic topologies*, TDMA protocols work best in *applications with periodical or on-demand data gathering*. Monitoring with non-critical data (no real-time requirements) is a good example, where sensors gather data for a long time, then they wake up and start transmitting to a central node or gateway.

Recent works have proposed TDMA-based MAC protocols for wireless sensor networks (especially for WBAN), but still, there is no WBAN protocol standard available.

4.2 Protocol implementation

The protocol presented in this paper has been developed considering explicitly BANs and their characteristics. It presents an attempt to overcome the problems in this type of networks. The protocol is designed to meet the following requirements:

- *Collision-free transfer* – collisions are one of the major power wastages in a sensor network. This protocol overcomes it by using the TDMA approach, thus keeping collisions on the lowest level possible.

- *Communication error tolerance* – our protocol exploits redundancy in the transmitted data to lower the chance of losing packets as much as possible. (Generally, due to the short node distances, packet loss happens only in case of hardware fault, so rarely.)

- *Energy efficiency* – the protocol presented allows long sleep times for sensors, without the need for channel listening. The communication has the smallest possible overhead.

- *Real time patient monitoring* – the protocol allows for all the channels to be observed in real time, without packet loss.

The network is designed allowing three type of nodes: slave, master and monitoring station. *Slave nodes* are used to sample the sensors, gather the information and send it to the nearest master node. *Master nodes* function is to collect data from several slave nodes assigned, and then, send them to the monitoring station. Finally, the *monitoring station* has to do the computation on the data received from masters, and to transmit the results to the user.

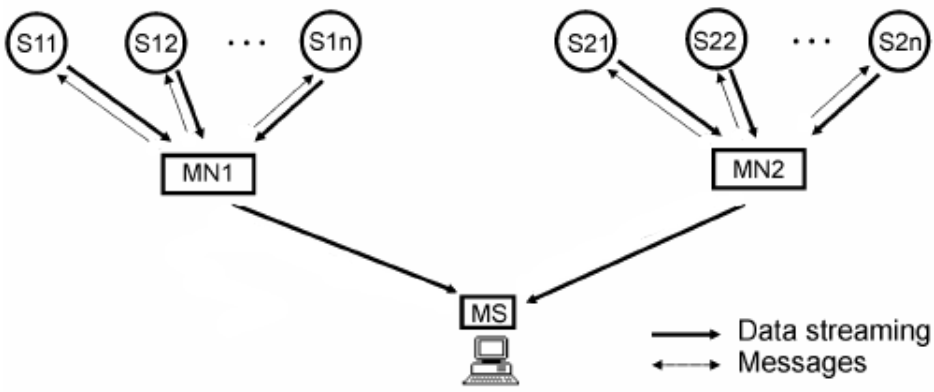

Figure 4.3: *Topology example of proposed WBAN.* ***S*** *= Slave node,* ***MN*** *= Master node,* ***MS*** *= Monitoring station*

The WBAN Network topology for the proposed MAC protocol is given in Figure 4.3. Considering the presented network architecture, there is the possibility of lowering the amount of unnecessary overhead and idle listening. The idea is to use a very simple but efficient TDMA MAC strategy. We used the master nodes for coordinating the synchronization and for the transmission to the monitoring stations.

Actually the realized network architecture is even simpler, because we used only one master node for collecting data from sensors.

4.2.1 Protocol timing and logic

The goal is to make communication time as small as possible compared to the duration of the sensor power down mode. During power down mode only the internal time counter and periodical ADC sampling are active. TDMA timing Timeslots S1-Sn are allocated to n different sensors.

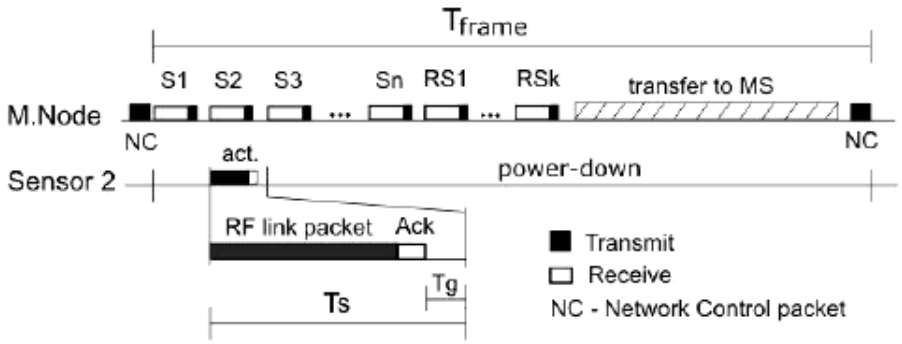

Figure 4.4: *TDMA timing division.*

Between every two consecutive slots a guard time (T_g) is inserted. This is necessary to avoid overlapping of the transmissions from different sensors due to clock drift. The remainder of the T_{frame} is used for transmission to the Monitoring station.

It must be noted that this is not the only possibility. The master node can have multiple transceivers, thus allowing the simultaneous communication between sensors and MS. At the end of the T_{frame} the master sends a Network Control (NC) packet, which is also used for timer synchronization (*SYNC message*) and network commands. The sensors will adjust the TDMA counters immediately after they receive the SYNC message.

After the initial synchronization the slot assignment procedure starts. Slot assignment can be done using different strategies. The first strategy is to use a *predetermined slot assignment*. This is the easiest way, but requires the sensors to have preconfigured information about time-slot structure.

The implemented and simulated BAN uses this method of slot assignment: each node computes a slot ID (named *MY_SLOT*) which is simply computed as

his own address less one. Of course the network address is assigned in a static way, as well. This strategy lowers the control overhead, but greatly reduces flexibility.

The second strategy for slot assignment consists of using the NC packet for the assignment. This is a more flexible option. Good strategies must be developed here to avoid collisions during the network forming. This might be a perfect cue for future research projects.

4.2.2 Node function description

The aim of this section is to describe with further details the structure of every kind of node in our network.

Slaves node (sensors)

As said before, slaves nodes in the network are responsible for the acquisition of data from the sensor. The sampling of the sensor layer has been achieved by setting a second timer which wakes up the analog-to-digital converter module of the microcontroller (*ADC12*) at every interrupt and reads a value from the sensor.

So, we have two concurrent timers running simultaneously: one for time slots timing and the other for data sampling. Of course they are not synchronized, but they don't have to be, because they are logically independent.

In figure 4.5 the logic flow of the software running on a slave node is shown. The sampler continuously fills a buffer with new data every second. It is obviously necessary that, when the frame enters in the time-slot chosen for the transmission, the data buffer containing the received samples from the sensor has to be ready (i.e., with enough samples to complete the packet payload).

After the basic microcontroller initialization procedure, the sampling starts by calling the function "*start_ADC_Sampling(SAMP_FREQ)*", and then the program enters the main loop where it repeatedly checks the current time-slot and acts consequently.

The "MPU initializations" block contains all the necessary functions to make the microcontroller working: the settings of the timers (for time slots and sam-

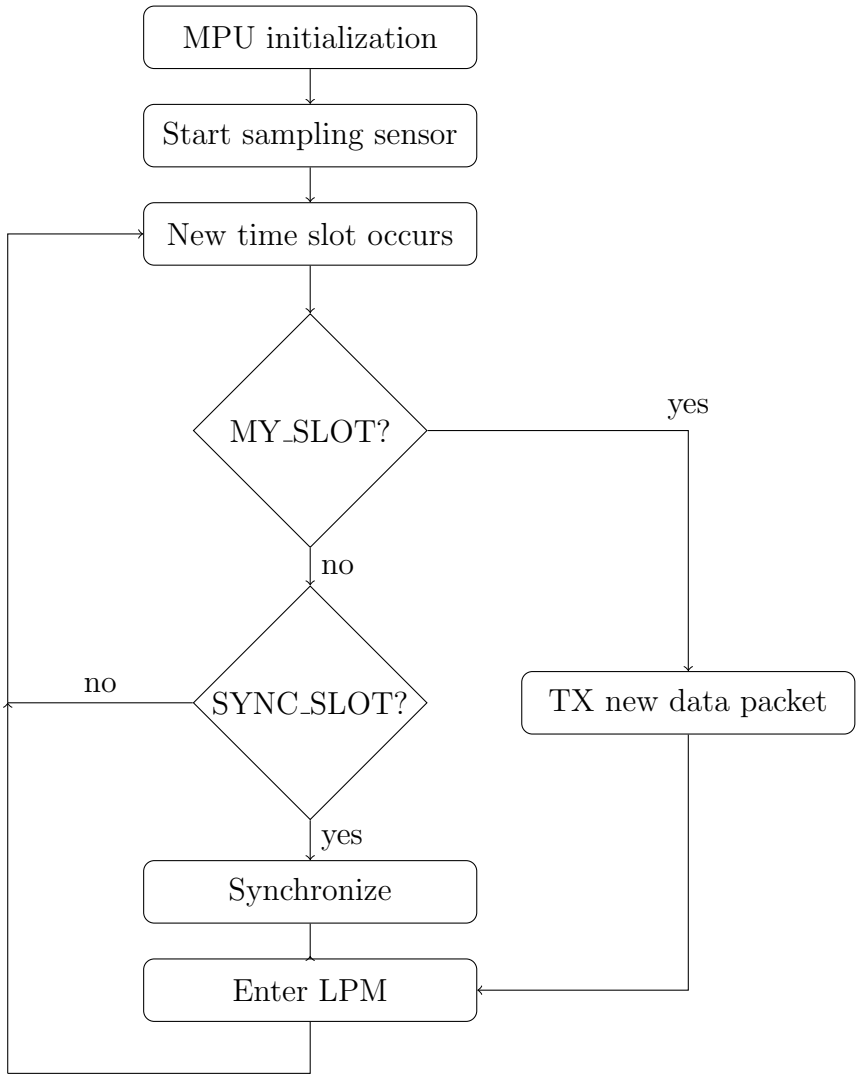

Figure 4.5: *Logic sequence of the slave node program*

pling), ADC, clock system, SPI bus and ADF7020 radio.

After the program has completed his control task, it enters in low power mode. *MSP430* allows users to choose among different low power modes, depending on which microcontroller part we want to shut down. This is possible because MSP430 provides to the designer a choice of three different clock trees, so it is up to him/her to select which one to use and when. In our implementation our choice has been *LPM3*, which switches off microcontroller CPU core, the master clock and the secondary master clock. The auxiliary clock is kept active; this fact

is important because the auxiliary clock is used by timers and some peripheral modules like the internal analog-to-digital converter.

It may seem that use exclusively low-power modes with the auxiliary clock on is the only possible choice. This is not true. Indeed, we can decide to switch off the auxiliary clock as well, by entering *LPM4*. Timers and other modules will continue to work, because by default the microcontroller has an internal control that keeps a clock tree active whether it is requested from some internal module.

Once entered into the main loop, the gathering of the data samples is carried on by an interrupt service routine, which "wakes up" the processor and reads the sensor, and finally coming back to the selected low-power.

Use of low power modes can reduce significantly the power consumption during the sampling operation and standby, but it doesn't change drastically the power needed to transmit the packet. Further details about protocol power consumption, will be given in chapter 5.

As seen before, the node synchronization is obtained by using a network control packet (*SYNC)* from master node to slave nodes. In particular, we chose a specific slot (26) where, every T_{frame} seconds, the software controls whether a resynchronization is needed or not (at the end of slot 26 the listening mode of the sensor starts, but the master sends the synchronization packet in slot number 27, the last one in the protocol cycle).

Each sensor can determine its own resynchronization period through the slot guard time. The number of TDMA cycles (N_R) that can pass before the sensor needs to resynchronize is calculated as [38]:

$$N_R \leq \frac{1}{2}\left(\frac{T_g}{T_{frame} \cdot C_{ac}}\right) \tag{4.1}$$

where T_g is the slot guard time, T_{frame} is the frame time and C_{ac} is the sensor clock accuracy. For resynchronization, each sensor needs to turn on its transceiver and wait for the *SYNC* message a time-slot earlier than the packet is expected. Once the network control packet is correctly received, the transceiver is switched off immediately, avoiding an useless energy waste.

Figure 4.6 shows with more precision the synchronization steps.

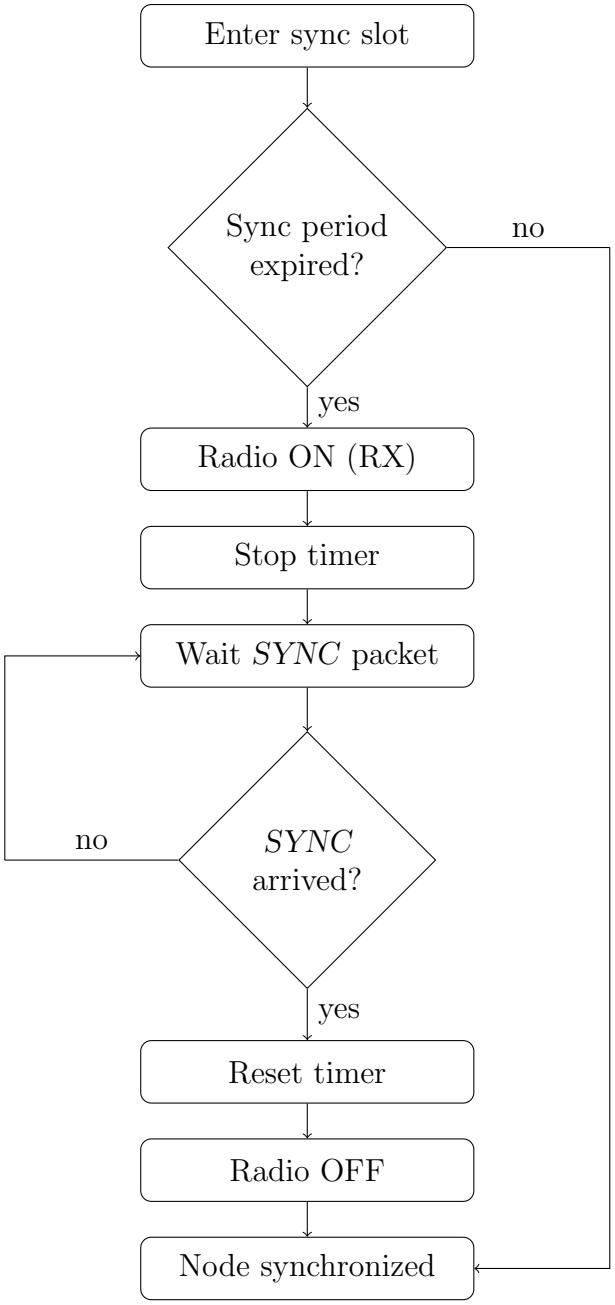

Figure 4.6: *Synchronization sequence for slave node*

The communication between the microcontroller and the transceiver is implemented by an SPI standard bus. In this case, *MSP430* works always as slave

device, because the clock for data transfer is provided by *ADF7020*.

To allow the reception of the message, SPI bus has to be set with input direction. This is automatically done by an interrupt service routine when an interrupt on a specific microcontroller port (*"PORT 1"*) is detected. Indeed, in the Tyndall mote layer we used, there is a pin on this port that is connected to an important *ADF7020* chip output signal (named *"RF_INT/LOCK"*). This signal is asserted by the transceiver every time a match for the packet preamble sequence has been recognized. This means that a new packet (in our case the *SYNC* message) is ready to be read on the slave node SPI input channel.

There is no real need for reading the payload of the *SYNC* message: when we know that the preamble is received correctly, we can already consider the node synchronized, so it it possible to reset the timer and the appropriate counters.

Once the node is synchronized, communication can start. When we want to transmit a data packet, the logical steps performed by the code are indicated in figure 4.7.

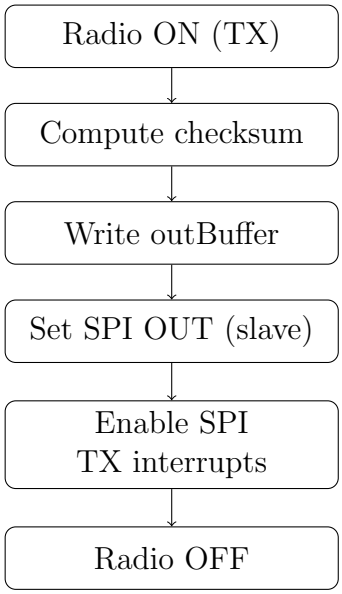

Figure 4.7: *Transmissions logic for slave node*

Firstly, we have to check that the output buffer is filled with all the necessary information (protocol preamble, payload with data samples and checksum). In

order to send the contents of the out buffer to the transceiver (which has been previously switched on), we have to configure the SPI module direction as output. After, simply by enabling SPI transmitting interrupts the communication can start.

Master node and Monitoring station

The program for the master node is pretty similar to the slave one. Indeed, after the initialization, the program enters the main loop and checks the arrival of data packets in the first eight slots. Every packet which is correctly received is saved in a matrix buffer, otherwise an error is signaled on the serial port.

When the listening phase ends (hence, from ninth time slot forward), the transmission to monitoring station starts.

The final task of the master node, as stated above, is to send the broadcast synchronization packet to the slaves nodes. To perform this, the master switches the transceiver on in the twenty-sixth time-slot, and sends the *SYNC* message in the following one.

In the case of the monitoring station, code is even simpler; it only have to collect the data and perform the required computation on them.

To make things easier, the master node has no low-power mode running on it, because most of the power is spent sending the packets and in the listening mode.

4.3 C code

4.3.1 Data acquisition

As explained in the previous sections, the sampling of the sensor layer has been carried out by using the internal analog to digital converter ADC12. This micro-controller module allows the designer to choose different data formats: 8, 10 an 12 bit per sample. In our case, we have set the module for sample with 12 bit data format, but we write in the output buffer only the 8 most significant bits.

Every slave node has three distinct buffers, each one containing a character data:

- *in_buffer* - is the buffer where the program stores one character when it senses a receive interrupt on the SPI bus.
- *out_buffer* - is the buffer used to transfer a data packet to the radio. It contains the preamble bits, the packets length, the sampled data and the checksum.
- *adc_buffer* - this is the buffer where sampled data from the sensor are stored.

During the microcontroller's initialization phase, *ADC12* module is initialized as well. This is achieved, by setting the voltage references for the conversion and conversion type, and selecting the correct channels of the converter.

We have chosen as voltage references for the analog-to-digital converter, the value of V_{DD} for the maximum level in the range and *GND* for the lowest one. In this case, the average of the samples offset, without any physical input (acceleration or heart pulse) is set more or less at the middle of the values range. An other possible option is to use the microcontroller internal voltage reference (1.5V or 2.5V), but we realized that this choice does not fit accurately with the kind of data we had to work with.

As regards the conversion type to perform, we set the converter module in order to work only through not repeated single conversions.

About the accelerometer, we can choose whether to sample one or three channels in sequence, while, for the SPO2 sensor, we always sample only one channel per time, depending on whether we want to collect red LED or infrared LED data samples.

Each sensor needs to be enabled to start working, otherwise they remain switched off to save energy. The sensor enabling and the configuration of the timer for the sampling interrupt, is handled into the function "*start_ADC_sampling(SAMP_FREQ)*".

Figure 4.8 shows the core of the function, that is the same for every sensor. As we can see the module uses in this case the auxiliary clock, so it is already

prepared to enter a low-power mode, by turning off the master clock tree without any affection of the timer functioning.

```
void start_ADC_sampling(short int samplFreq )
{
    int cycles = 0;

    set sensor enable pin...

    cycles = (int)(f_aclk/samplFreq);
    TB0CCR0 = (cycles - 1);
    TB0CTL = TBSSEL__ACLK | MC_1 | TBCLR;   // upmode, clear count
    TB0CCTL0 = CCIE;                        // enab timer interrupt
}
```

Figure 4.8: *Pseudo-code of sampling data function.*

Once the sensor is enabled, the sampling starts by setting the timer interrupts bits in the status register. Figure 4.9 lists the partial code of interrupt service routine *(ISR)* which implements the accelerometer sampling.

```
#pragma vector=TIMER0_B0_VECTOR
__interrupt void TIMER0_B0_ISR(void)
{
    unsigned int x;

    "turn on ADC12 module, enable then start conversion..."

    if( SINGLE_CH )
    {
        x = ADC12MEM0;
        adc_buf[ adc_bufIndex++ ] = x>>4;
    }
    else
    {
        if( (adc_bufIndex+2) < ADC_BUF_SIZE )
        {
            x = ADC12MEM0;
            adc_buf[ adc_bufIndex ] = x>>4;
            x = ADC12MEM1;
            adc_buf[ adc_bufIndex + 1 ] = x>>4;
            x = ADC12MEM2;
            adc_buf[ adc_bufIndex + 2 ] = x>>4;
        }
        adc_bufIndex += 3;
    }
    "switch off ADC12 module"
}
```

Figure 4.9: *Pseudo-code of accelerometer's sampling timer ISR.*

So every time that a timer interrupt occurs, a new data byte is stored into the *adc_buffer*. Obviously buffer size has to be set accordingly with the sampling frequency we are using.

Note that the protocol meets the real-time requirement. The aim is to guarantee that every second a number of data byte (according to the sample frequency) is transmitted. This goal is reached by using a sampling frequency slightly higher than the payload size, so we can be sure that in the next time-slot interrupt, the *adc_buffer* will be filled with the necessary number of samples, ready for transmission. For instance, we need sampling frequencies on the order of 60 Hz for the SPO2 and 120 Hz for the accelerometer, but the effective sampling frequencies are increased by 5 Hz each.

Typical data sequences extracted from sensors are reported in figure 4.10 and figure 4.11. A more precise description of the involved signal waveforms has been given in chapter 2, section 2.

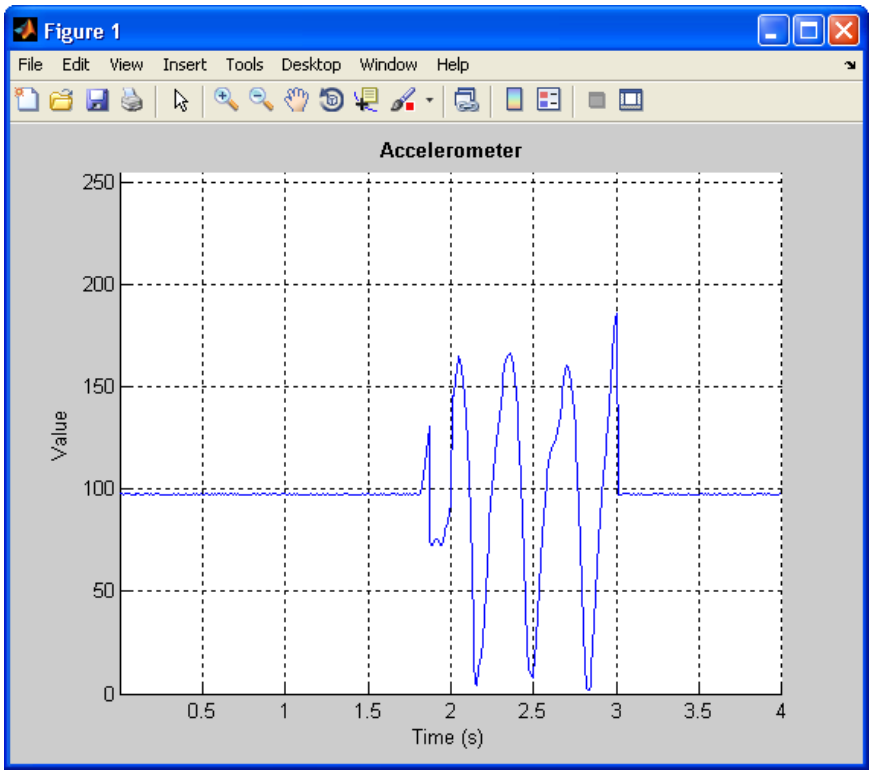

Figure 4.10: *Example of Tyndall accelerometer layer data.*

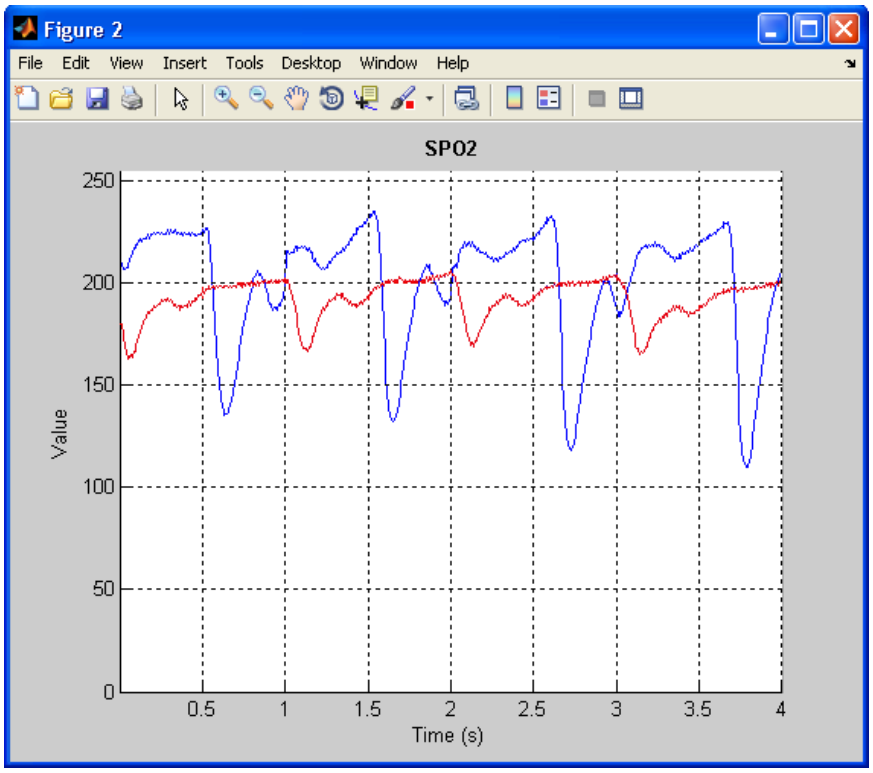

Figure 4.11: *Example of Tyndall SPO2 layer data. In red is the red-LED signal waveform, in blue the infrared-LED one.*

The ADC converter module is then switched on only during the effective sampling period, while the sensor is always activated at the beginning of the sampling and never disabled.

4.3.2 Transmitting and receiving data

This section goal is to describe the core of the protocol, that is the interrupt service routine designed to manage the transmission and reception of data.

MSP430 implements the standard SPI bus communication protocol by using an instance of the microcontroller module named "*USCI*", which can handle several serial bus standards ($UART$, SPI or I^2C).

First of all, the program has to check the interrupt vector register of this module to realize whether a transmission or reception event has occurred, as shown in figure 4.12.

```
#pragma vector=USCI_B0_VECTOR
__interrupt void USCI_B0_ISR(void)
{
   switch("check SPI interrupts")
   {
      case 2:     //RX interrupt
         switch (REC_STATE)
          {
            case 0:
               if ("packet lenght is wrong")
               {
                  REC_STATE = 0;
                  disable SPI RX interrupts...
               }
               else
               {
                  REC_STATE++;      //next state
                  chcksum = 0;
               }
               break;
            case 1:
               read data byte from SPI RX buffer...
               if ("end of the packet")
               {
                  if ("chcksum is correct")
                     ERROR = 0;
                  else
                     ERROR = 1;

                  clear counters for the packet...
                  REC_STATE = 0;          // reset logic RX state
                  RECEIVED = 1;           // packet received!
                  disable RX interrupt, put SPI to HZ...
                  break;
               }
               update checksum with adding last byte...
            break;
          }
      break;
      case 4:           // TX interrupt
         if ("packet NOT ended")
            write next data byte in SPI TX buffer...
         else
         {
            disable TX interrupt...
            put SPI to high impedance...
         }
         break;

       default: break;
   }
}
```

Figure 4.12: *Pseudo-code SPI bus manager ISR*

In case of reception of a byte, there is a second switch-case statement that handles the interpretation of data. The flag "*REC_STATE*", is used to determine which part of the packet we are reading (preamble, protocol information or data).

The first control implemented verifies if the packet length itself exceeds the maximum allowed packet size: if so, data loss has occurred during the transmission and the packet is discarded (it is also possible to set here a flag to send a "retransmission" packet in the next synchronization slot).

If the packet size is correct, the reception of data bytes starts: the error in that case, might be given by an incorrect value of the checksum (some byte in payload has changed its value during the transmission). If no error has encountered, a flag signals that the packet has been received fine, and receiving interrupts are disabled.

During the data sending phase instead, transmitting interrupts stay enabled until the last data byte has been sent, then the SPI serial bus is put in high impedance.

Chapter 5

Power consumption measurements

5.1 Experimental configuration

The power consumption analysis treated in this chapter has been carried out either on the protocol implementation or on the implementations of the algorithm.

The power measurements, has been conducted on the 25 mm Tyndall motes, using the same operating conditions for every part of software tested.

The issue of the power consumption profiling may have different solutions, depending on the desired measurement accuracy and the kind of device under test.

Most current measurement applications employ either the *low-side principle*, in which the sense resistor connects in series with the ground path (between the ground of the mote and the negative power supply); or the *high-side principle*, in which the resistor connects the positive power supply and the mote power input. The low-side resistor adds undesirable extraneous resistance in the ground path and this might, sometimes, cause the ground to bounce as the current in the system fluctuates. The high-side resistor instead, could introduce higher common-mode voltage values.

Technical literature about the power/current measurements is crowded of alternative schemes, involving current mirrors, capacitors, amplifier stages, etc; but

this goes beyond our needs.

Figure 5.1 represents the circuital scheme of our application.

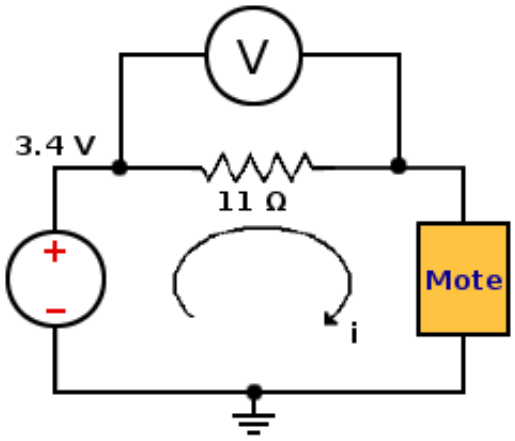

Figure 5.1: *Scheme of power measurement application circuit.*

Using a simple sense resistor in series to the current path (between the power supply and the mote) may not give enough vertical resolution for the oscilloscope when the current consumption is under 1 mA [39]. However, given that we are interested in the study of the active state power analysis and not of the low-power mode one, we adopted this method for our measurements to make things easier.

The size of the sense resistor is small (11 Ω with 1% tolerance) since we have a low input voltage drop, and to guarantee enough sensitivity to accurately measure the low-power consumption.

Obviously, this method can work only if the load (i.e. the mote) does not drain too much current; otherwise the voltage across the resistor would significantly increase, thereby compromising the voltage supply of the load itself.

Each measurements has been accomplished on the same mote device, always using the same type of antenna. To obtain the power consumption of only the instructions effectively involved into the task under test, we measured the voltage waveform on the sense resistor by using an oscilloscope. In order to achieve the real computation power, we firstly have obtained the current contribute of the tested instructions by measuring the graph beneath area of the recorded

waveform, in a certain amount of time. Then, we must subtract from the obtained current value the contribute in current of the standby state of the microcontroller.

To help the placing of the oscilloscope cursors on the exact instants, before and after the calculation, the CPU is immediately put in low-power mode (*LPM4*). Even though in this mode the microcontroller is almost switched off, there is still a little voltage bias that introduces a noise in the measurements. The formula used to compute the power P is the following:

$$P = \frac{(I_{avg} \cdot V_{DD})\Delta t}{T_{frame}} \tag{5.1}$$

where V_{DD} is the microcontroller voltage supply; Δt is the effective computation time of the algorithm; and I_{avg} is the effective calculation current (after subtraction). Figure 5.2 below shows a picture of the experimental setup used:

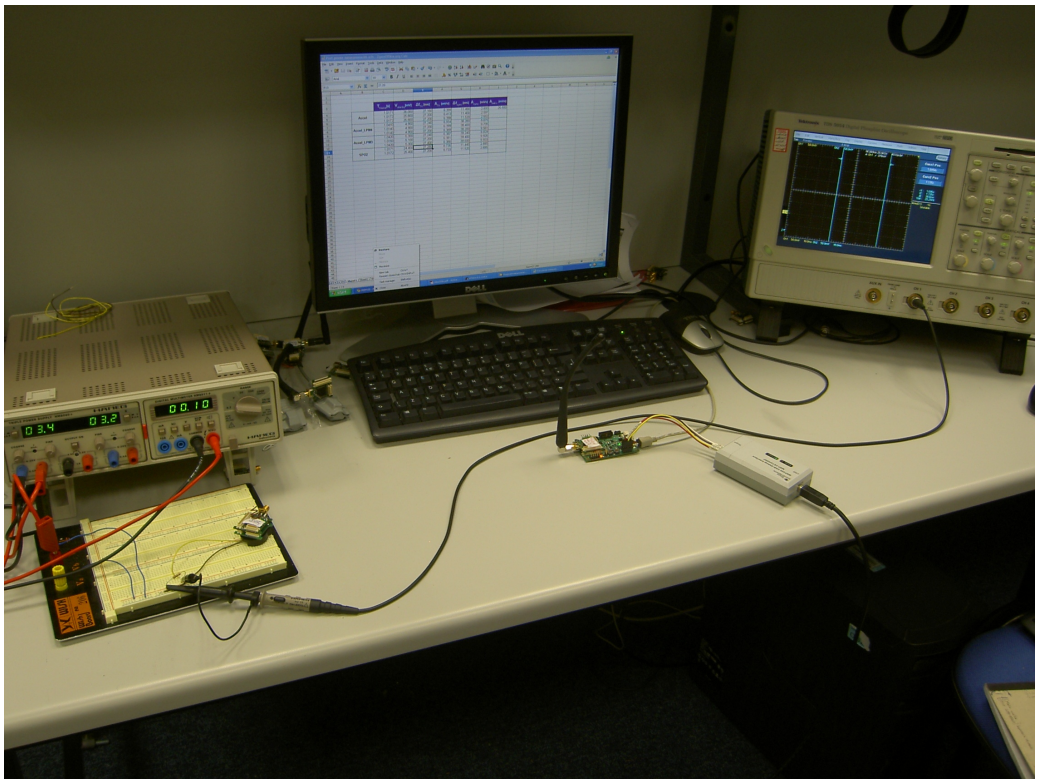

Figure 5.2: *Picture of the experimental setup used.*

5.2 Measurements

5.2.1 Protocol power consumption

About the protocol measurements, we only have taken care of the slave and master nodes power consumption, since the monitoring station is plugged to the PC serial port and it is anyway supposed to be the most power hungry node in the network (because of the calculation).

We firstly tested the slave nodes noting that, analyzing the same amount of time, both the slave node implementations (*accelerometer* and *SPO2*) consume the identical level of power, since it only depends on the number of transmitted packets and the radio settings (and not on the sensor itself).

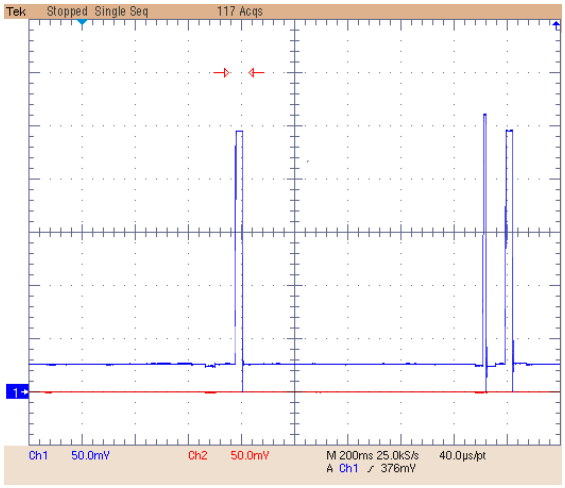

Figure 5.3: *Voltage waveform example of slave node.*

With regard to the master node instead, power consumption is definitely much higher, because we are always keeping the radio chip switched on during both the listening and transmitting time-slots.

An example of master node receiving four packets is shown in figure 5.4.

Transmitting power in that case is set to an high value because of the huge distance from master node and monitoring station (which is markedly bigger

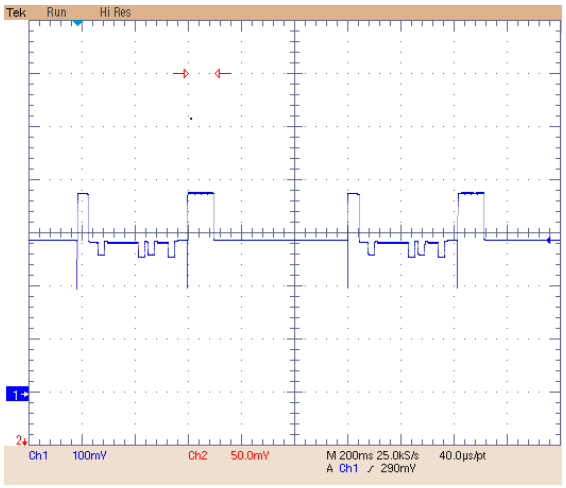

Figure 5.4: *Voltage waveform example for master node.*

than the slave-master distance). It is possible to note that, during the effective receiving time, the level of power is actually lower than the normal "Rx listening radio": it means that the more packets it receives during the listening time slot, the lower is the overall power consumption of this fist phase.

Reckoning with this, we made a comparison to see whether the trend of the overall power consumption, considering both receiving and transmitting phase, could decrease with the increasing of the number of received packets.

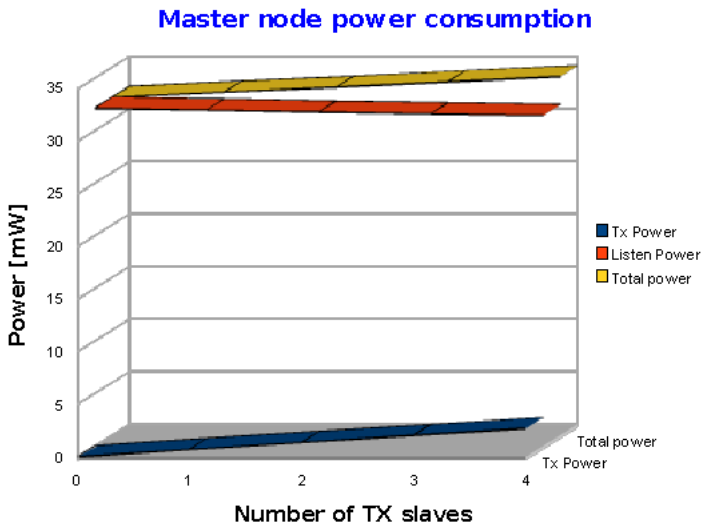

Figure 5.5: *Master power consumption versus number of transmitting slave nodes.*

As figure 5.5 shows, we have a small fall off trend on the power requested in reception, but it is overcompensated by a strong increase of power required by the transmission. So, in the end, as we expected, the total required power is anyway linearly growing with the number of transmitting slaves nodes.

5.2.2 Algorithm power consumption

As known, this model of MSP430 microcontroller allows the user to set the internal DCO module, in order to get different clock frequencies, up to 25 MHz.

The goal of the carried out analysis was to figure out which is best frequency in order to low the computation power consumption as much as possible (of course neglecting any kind of timing constraint).

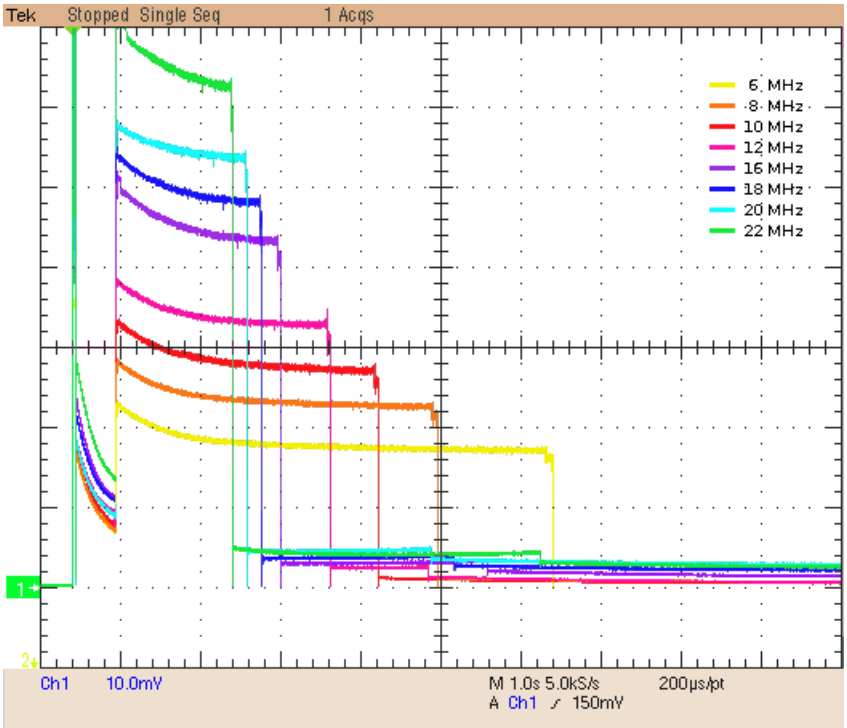

Figure 5.6: *Example of different voltage waveforms profiles of the algorithm.*

For our test, we have chosen a T_{frame} of 8 seconds for the pulse-oximeter algorithm implementation, and a 3 seconds T_{frame} for the accelerometer one; where here T_{frame} is the maximum time interval considered when running the

algorithm. We have tested the algorithm under different operating frequencies in the range from 6 MHz to 22 MHz (see an example in figure 5.6).

Each measurement comes from the average of ten values: this is necessary because of the high level of noise on the standby voltage waveform while running the low-power mode.

We have not used lower frequency values because in that case the execution time would has exceeded the bound of T_{frame}, while higher values are not reachable due to hardware limitations. Indeed, in order to use frequencies from 19MHz forward, is mandatory to set the microcontroller core voltage to the maximum level. Even after that, an high V_{DD} voltage is required (up to 3.6V), but in the Tyndall mote we have used V_{DD} has been fixed to 3.3V. For this reason, the last stable frequency value for this device was experimentally determined around 22.6 MHz.

Thus, for every frequency, the execution time and power consumption only of the effective instructions involved into the algorithm calculation have been measured. In table 5.1 an example of the kind of result we have had for the 18 MHz test is reported.

	Time		**Power**	
	[ms]	**(%)**	**[mW]**	**(%)**
IIR	1780	95.442	3.232	94.117
FFT	45	2.413	0.116	3.372
Downsampling	16	0.858	0.035	0.035
Initial + extrac.	24	1.287	0.052	0.052

Table 5.1: *Example of power and time repartition at 18 MHz.*

Of course we know that the instant power utilization for the algorithm computation raises with the frequency. Anyway, considering the decreasing trend of the execution time versus the frequency (see section 3.1.3), we expected a convex shape curve to describe the overall power consumption of the algorithm.

As shown in figure 5.7, this is roughly what we obtained for both kind of the

implementations.

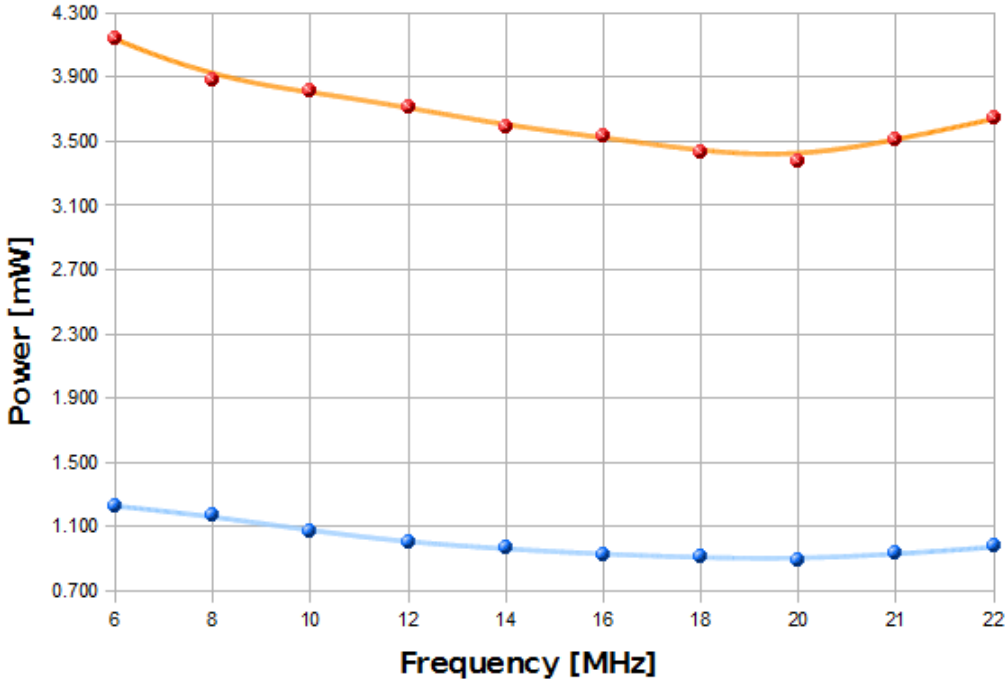

Figure 5.7: *RespRate power consumption for both the implementations.*

In our study we also have performed an other type of power profiling, which is shown in figure 5.8 and 5.9. The aim in this case, is to understand how the power consumption is divided up among the diffent sections of the algorithm.

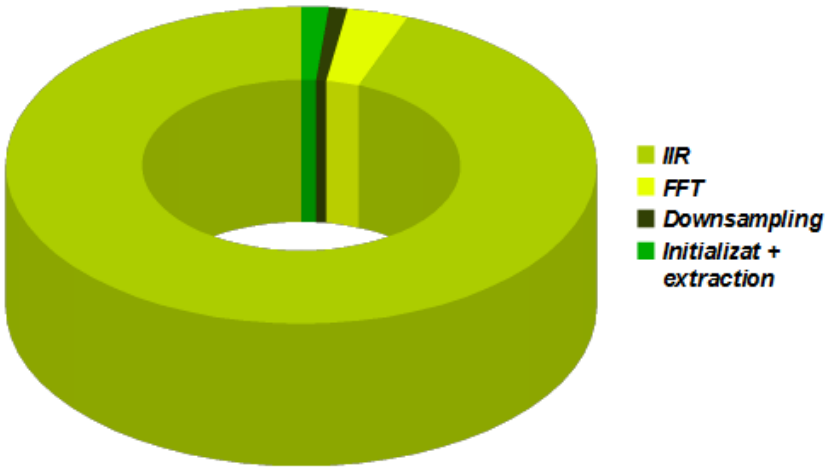

Figure 5.8: *Power repartition for pulse-oximeter algorithm.*

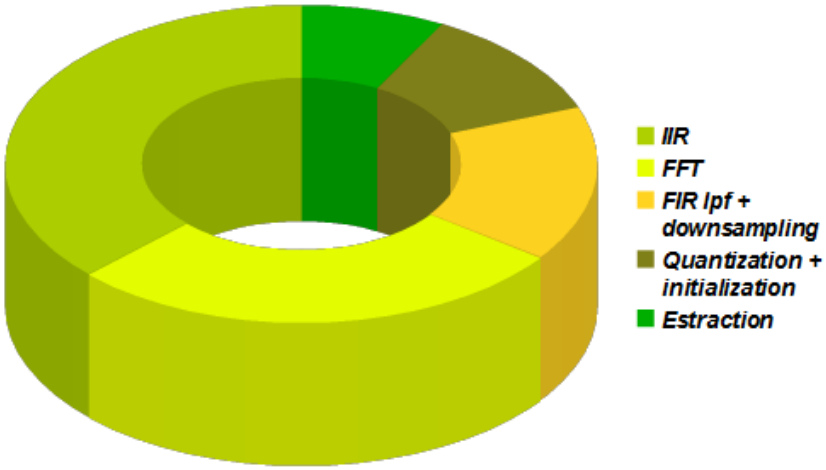

Figure 5.9: *Power repartition for accelerometer algorithm.*

Even though these results may change a bit with the frequency, the rough indication of the percentage remains the same. A comparison between the two different implementations is possible due to the presence of common functions in the codes (IIR, FFT, initialization and downsampling).

After this analysis, we have noticed that in the algorithm for pulse-oximeter data, almost 94% of the power is spent on the four-stage IIR, while in the accelerometer implementation only 37%. This is consequence of the fact that, the IIR filter, is computed over a huge amount of data in the pulse-oximeter version but not in the accelerometer one; and it is also the reason why the overall power consumption of the implementation for the accelerometer sensor is about four time smaller than the other. In both cases, the IIR filter is the most power hungry part of the calculation, followed by the fast Fourier transform.

Chapter 6

Conclusions and future works

6.1 Achieved results

Results show that the estimation of respiratory rate, using a finger pulse oximeter and a chest accelerometer, can be achieved with good accuracy using simply and efficient solutions. The main drawbacks for these algorithms, is that they are not robust against motion artifacts, simply because no precautions for this kind of noises have been implemented.

These methods however, have been tested only on healthy subjects and thus, they may not obtain accurate results for irregular breathing. The reason of this, is that we are assuming the existence of a "dominant" breathing frequency, and we process the signals in order to extract it. In case of deep irregular breathing however, is no longer guaranteed that such a frequency exists, and this may lead to wrong results in estimating the RR. In such cases, a different approach from the simple digital filtering has to be adopted.

By analyzing the timing chart (see 3.1.3), we can see a sort hyperbolic trend of the execution time versus the clock frequency. This suggests us that, over a certain value of clock speed (let's say 20 MHz), a further reduction of the frequency does not worth it: the decrement of the execution time is actually too small, compared to the increase of the power necessary to boost the clock frequency.

Results also show that the estimation of respiratory rate using the accelerometer, can be achieved with a faster code and with less power consumption, in comparison with the SpO2 version. Indeed, the accelerometer's code requires less iterations than pulse oximeter one, because it has been possible to implement the downsampling before the IIR filtering. In the code for the SpO2 instead, the IIR filter has to be performed on the whole amount of sampled data, because downsampling cannot be anticipated in order to reach good accuracy; this increase the required execution time and power consumption. This difference has emerged since the Matlab-testing preliminary phase.

A possible explanation could be found in the nature of the pulse oximeter signal: in this waveform the heart rate(HR) component is overwhelming, and some subharmonics of the heart rate can also appear in the breathing sector of interest. This forced us to filter opportunely the data before reducing the sampling frequency, in order to remove as many unwanted frequencies as possible.

6.2 Future development

The future developments for this thesis can follow two different patterns: one related to the algorithms implementation, the other one related to the protocol.

With regard to the protocol, the main improvement to do might by the implementation of the real-time calculation. Indeed, so far the computing about the algorithms have been carried out by the monitoring station in a "static way", since the outcome is obtained from pre-stored data. Thus, the first enhancement could be to modify the code in order to better integrate the calculation with the wireless protocol and so, periodically compute the algorithms by using constant refreshed data.

This means that the node which performs the calculation must receive the data from an other node via wireless, run the algorithm, refresh the data buffer with new wireless data, compute again the algorithm and so on, over and over. Obviously, all these task must be accomplished maintaining the protocol synchronization.

This goal can be easily be achieved on the node which implements the accelerometer version of the algorithm, because it requires less amount of data, and it does not override the input samples from the previous calculation. The issue is a bit harder speaking about the implementation of the pulse-oximeter version, because it requires much more data samples (RAM is almost completely full) and, moreover, the IIR filter overwrites the input data buffer since the first stage.

Bibliography

[1] **S.G. Fleming, L. Tarassenko**, *"A Comparison of signal processing techniques for the extraction of breathing rate from the photoplethysmogram"*, World Academy of Science, Engineering and Technology, 2007.

[2] **L. Nilsson, A. Johansson, and S. Kalman**, *"Respiration can be monitored by photoplethysmography with high sensitivity and specificity regardless of anaesthesia and ventilatory mode"*, Acta Anaesthesiol. Scand., vol. 49, no. 8, pp. 1157–62, Sep. 2005.

[3] **L. Nilsson, T. Goscinski, A. Johansson, L. G. Lindberg, and S. Kalman**, *"Age and gender do not influence the ability to detect respiration by photoplethysmography"*, J. Clin. Monit. Comput., vol. 20, no. 6, pp. 431–6, Dec. 2006.

[4] **L. Nilsson, T. Goscinski, S. Kalman, L.G. Lindberg, and A. Johansson**, *"Combined photoplethysmographic monitoring of respiration rate and pulse: a comparison between different measurement sites in spontaneously breathing subjects"*, Acta Anaesthesiol Scand 2007; 51: 1250–1257

[5] **K. Nakajima, T. Tamura, and H. Miike**, *"Monitoring of heart and respiratory rates by photoplethysmography using a digital filtering technique"*, Med. Eng. Phys., vol. 18, no. 5, pp. 365–72, Jul. 1996.

[6] **J. Lee, K.H. Chon**, *"Respiratory rate extraction via an autoregressive model using the optimal parameter search criterion"*, Annals of Biomedical Engineering 2010

[7] **P. Addison, J. Watson**, *"Secondary wavelet feature decoupling (SWFD) and its use in detecting patient respiration from the photoplethysmogram"*, in Engineering in Medicine and Biology Society. Proceedings of the 25th Annual International Conference of the IEEE, Sep. 2003, pp. 2602–5.

[8] **P. Leonard, T.F. Beattie, P.S. Addison, and J.N. Watson**, *"Standard pulse oximeters can be used to monitor respiratory rate"*, Emerg. Med.J., vol. 20, no. 6, pp. 524–5, Nov. 2003.

[9] **P. Leonard, N.R. Grubb, P.S. Addison, D. Clifton, and J.N. Watson**, *"An algorithm for the detection of individual breaths from the pulse oximeter waveform"*, J. Clin. Monit. Comput., vol. 18, no. 5–6, pp. 309–12, Dec. 2004.

[10] **P.A. Leonard, J.G. Douglas, N.R. Grubb, D. Clifton, P.S. Addison,and J. N. Watson**, *"A fully automated algorithm for the determination of respiratory rate from the photoplethysmogram"*. J. Clin. Monit. Comput.,vol. 20, no. 1, pp. 33–6, Feb. 2006.

[11] **K.H. Shelley , A.A. Awad** , *"The use of joint time frequency analysis to quantify the effect of ventilation on the pulse oximeter waveform"*, Stout RG et al., J Clin Monit Comput 2006; 20: 81–7.

[12] **Chon K.H., S. Dash, and K. Ju** , *"Estimation of respiratory rate from photoplethysmogram data using time–frequency spectral estimation"*, IEEE Trans. Biomed. Eng. 56(8):2054–2063, 2009.

[13] **H. Wang, K. Siu, K. Ju, and K.H. Chon** , *"A high resolution approach to estimating time-frequency spectra and their amplitudes"*, Annals of Biomedical Engineering, Vol. 34, No. 2, pp. 326–338, February 2006

[14] **K.H. Shelley, D.H. Jablonka, A.A. Awad, R.G. Stout, H. Rezkanna, D.G. Silverman** , *"What is the best site for measuring the effect of ventilation on the pulse oximeter waveform?"*, International Anesthesia Research Society; 103:372–7, 2006

[15] **T. Reinvuo, M.Hannula, H. Sorvoja, E. Alasaarela and R. Myllylä** , *"Measurement of Respiratory Rate with High-Resolution Accelerometer and EMFit Pressure Sensor"*, IEEE Sensors Applications Symposium Houston, Texas USA, 7-9 February 2006

[16] **D.S. Morillo, J.L.Rojas Ojeda, L.F.Crespo Foix, D.Barbosa Rendón, A.León** , *"Monitoring and Analysis of Cardio Respiratory and Snoring Signals by using an Accelerometer"*, Conference of the IEEE EMBS, Lyon, France August 23-26, 2007.

[17] **D.H. Phan, S. Bonnet, R. Guillemaud, E. Castelli, N.Y. Phan Thi** , *"Estimation of respiratory waveform and heart rate using an accelerometer"*, 30th Annual International IEEE EMBS Conference Vancouver, Canada, August 20-24, 2008

[18] **A. Jin, B. Yin, G. Morren, and H. Duric** , *"Performance evaluation of a tri-axial accelerometry-based respiration monitoring for ambient assisted living"*, Conf. Proc. IEEE Eng. Med. Biol. Soc., vol. 1, pp. 5677–80, Jan 2009.

[19] **A. Bates, M.J. Ling, J. Mann and D.K. Arvind** , *"Respiratory rate and flow waveform estimation from tri-axial accelerometer data"*, 2010 International Conference on Body Sensor Networks

[20] Technical documentation available at http://www.mathworks.com .

[21] **B. O'Flynn, P. Angove, J. Barton, A. Gonzalez, J. O'Donoghue, and J. Herbert**, *"Wireless Biomonitor for Ambient Assisted Living"*, Proceedings of International Conference on Signals and Electronic Systems, ICSES'06 , Lodz, Poland ,September 2006;

[22] **F. L. Lewis**, *"Wireless Sensors Network"*, The University of Texas at Arlington ;

[23] **Riccardo Puppo**, *"Real-Time Fall Detection Systems in Wearable Devices for Elder People"*, Master degree thesis, University of Genoa, 2009;

[24] **Arne Martin Holberg, Asmund Saetre** , *"Innovative Techniques for Extremely Low Power Consumption with 8-bit Microcontrollers"*, Atmel technical documentation

[25] **Bahareh Gholamzadeh, Hooman Nabovati**, *"Concepts for Designing Low Power Wireless Sensor Network"*, World Academy of Science, Engineering and Technology, 2008

[26] Technical documentation available at http://www.atmel.com .

[27] Technical documentation available at http://www.ti.com .

[28] **P. H. Chou, and Chulsung Park**, *"Energy-efficient platform designs for real-world wireless sensing application"*, in Proc. 2005 IEE/ACM International Conf. Computer-aided design, San Jose, 2005.

[29] **Holger Karl, and Andreas Willig**, *"Protocols and Architectures for Wireless Sensor Networks"*. John Wiley & Sons, 2005, pp. 15-329.

[30] **Chris Townsend, Steven Arms**, *"Wireless sensor network: principle and applications"*, Microstrain Incorporated documentation, chapter 22

[31] Technical documentation available at http://www.tyndall.ie .

[32] **John H. Davies**, *"MSP430 Microcontroller Basics"*. Newnes, August 2008, pp. 21-590 (ISBN: 978-0-7506-8276-3).

[33] **Chris Nagy**, *"Embedded Systems Design using the TI MSP430 Series"*. Newnes, 2003, pp. 25-290 (ISBN: 0-7506-7623-X).

[34] Technical documentation available at http://www.analog.com .

[35] Open documentation available at http://en.wikipedia.org .

[36] **S. Drude**, *"Requirements and application scenarios for body area networks"*. Mobile and Wireless Communications Summit, 16th IST, pp.1-5, 2007.

[37] **Stevan Marinkovic, Christian Spagnol, Emanuel Popovici**, *"Energy-Efficient TDMA-based MAC Protocol for Wireless Body Area Networks"*. 3rd International Conference on Sensor Technologies and Applications, 2009.

[38] **Victor Cionca, Thomas Newe, Vasile Dădârlat**, *"TDMA Protocol Requirements for Wireless Sensor Networks"*. The 2nd International Conference on Sensor Technologies and Applications.

[39] **Fabio Di Franco, Christos Tachtatzis, Ben Graham, Marek Bykowski, David C. Tracey, Nick F. Timmons and Jim Morrison**, *"Current Characterisation for Ultra Low Power Wireless Body Area Networks"*. Workshop on Intelligent Solutions in Embedded Systems (WISES), Heraklion, Crete, Greece, July 2010.