POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Master of Science in Telecommunications Engineering

# ANALYSIS AND IMPLEMENTATION OF A VIRTUAL TRAINER FOR THE BENCH PRESS PRACTICE BUILT ON A WIRELESS SENSOR NETWORK

Supervisor: Prof. Antonio Capone

Assistant Supervisor: Eng. Alessandro Redondi

Master Graduation Thesis by:
Alessandro Sciuto
Student ID 734546

Academic Year 2010 - 2011

POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Specialistica in Ingegneria delle Telecomunicazioni

# STUDIO E IMPLEMENTAZIONE DI UN VIRTUAL TRAINER PER IL SOLLEVAMENTO PESI SU PANCA PIANA BASATO SU UNA RETE DI SENSORI

Relatore: Prof. Antonio Capone

Correlatore: Ing. Alessandro Redondi

Tesi di Laurea Specialistica di:
Alessandro Sciuto
Matricola 734546

Anno Accademico 2010 - 2011

# Abstract

Wireless Sensors Networks (WSN) have widely expanded during the last decade within a huge range of applications, among which Sport Applications. This dissertation focuses on a WSN employed as a Virtual Trainer in a Bench Press exercise.

The work has been divided into two main parts: in the first section, a network protocol has been implemented, in order to allow many similar WSNs to simultaneously transmit at the lowest transmission Power Level possible without interfering with each others. In the second section, the application algorithm is presented, which consists of acquisition, processing and analysis of the data coming from a 3-axis accelerometer. The user's Training Profile is estimated with a known "force-velocity" model and used to evaluate the user's performance.

Both building blocks are eventually tested and experimentally evaluated.

# Sommario

Negli ultimi anni, le reti di sensori (Wireless Sensor Networks) hanno trovato applicazione in un ampio numero di settori, tra cui quello sportivo. Il lavoro proposto presenta una rete di sensori utilizzata come Virtual Trainer in un esercizio di sollevamento del bilanciere su panca piana.

La Tesi é divisa in due blocchi: nella prima, viene descritto un protocollo di rete che mira a consentire la coesistenza di un gran numero di reti di sensori che trasmettono contemporaneamente minimizzando allo stesso tempo la potenza in trasmissione. Nella seconda parte, invece, viene descritto l'algoritmo applicativo, il cui compito é quello di acquisire, processare e analizzare i dati provenienti da un accelerometro a tre assi. Il profilo d'allenamento (Training Profile) dell'atleta é stimato mediante un noto modello "forza-velocitá" ed é impiegato per valutarne le performance.

Infine, alcuni test sono stati effettuati su entrambe le parti presentate al fine di misurarne l'efficienza.

# Contents

# Chapter 1

# Introduction

In the later years, the scientific community has highly focused on the development of tiny low-cost wireless devices. These efforts brought to the growth, the expansion and the success of the Wireless Sensor Networks (WSNs). A Wireless Sensor Network is a group of low-cost low-speed and low-power sensors (often called *motes*) able to sense data from the environment and to transmit them via wireless links.

Although the peculiarities and the features of this network were well-suitable for a wide range of applications, at first these devices were almost used only in Military Applications. However, the easy scalability, the low complexity and the high customization allow the integration of this network in more area such as Environment, Home, Healthcare or Sport applications.

This Master Thesis presented a Body Sensor Network designed and developed for a Sport Applications. A Body Sensor Network (BSN) is a group made of wearable motes such as the one just described. These sensors were initially developed for healthcare applications, but their small size, light weight and low complexity made them very useful also in a sport environment.

Indeed, Virtual Trainers revolutionized the training methods of both professional and amateur athletes. Nowadays, the gap between a winner and a loser in the high level competitions is really small. Improving the performance of few milliseconds or few meters per second is the goal of the most part of trainers and coaches. This can be achieved with the aid of these devices that, unlike the systems developed inside laboratories or “off-court” environments, allows the athletes to keep training in their usual context (such as a golf pitch or a swimming pool), obtaining more meaningful data than the ones available with the older techniques. Besides, amateur athletes or people that sporadically train, can take advantage of the virtual trainers in order to learn the basic movements of a specific sport or gym exercise, replacing sometimes the "real" trainers who can be often unavailable like, for instance, during peak-time in a crowded gym.

The aim of the WSN proposed is to obtain a Training Profile and a Power Evaluation of the user in a Bench Press exercise. This WSN consists of two motes, a Base Station connected to a PC and a mote with an integrated accelerometer attached to the barbell. This work was divided into two different sections.

In the first part a Network Protocol was designed and built in order to improve the Quality of Service (QoS) when a large number of wireless networks is contemporaneously transmitting, using, though, the lowest power level possible to minimize the power consumption. In this protocol, the BS, after evaluating the Noise Floor values of all the radio channels, selects the best one and starts sending broadcast beacon messages. When the mote receives this message, associates with the BS replying with an association packet and waiting for the relative ACK. Eventually, the two motes choose the Tx Power Level according to the QoS parameters decided.

In the second part the Application is introduced, including the Data acquisition, sending and processing. It shows how to get meaningful information from the accelerometer readings, focusing then on the practical purpose of this Thesis. It explains, in fact, what the two outputs mean, how to obtain and how to analyse them.

The outcome of the work will be deeply reported and explained in the rest of the dissertation:
Chapter 2 introduces the Wireless Sensor Networks, the main features and the common applications.
Chapter 3 presents a detailed description of the Software and the Hardware components employed during the whole work.
Chapter 4 is an overview on the history and the development of WSNs in the Sport environment, and a couple of practical examples are also presented.
Chapter 5 describes the first main part of this Master Thesis, that is the Network Protocol.
Chapter 6 is about the second part of this work, the Application one, that ex-

plains the outputs and the aim of the WSN proposed.
Chapter 7 illustrates the performance of the WSN, for both the Network Protocol and the Application algorithms. Finally, in Chapter 8 some considerations about the implemented network are presented, along with possible future developments.

# Chapter 2

# Wireless Sensor Networks

Wireless Sensor Network refers to a group of spatially dispersed and dedicated sensors. This technology greatly expanded at the beginning of the 21st century, and its features fit for a wide range of feasible applications, such as military applications, environment monitoring, surveillance, health care and virtual trainer in sport. Examples of monitored data range from simple measurements, like temperature or humidity values, to more elaborated information, like accelerations or vibrations.

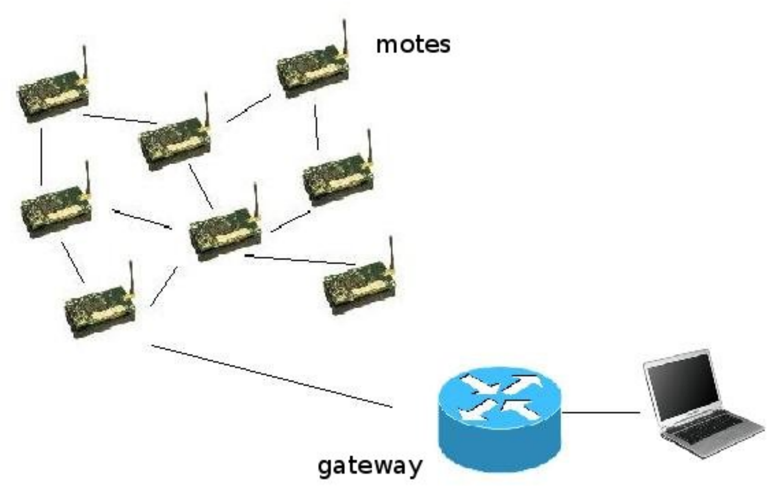


Figure 2.1: Typical Wireless Sensor Network architecture

## 2.1 Sensors

A sensor node (also known as “mote”) is a device capable of processing, communicating and gathering data from the environment. These devices have already been used in decades, but the project of the present sensor nodes has been started in the 1998 by the DARPA ( “Defense Advanced Research Projects Agency”) and the NASA (“National Aeronautics and Space Administration”) .

Typically, the modern mote consists of a controller, a transceiver, RAM and flash memories, power source and one or more sensors[1]. The controller performs tasks, data processing and supervises the other components. Usually, the controller is actually a microcontroller (although other options are for instance desktop microprocessor or digital signal processors) because of its low cost and power consumption and the ease of programming. The transceiver has four operational states: transmit, receive, idle and sleep. It transmits and receives data over the radio channel. Apart from the infrared and lasers, it needs an antenna in order to communicate within the radio frequency space, where WSN tends to make use of license-free communication frequencies. The RAM memory is used to store dynamic data, while the flash one to save long-lived data and the execution code. The power source provides power the mote needs to sense, communicate and process the data. It can be stored either in batteries or capacitors. Eventually, according to the goal is designed for,the mote can be equipped with one or more sensors. These sensors are hardware devices that can be classified into passive active ones. Passive sensors do not need active probing to sense the data and they are self powered. The active ones, instead, probe the environment and need energy from a power source. An example of passive sensor is a PIR (Passive Infrared Sensor), while a camera is an active one. Figure 2.2 shows some examples of motes.

## 2.2 Network

In a WSN, the intercommunication between nodes can be achieved with different approaches, according to the purpose and the specific tasks the sensors are designed for. The range of factors that may influence a WSN design is wide, and it includes *fault tolerance, scalability, sensor network topology, hardware constraints and power consumption* [2]. For instance, sensors may break down due

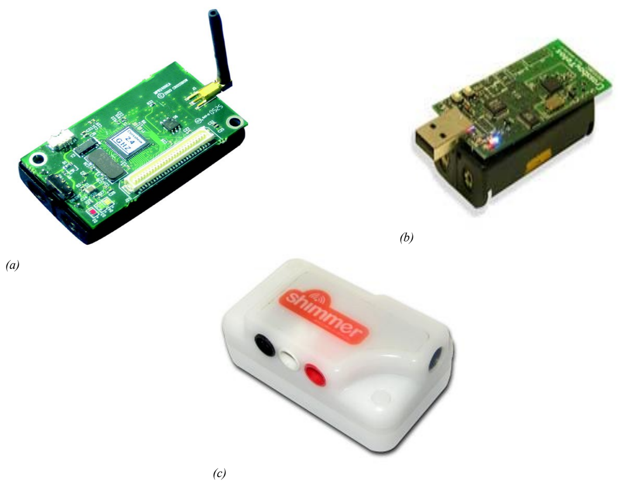


Figure 2.2: Different motes: (a) MICAz - (b) TelosB - (c) Shimmer

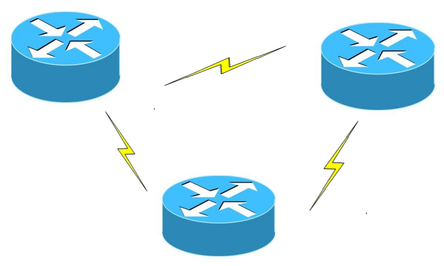
for example to a lack of power, but this failure should not affect the whole network task. Moreover, there are transmission techniques (e.g. Bluetooth) that are not efficient when a large number of nodes is employed or other ones, such as Infrared, that are tolerant to interference but requires LOS (Line of Sight) transmission.

The four main WSN topologies are:

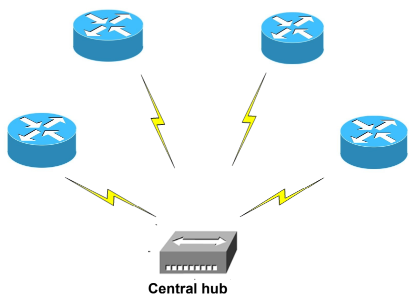
- Point-to-Point;
- Star;
- Tree;
- Mesh.

The Point-to-Point network (also know as "Peer-to-Peer") allows each node (or peer) of the network to communicate with another node without going through a centralize hub (as shown in Figure 2.3a).

The Star network consists in one or more sensors linked to a centralized hub, that is connected with every node of the network, while the latter are not able to communicate with each others (Figure 2.3b).



(a)

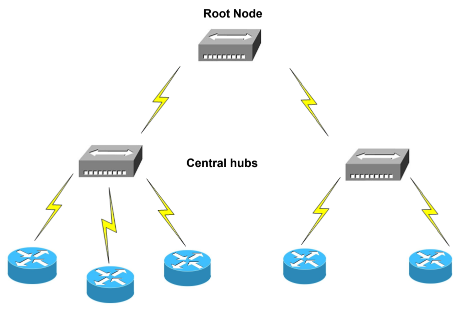


(b)

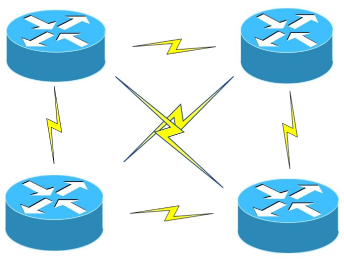
Figure 2.3: Point-to-Point (a) and Star (b) topology

The Tree network is a hierarchical network. At the top of the tree there is a node, called *Root Node*, that is the main router of the network. One level down, there are one or more central hubs connected to the node of the level above. These hubs are then linked to one or more nodes, as happens in the star network. Indeed, the tree topology could be seen as an hybrid of both the two already described before. An example of the Tree network is shown below in Figure 2.4a.

In the Mesh network, instead, each node may communicate with every other node of the network, allowing data to "hop" from one node to another. This "freedom" makes the Mesh topology one of the most complex and most expensive one. A typical model of Mesh network is illustrated in Figure 2.4b.



(a)



(b)

Figure 2.4: Tree (a) and Mesh (b) topology

### 2.2.1 IEEE 802.15.4 Standard

The majority of WSN adopts the IEEE 802.15.4 Standard. This protocol provides only the lower network layers, MAC (Media Access Control) and the PHY (PHYsical) layers, of a personal area network (WPAN) mainly used for low-cost low-power low-speed ubiquitous communication[3]. The idea of IEEE 802.15.4 Standard is to allow communications between devices up to 10 meters with a rate of 250 kbps. Key features of this standard are also real-time suitability and collision avoidance (CSMA/CA).

The PHY layer handles the data transmission service, operating in one of the unlicensed frequency bands:

- 868.0-868.6 MHz (Europe);
- 902-928 MHz (North America);
- 2400-2483.5 MHz (Worldwide);

The MAC layer transmits MAC frames, which structure is very flexible, in order to satisfy the needs of the different tasks and network topologies, through the physical channel [4]. In a beacon-enable network, the slotted CSMA/CA is adopted, while in a non beacon-enable one it utilizes the unslotted CSMA/CA.

## 2.3 Applications

It is possible to classify WSN applications in several categories, as follows.

### 2.3.1 Military

At the beginning, WSN development was especially focused on military purposes. Battlefield surveillance, like area monitoring for detecting enemies intrusion, was one of the first uses of this technology. In this scenario fault tolerance, self-organizing capacity and rapid deployment are crucial. Moreover, the low-cost and the high density of the motes make sure that destroying part of the nodes does not affect the functionality of the whole network, that is why WSNs are widely used in military applications. Indeed, apart from battlefield surveillance, they are employed in: targeting and target tracking systems, nuclear, biological or chemical attack detection.

### 2.3.2 Environment

Area monitoring is a common use of WSNs for military operations as much as for environment intents. Motes sensing may detect and generate alarms for particular events, such as fires or flood, but they could also be useful for precision agriculture, tracking of movements of small animals or pollution studies.

### 2.3.3 Home

Power consumption control is really important in WSN home applications. In fact, it is possible and easy to save energy monitoring and controlling the domestic appliances and the other home equipments just making them communicate with one sensor attached to each of this device and letting them interact (e.g. with internet connection).

### 2.3.4 Health

WSNs can find place also in the healthcare area. This technology, for example, can be employed in hospital, where it is possible to track and monitor patients. Wearable sensors are largely used for this reason as much as for getting patients' heartbeat or patients' unexpected behaviours (like a fall or some kinds of limb movements in order to detect convulsions). Wearable sensors (it will be widely shown in Chapter 4) are also used in many sports as virtual trainers to improve athletes' performance.

# Chapter 3

# Software and Hardware

This Chapter will discuss all the software and hardware components used to implement this work. At first, the software ones will be introduced, where an overview of the TinyOS operative system will be presented. Afterwards, the different hardware devices will be described in details.

## 3.1 TinyOS

TinyOS [5] is an open-source operative system developed by the Univerisity of California at Berkeley and designed for low-power wireless devices. However, there are some differences between the typical operative system and TinyOS. The main one is that the first has a kernel[1] that interfaces all the components, while the latter offers a direct access to the hardware. Furthermore, TinyOS allows only static memory allocation, guaranteeing time and space efficiency.

The first version was released on October 2002, and it was followed by many updates until the last 1.15 version on December 2005. The first 2.x version came out on February 2006, with a completely new software, simpler and more editable, capable to be adapted to different hardware platforms.

TinyOS has been implemented in *NesC*, a programming language created for the developing of embedded system applications that will be better described in the next section.

[1]The kernel is a program that constitutes the central core of a computer operating system. It has complete control over everything that occurs in the system.

## 3.2 NesC

NesC (“Network Embedded Systems C”) is a dialect of the C programming language, which allows the development of robust and modular embedded systems. Its main features are:

- *Separation of construction and composition:* in a NesC application, users may assemble pre-existent components with others created ad-hoc. These components are “wired” in order to build whole programs and have to define and implement their specifications.
- *Component specifications through interfaces:* a component may either provide or use interfaces. The first ones describe which functionalities the component provides to its user, the latter are the interfaces it needs to obtain the desired output.
- *Interfaces are bidirectional:* interfaces implement two different sets of functions: one to be implemented by the interface’s provider (*commands*) and the other one to be implemented by the interface’s user (*events*).
- *Static structure:* components are statically interconnected to each others, and this structure can not be edited run-time.
- *Concurrency:* the NesC concurrency model (as the TinyOS one) makes sure that the processes execute their activities on shared data at the same time.

### Components and interfaces

As stated earlier, a NesC application consists of one or more components *wired* to form an application executable. A component is a code block that executes some operations that can “provide” or “use” interfaces. An interface is a file within are listed a set of events and commands. Components may be *modules* or *configurations*. Modules implement one or more interfaces and their functions and events, while Configurations wire together all the components used in the application, connecting the interfaces a component uses with the ones other components provide.

### Commands and events

A interface's command can be called by a component only if it uses that interface. Indeed, it is declared in the interface file, it requires input parameters and returns an output value. They can be invoked with the `call` primitive. A command may also not return an output, as in the *void* functions. For instance, in order to start a timer that fires every "TIME" *ms*, the code line is:

```
call Timer<TMilli>.startPeriodic(TIME);
```

Events, instead, represent hardware interruption managers. They must be implemented by the application that uses the module. The hardware interruptions may are due to internal reasons, like a fired timer, or to external ones, like the reception of a message. The syntax is similar to the commands one, but this time the primitive used to invoke the event is `signal`. Both commands and events can have an *asynchronous* or *synchronous* concurrency behaviour. The asynchronous event or command is generated by a hardware interruption that, once it has been triggered, starts immediately delaying the execution of the program, that will continue from the point it has been stopped right away the end of the interruption. The synchronous one is the default type: it is not able to manage hardware interruptions, it can be stopped by an asynchronous event and can be called only by tasks.

### Tasks

Tasks are a set of instructions that can be execute without been stopped by any other command. This allows tasks to run even while a node is executing other functions: indeed, they are useful when a mote has to compute heavy and long operations. Tasks enter in a tasks queue that works with a FIFO (First In First Out) policy. This behaviour avoids race conditions on shared variables. An example of task declaration is:

```
task void taskname (){
...
}
```

where *taskname* is whatever name is assigned to the task, and it can be dispatched with the sintax below:

post *taskname* ()

.

## 3.3 Hardware Abstraction Architecture

The Hardware Abstraction Architecture (HAA) for TinyOS 2.0 is an hardware abstraction model which is supposed to balance the requirements of code reusability and portability as well as the optimization and the performance efficiency [6]. As shown in Figure 3.1, it includes three main layers:

- Hardware Presentation Layer (HPL);
- Hardware Abstraction Layer (HAL);
- Hardware Interface Layer (HIL).

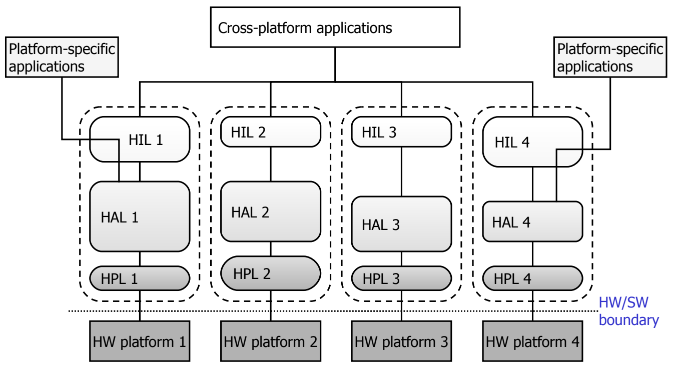


Figure 3.1: Hardware Abstraction Architecture in TinyOS

#### Hardware Presentation Layer

It is the lowest level in the model, and its components are placed directly over the HW/SW interface. It communicates with the hardware offering to the components above a simpler interface with more usable function calls. HPL components are stateless and are able to initialize, start and stop hardware modules, to get and set the operations in the control registers, to enable/disable interrupts and to provide service routines for the generated interrupts.

#### Hardware Abstraction Layer

The components of this layer represent the core of the architecture. They take advantage of the raw interfaces offered by the HPL in order to build abstractions that hide the complexity of the hardware resources. Unlike the HPL components, the HAL ones can maintain state in order to perform resource control if needed. The abstractions provided at this level are simpler than the HPL ones, even if the HAL is still dependent on the hardware layer.

#### Hardware Interface Layer

This level provides to the application level hardware-independent functions. These functions allow the abstractions offered by the hardware layer to run on different platforms. The complexity of the HIL components is represented by the trade-off between the capabilities of the hardware and the current "API contract"[2]. If the first exceeds the latter, the HIL "downgrades" the platform-specific applications provided by the HAL in order to level-off with the chosen interface.

## 3.4 Hardware

In this work, two different types of "motes" have been used: MICAz and Shimmer. The features of both will be presented in the following sections.

### 3.4.1 MEMSIC MICAz

The MEMSIC MICAz [7] is a third generation device used for low-power,

[2]An application programming interface (API) is a source code-based specification intended to be used as an interface by software components to communicate with each other.

wireless sensor network (Figure 3.2) that works in the unlicensed 2.4 GHz frequency. It is a 58 $\times$ 32 $\times$ 7 mm mote, powered by two AA batteries. Its processor

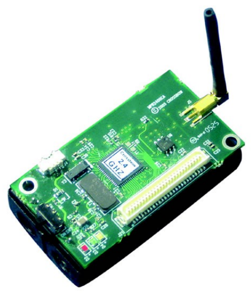


Figure 3.2: The MEMSIC MICAz mote

is based on the *Atmel ATmega128L*, a low-power microcontroller equipped with 128 KBytes of flash memory, 4 KBytes of SRAM[3], a 51-pin expansion connector and its peak operating frequency is 16 MHz. The transceiver is a single radio chip, CC2420, which is well-adapted to the IEEE 802.15.4 standard (2.2.1). In order to connect the mote to a PC and install the application, a programming board is needed. It could be a USB board (MIB520) or a serial one (MIB510), as shown in Figure 3.3.

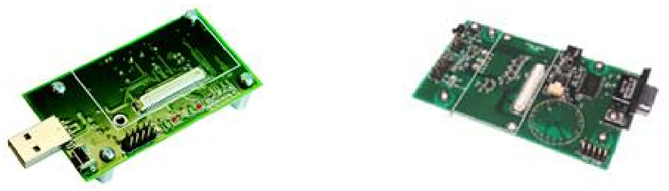


Figure 3.3: A MIB520 (left) and a MIB510 (right) Programming Board

## Radio

As just mentioned, CC2420 is the radio chip used by the MICAz family. The

[3]The SRAM (Static Random Access Memory) is a type of semiconductor memory where the word static indicates that, unlike dynamic RAM (DRAM), it does not need to be periodically refreshed

| **Power Level** | **Power [dbM]** | **Current [mA]** |
|---|---|---|
| 31 | 0 | 17.4 |
| 27 | $-1$ | 16.5 |
| 23 | $-3$ | 15.2 |
| 19 | $-5$ | 13.9 |
| 15 | $-7$ | 12.5 |
| 11 | $-10$ | 11.2 |
| 7 | $-15$ | 9.9 |
| 3 | $-25$ | 8.9 |

Table 3.1: CC2420 Power Levels

CC2420 is a true single-chip 2.4 GHz IEEE 802.15.4 compliant designed for low-power and low-voltage wireless applications [8]. The CC2420 supports a 250 kb/s data rate with 16 channels.The RF frequency of these channels is given by the equation:

$$F_{ch} = 2405 + 5 \times (k - 11) \quad \text{MHz}, \qquad k = 11, 12, ..., 26$$

The radio has 4 different working states: *sleep*, *idle*, *receiver* and *transmitter*. If the radio is in the first state, the chip is disabled. In the second one, the chip is still not busy, but it is just partially turned off, ready to switch immediately on if needed. The last two states are the activity states. The CC2420 can transmit at 8 different power levels, and table 3.1 shows the correlation between the power level, the power in dbM and the current draw.

## 3.4.2 Shimmer

Shimmer is a small sensor platform well-suited for wearable applications [9]. The core element in this mote is the low-power *MPS430F1611* microprocessor, that is in charge of controlling all the device operations. It provides 48 Kbytes of program flash memory, 256 Kbytes of data flash memory and 10 Kbytes of RAM. It is also equipped with an integrated 3-axis accelerometer and with a MicroSD slot with capacities up to 2 Gbytes that allows large storage and long-term data acquisitions. It has a long operating life, because of its 280 mAh battery, that can be easily rechargeable with the provided USB External “Dock” (Figure 3.4).

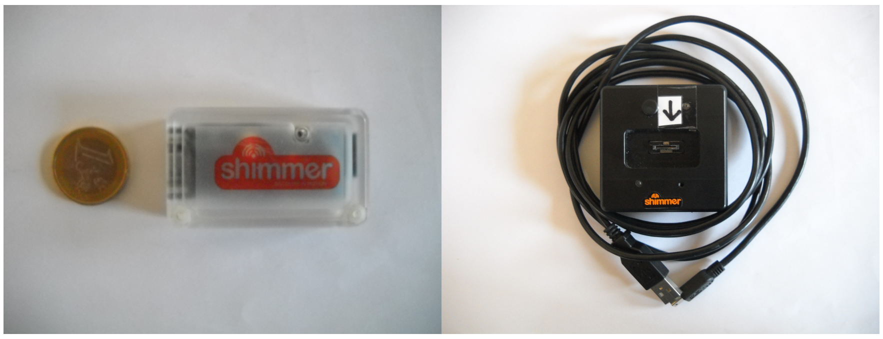

(a) (b)

Figure 3.4: (a) The Shimmer Sensor Mote - (b) The USB External "Dock"

### Radio Overview

For wireless data streaming this platform presents both Bluetooth [4] and IEEE 802.15.4 radio modules. It utilizes the Roving Networks RN-46 Class 2 Bluetooth module [10] to support the first, and the ChipCon CC2420 radio transceiver just discussed above (3.4.1) for the latter. Depending on the protocol, almost every feature of the Shimmer Platform behaves in a different way, this is why one or the other should be chosen according to the needs of the selected application. The table 3.2 shows the main differences between the two techniques.

| **Feature** | **802.15.4** | **Bluetooth** |
|---|---|---|
| Power Consumption | Better | Worse |
| Agility/Connection Speed | Better | Worse |
| Pre-built Application | No | Yes |
| Number of nodes | Good | Poor |
| Range | Ok | Ok |
| Mesh Implementations | Yes | No |
| Ability to customize | Yes | No |
| FCC Modular Certification | No | Yes |
| Data Rate | Worse | Better |

Table 3.2: IEEE 802.15.4 vs Bluetooth

[4]Bluetooth is a low-cost, low-power, robust, short-range wireless communication protocol which was founded by Ericsson in 1994 in order to replace the traditional phone and computer cables with wireless links.

### Accelerometer

An accelerometer is a device that measures is own proper acceleration, that is the acceleration relative to an inertial observer who is at rest relative to the object being measured. The quantity measured is known as *g-force*, which corresponds to the acceleration of the object relative to free-fall. Because of this characteristics, accelerometers are well-suited for detecting sense orientation, vibration, shock or falling. An accelerometer may be single or multi axis: in the Shimmer motes employed in this work, a Freescale MMA7260Q [18] 3-axis accelerometer is integrated (Figure 3.5).

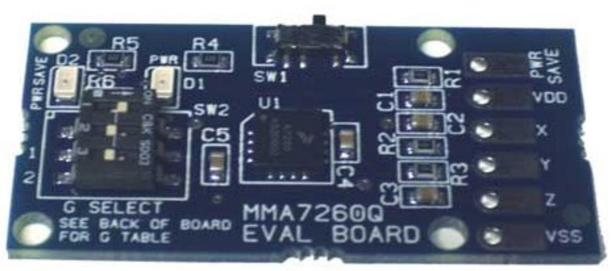


Figure 3.5: Freescale MMA7260Q 3-axis Accelerometers integrated in Shimmer motes

The main features of this device are: the flexibility to select 1.5g,2g,4g or 6g of acceleration range, low power consumption, low component count, high sensitivity, high frequency and high resolution for fall, tilt, motion or vibration sensing.

# Chapter 4

# WSN in Sport: a survey

In the last decade, athletes' performance has been helped by training as much as by science. Indeed, the difference between a winner and a loser is getting thinner and thinner, and the aid of a technology support or device may be vital to "cross the line" that divides an average player from a champion. Wireless Sensor Networks fit perfectly in this scenario: the easiness of the topology, along with the opportunity to attach the sensors directly on the athlete's body without caring about cables or wires, makes the WSNs widely used in a lot of different sports. After a brief introduction to Body Sensor Networks, this Chapter will present an overview on WSNs sport applications with a special care for the one concerning this work.

## 4.1 Body Sensor Networks

A Body Sensor Network (BSN), also called Body Area Network (BAN), describes a network made up of wearable computing devices. At first, BSN devices were developed only for healthcare applications. Indeed, with their flexible and compact design, they became a key element in order to obtain a more proactive and affordable approach to healthcare area [11] [12].

Since the number of elderly over age 65 is expected to double in about ten years' time [13], researches and projects around BSNs notably increased, because they were supposed to represent the ideal support for monitoring patients, especially aged people in ambulatory or rest home settings. However, the researches have continued, looking for even more feasible scenarios and useful applications in the medical field. As a result, in the later years there has been a remarkable increase in the number (and in the variety) of wearable sensor devices for health monitoring, ranging from simple pulse monitors to much more expensive implantable sensors. Figure 4.1 shows a simple example of a BSN application.

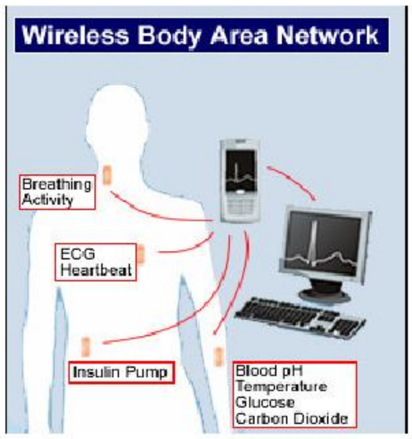


Figure 4.1: Example of BSN for healthcare monitoring

During the development and testing of this technology, there have been some medical and social challenges to go through such as:

- *Interoperability and Scalability:* BSNs should ensure seamless data transfer across the most common WSN protocol (Bluetooth, ZigBee[1], IEEE 802.15.4 standard), plug-and-play[2] device interaction and high scalability in order to provide all the functionalities regardless of the number of nodes actually connected to the network;

- *System devices:* the motes employed would have to be small sized, light, with low power consumption and low in complexity. Moreover, especially for long-term usage, they should be equipped with additional data storage when real-time connection is not available.

[1]ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks.

[2]In computing, plug-and-play is a term used to describe the characteristic of a computer bus, or device specification, which facilitates the discovery of a hardware component in a system, without the need for physical device configuration, or user intervention in resolving resource conflicts.

- *Invasion of privacy:* some patients may look at BSNs as a threat for their privacy. In order to go beyond this sceptical view, the network and the devices used have to ensure secure and safe data transmission.
- *Data consistency:* since there might be several nodes attached to each patient, all data have to be collected and examined in such a way that, even if they are transmitted across more than one mote or gathered in more than one PC or similar, they will eventually provide all the information needed, thus the quality of patient care will not degrade.
- *Interference:* either in an ambulatory or in a hospital or in a rest home scenario, the number of different BSNs can be considerable, this is why the wireless links used need to ensure low interference and high coexistence between devices belonging to different networks.

## 4.2 Virtual Trainers

However the growth of BSNs was due in particular to medical and clinical applications, nowadays they are employed in healthcare as much as in sport. In fact, traditionally, the measurements of athlete performance were done in a laboratory environment. But, even the more sophisticated and advanced laboratory was not able to reproduce exactly the real scenario that the athlete should have faced during his performance (e.g. a football pitch, a tennis court, a swimming pool, etc.) [14]. The most common instrument used to evaluate athletes and players movements was an optical motion tracker [15]. This system could keep track of people movements because of some reflective markers positioned on the body and, with an array of high-speed IR-illuminated video cameras, was able to convert the different positions acquired into joint angles, driving eventually a stick-figure animation. As an example, in the past some measurements were acquired from some baseball players in USA, employing this technique. Apart from the difference with the real sport scenario mentioned just above, the critical issue for most of the players was that the markers were quite uncomfortable, and they could not act like in real games.

Following the success of BSNs, trainers started to use this innovation in place of the older systems, and the results were high remarkable. For instance, a runner gait changes considerably from treadmill running and outside running. Even better, the improvement due to the first wireless virtual trainers was higher for sports like tennis, volleyball or similar, where studying athlete's performance during a proper training or match makes possible to evaluate the mental factors as well as the physical ones. Some specific applications concerning wireless sensors employed as virtual trainers will be now illustrated in the following sections.

### 4.2.1 Golf

The popular sport of Golf requires some detailed and well-executed movements in order to swing the golf club properly. A good and consistent golf swing means a lot (and improves the score) for a golf amateur as well as for a professional. After a deep analysis of the Golf model swing, a BSN has been used in order to detect the most common mistakes people that approach golf make while playing a swing [16].

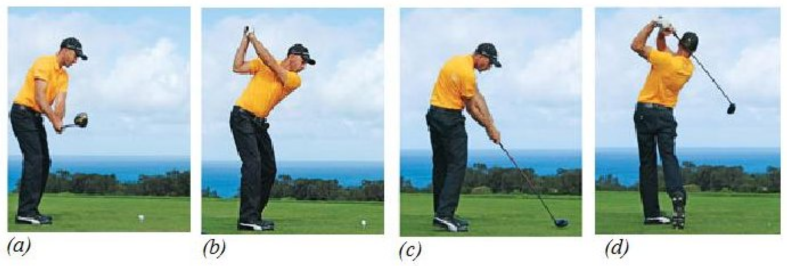


Figure 4.2: The four-phase motion of a golf swing: (a) Takeaway, (b) Backswing, (c) Downswing, (d) Follow-through

Given that a proper swing consists of a four-phase motion (**Takeaway**, **Backswing**, **Downswing**, **Follow-through**) as shown in Figure 4.2, there are two typical mistakes a new player makes, resulting in a bad shot: *wrist rotation* and *out-of-plane* movements. Wrist rotation is the clockwise or counter-clockwise rotation of the wrist, causing a deviation of the ball flight path on the right or on the left of the target respectively. The second mistake occurs when, during the whole swing, the golf club does not remain on a plane at the address position, and can be due to wrong movements like over-bending the elbows, raising the arms too high or not raising the arms enough.

The BSN is structured as follows (Figure 4.3):

- two nodes on the golf club (red dots);
- one on the right wrist (green dot);
- one on the left arm (orange dot);
- one on the back at waist level (blue dot).

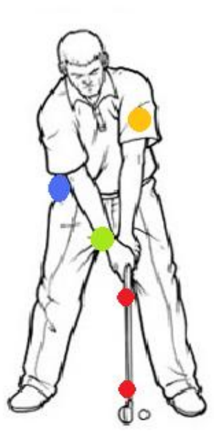


Figure 4.3: BSN for detecting wrist rotation and out-of-plane mistakes in a golf swing

TelosB motes embedded with a tri-axial accelerometer and a bi-axial gyroscope[3] are used for the measurements. The sampling frequency chosen, a trade-off between the bandwidth constraints and a valid samples resolution, is 50 Hz. These sensors transmit all the data to a Base Station (it can be an other mote or a PDA[4]), that is in charge of collecting all data from the other nodes, processing them and providing the final feedback on the golf swing played.

[3]A gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum.

[4]A personal digital assistant (PDA), also known as a palmtop computer, or personal data assistant, is a mobile device that functions as a personal information manager.

Each sensor, before sending the samples acquired to the Base Station, performs a *preprocessing* with a five-point moving average filter, in order to remove the noise. Afterwards, the data collected are manually separated (with the help of a video recording) depending on whether swing segment described in Figure 4.2 they belong to. For each segment, a quantitative model is built, consisting of a PCA [21] and a LDA [22], which outputs the parameters of a linear regression that eventually quantifies the degree of improperness of the movement.

### 4.2.2 Tennis

The service swing for a tennis player is crucial as much as the swing for a golf player discussed in the previous paragraph. In order to perform a powerful serve, the player needs to execute the correct movement with all the joints and the limbs concerned. Anticipated by some studies on the *kinematic model*[5] of the tennis serve, a BSN has been designed to analyse and evaluate this important shot [17].

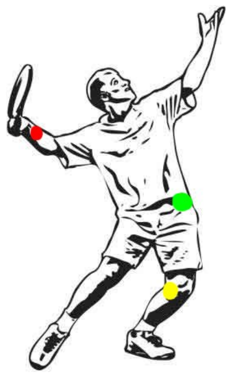


Figure 4.4: BSN with accelerometers for examining a tennis player's serve

The network consists of three different nodes equipped with either ±10g ADXL210 or ±2g ADXL202 accelerometers set as illustrated in Figure 4.4:

- one ADXL202 accelerometer attached on the knee (yellow dot);
- one ADXL202 accelerometer attached on the waist (green dot);

[5]Kinematics is the branch of classical mechanics that describes the motion of points, bodies (objects) and systems of bodies (groups of objects) without consideration of the forces that cause it

- one ADXL210 accelerometer attached on the wrist (red dot).

The sampling frequency is set at 500 Hz, the data are transmitted via a Bluetooth connection to a Base Station and two video cameras are positioned in front of the players to register the whole acquisition. The goal of the experiment is to underline the differences between an elite player serve and an amateur one. Data processing in this case is actually quite simple. Indeed, the yardstick to compare the two categories of player is the average acceleration acquired by the motes, and the standard deviation is also calculated to indicate the repeatability of the serve swings belonging to the same class. As a result, the BSN identified the major differences around the impact time: professional players, in fact, generate more waist torsion as well as more acceleration magnitude of the hand right before hitting the ball.

### 4.2.3 Body Training and the gym scenario

An other scenario where wireless virtual trainers are widely employed is body training. Everyone, from a professional athlete to people who practise sporadically, can profit by the use of these devices during, for instance, a gym session.

Figure 4.5: An example of virtual trainer used for detecting lower limbs explosive strength .

Indeed, they may represent an add-on to improve trainings (as already mentioned above), a “replacement” of real trainers who sometimes can be unavailable (e.g. at peak-time in a gym), or also an useful device in order to avoid injuries.

Consequently, people may save money, time and “pain”: this is why nowadays the use of such devices keeps increasing also in the gyms.

Some examples of applications in this scenario are: detecting upper and lower limbs explosive strength (Figure 4.5), measuring upper and lower limbs power, providing a training profile (Hill’s Curve, explained in details in Chapter 6), training optimization and much more besides. Business around these products is increasing as well, in fact there are some companies which are already developing and selling devices that perform these operations (e.g. Sensorize [19], in Italy, is greatly focused on these types of sensors), but lately more and more semiconductor companies are getting interested on this field too.

The following chapters will describe a WSN projected for an application similar to the one just illustrated in this paragraph: indeed, its goal will be provide a training profile of the user and evaluating the power generated from his (or her) upper limbs during a bench press practice.

# Chapter 5

# Network Protocol

This Chapter will describe the first section of this work, that is a network protocol developed in order to obtain a connection with high *QoS* ("Quality of Service") while transmitting with the lowest possible power level. The WSN topology is quite simple: a single Shimmer mote attached to the barbell and a Base Station in charge of collecting and analysing the data. This protocol is divided in 4 different step:

1. **Channel Scan:** when both the Base Station and the Mote switch on, they start scanning the 16 radio channels (3.4.1): the first one executes several readings of the noise floor of each channel, selecting eventually the channel with the lowest (or more negative) value; the latter listens periodically to every channel until it finds the beacon message of the Base Station;

2. **Beacon Sending:** after that the best channel is selected, the Base Station begins to send beacon frames, waiting for the mote's reply;

3. **Association:** when the mote intercepts a beacon message, replies to the BS with an association message, in order to set a ubiquitous and exclusive connection with the BS;

4. **Power Level Selection:** once the connection has been established, the two sensors select the lowest power level that satisfies the QoS parameters chosen with an ad-hoc algorithm.

## 5.1 Channel Scan

The very first step when the Base Station switches on is the Channel Scan. Indeed, since the WSN presented may be widely employed in crowded places like gyms, and consequently a large number of this sensors might be used at the same time, this protocol must allow to transmit the data in the best channel possible. It means that the packet traffic in the channel has to be as little as possible, thus also the Rssi[1] value must be the lowest possible.

The Rssi is a measure in dBm of the signal power on the radio link. Apart from detecting the transmitting channel QoS, it can be also useful for calculating the distance between two nodes. In the CC2420 radio chip, the Rssi is provided by a platform-specific HAL interface. However in this section the Rssi measurements are not related to a received packet (as it will be discussed in the last section), but they represent **noise floor** readings. The noise floor is the noise given from the sum of all the undesired sources in a specific channel. The lower is this value, the better is the channel.

In order to achieve noise floor readings with the CC2420 chip, a *CC2420ControlP* component is employed, wired to two different interfaces: the *CC2420Config* and the *Read<uint16_t>* interfaces (from now on called `ChannelConfig` and `RssiRead` respectively[2] ). For each channel, 40 readings of the noise value are performed during a 1.2 seconds interval (one reading each 30 ms): indeed, observing the channel for such a relatively long time allows the Base Station to obtain more meaningful results about the quality of the channel listened.

The Channel Scan works as follows: right after the Base Station switches on, and so does its radio, the `startPeriodic` command of the `ListeningTimer` (Timer<TMilli>) interface is called, and every 30 ms it fires in order to perform a noise floor reading. Before starting listening the noise level, the BS has to set its radio on the channel selected. For this reason, the first time that the `fired()` event occurs, it selects the channel calling the following command:

[1]In telecommunications, Rssi ("Received Signal Strength Indication") is a measurement of the power present in a received radio signal.

[2]In nesC, an interface can be renamed with the *as* keyword. In fact, in the same application, more than one instance of the same interface may be used. If so, renaming the interfaces is an useful method in order to avoid confusion and differentiate each interface. From now on, in the discussion the nickname (if it exists) of the interface will be followed by its real name in brackets.

`call ChannelConfig.setChannel(j);`
where $j$ is an `uint8_t` variable that identifies the number of the channel set. However, the channel setting is not over. Indeed, the `sync()` command, always belonging to the `ChannelConfig` interface, must be called. This command, along with the `syncDone( error_t error)` void event, form what in nesC is called a *split-phase* operation [20]. Basically, this means that the synchronization will be completed only when the called abstraction will issue a callback.

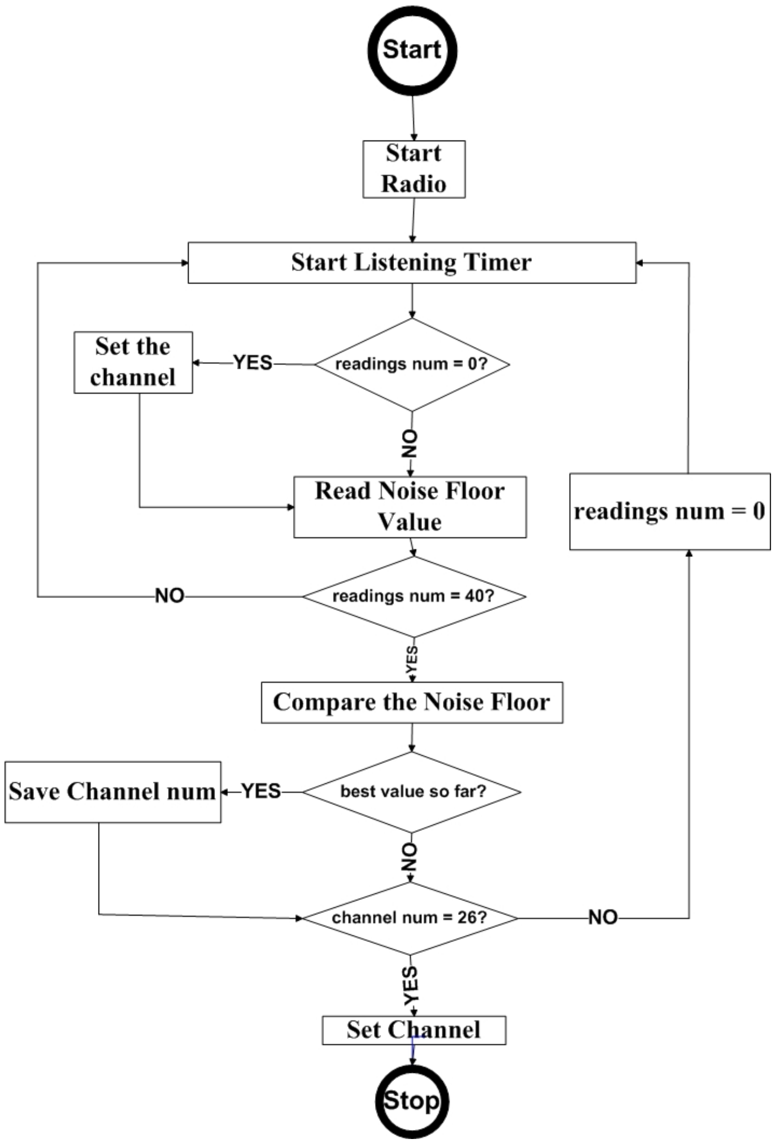


Figure 5.1: Channel Scan Flowchart

When the channel is eventually selected, the BS reads the first noise floor value calling the `read()` command of the `RssiRead` interface. This command is also part of a split-phase operation, thus the result of this call will be found in the `readDone(error_t error, uint16_t data)` void event. The `data` parameter represents the noise floor value. However, the Rssi values given by TinyOS are not in dBm units hence, in order to get a significant value, it has been converted using the following equation:

```
int8_t noise_val= (int8_t)(((data-0x7F)&0xFF) - 45);
```

Executing the line code above, the BS turns a 16 bits unsigned integer in a 8 bits signed one, and then 45 is subtracted from the result to obtain the correct Rssi value in dBm. This operation is repeated every time the `ListeningTimer` fires, until the readings counter reaches 40. Each time a value is read, the mean of Rssi is updated using the standard formula:

$$new\_mean = \frac{old\_mean \times (N-1) + val}{N};$$

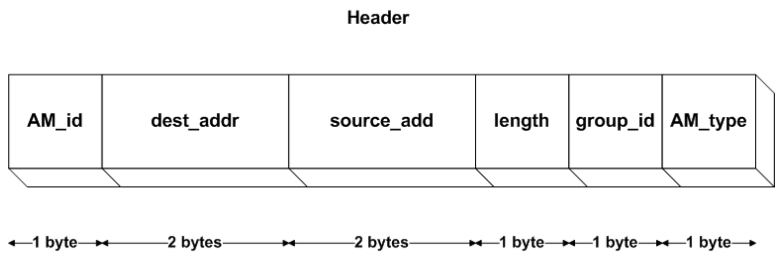
where $val$ is the last value acquired, $N$ the number of readings, $old\_mean$ and $new\_mean$ the mean value before and after the last reading. Every time a better Rssi value is found, it is chosen as a threshold, and the next values will be compared with it. Eventually, when this operation has been done for all the 16 channels of the CC2420 radio chip, the best channel is selected, and the BS is ready to go through step 2: Beacon Sending.

## 5.2 Beacon Sending

Right after the transmitting channel is chosen, the Base Station starts sending broadcast beacon frames. The beacon frame is the typical TinyOS packet, with 8 bytes of header followed by the payload (Figure 5.2). The header is constituted by:

- one byte to indicate whether the packet is an AM Packet or not;
- two bytes for the Destination Address (FF FF if broadcast);
- two bytes for the Source Address;

- one byte to indicate the Message Length;
- one byte for the Group ID;
- one byte for the for the Active Message Handler Type.



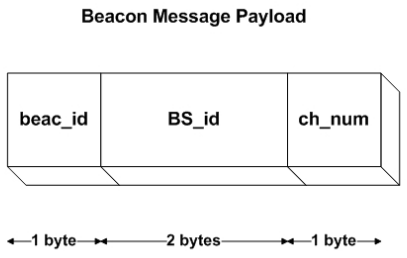


Figure 5.2: The Header of the typical TinyOS packet (above) and the Beacon Message payload (below)

The Beacon Message Payload, instead, is a 4 bytes payload that includes the Base Station ID and the channel number. The first byte is an unsigned 8 bits used as a frame indicator: each type of frame used in this work, indeed, contains as first byte of the payload an exclusive value that must set every frame apart from the others. The beacon packets are transmitted every 50 ms waiting for the mote to intercept them.

Indeed, when the mote switches on, it starts scanning every channel periodically in order to find the beacon frame of its own Base Station, without transmitting any packet. The scan is performed as explained above for the BS: the `channelScan` timer fires every 100 ms and, when it does, it sets a new radio channel always using the `sync()` command and the `syncDone(error_t error)` event of the `BSResearch` (CC2420Config) interface. Whenever the mote receives a packet, it checks for its payload and especially for the beacon frame indicator and the BS ID. Indeed, in order to intercept only the beacon messages of

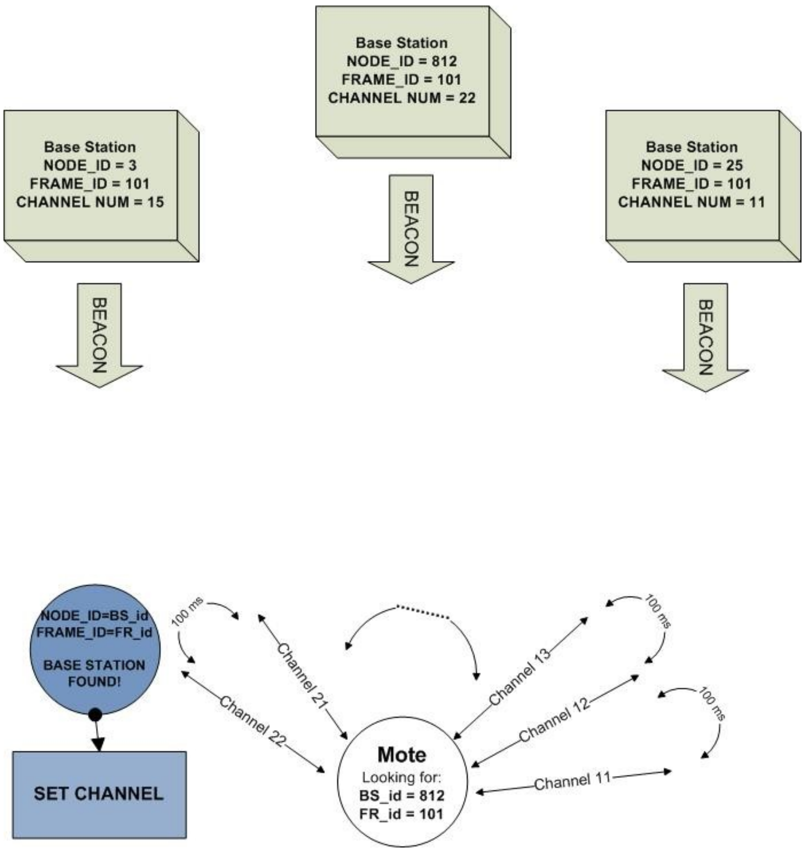


Figure 5.3: Example of Beacon Sending: the mote is listening to the radio channel waiting for intercepting the beacon message that contains the known BS id. If it receives the beacon of other BS (ch. 11 and 15), it keeps scanning the channels, until it finds its BS (ch. 22)

the BS which it is supposed to connect, the mote, after it switches on, already knows the indicator value and Base Station ID which it is looking for. Hence, if these values are the ones expected, it stops scanning, and starts communicating with the BS in the channel where the beacon has been found, in order to set an ubiquitous connection as it will show in the next paragraph. Once the BS starts sending beacon frames, if it does not receive any packet from the mote for the following 2 minutes, it starts again from the Channel Scan step. This happens because the BS could not receive anything from the mote either because it does not switch on yet or because it has some trouble in replying to the beacon: hence, the BS starts again the channel scanning, indeed the noise floor situation might be changed during the time spent waiting for the mote reply.

## 5.3 Association

When the mote intercepts the beacon packet of its own BS in one of the channel, and selects the latter as the transmitting channel, it sends an association packet to the BS in order to prove it is the mote which the Base Station is associated. Along with the Beacon procedure already discussed, this might be seen almost like a **three-way handshake**.

This method is widely used in TCP ("Transmission Control Protocol") to establish a connection between a server and a client. The name of this technique refers to the three messages exchanged between the two entities, that usually are:

1. **SYN:** is the message the Client sends to the Server, that contains a random value A in the sequence number field;
2. **SYN-ACK:** the Server replies with a SYN-ACK message, where the acknowledged random value is set to A+1 and another random value Y is added as sequence number;
3. **ACK:** eventually, the Client confirms the connection sending to the Server the acknowledged value Y+1 and setting the sequence number to A+1.

In this scenario, the Beacon Message could be seen as the first of this three steps. The second one is represented by the association packet the mote sends to the BS. The payload of this packet includes 7 bytes, divided as shown in Figure 5.4:

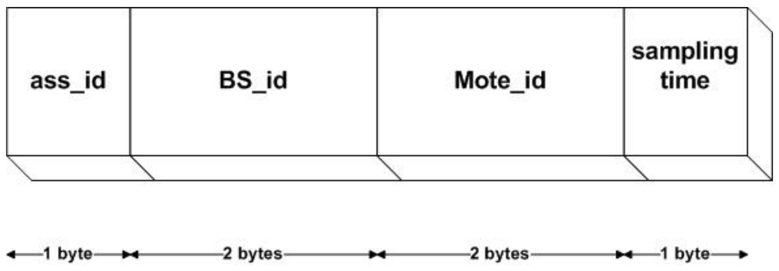


Figure 5.4: Association Packet Payload

where *ass_ id* is the identification value of the association frames, *BS_ id* and *mote_ id* are respectively the Base Station and the mote node id and *sampling time* indicates the rate the accelerometer will sample the data. Right before this message is forwarded, an ACK is requested using the `Ack` (PacketAknowledgments) interface. When the packet is received by the BS, this ACK is sent to the mote, and it corresponds to the final step of the association protocol. If the mote does not receive the ACK, it tries to resend the packet again. If the ACK does not arrive after 10 attempts, the association fails and the protocol has to start again from the beginning (hence the mote returns scanning the channel waiting for listening to the beacon frames). However, if it does, the association is finished and the protocol can go through its fourth and last step: the Power Level Selection.

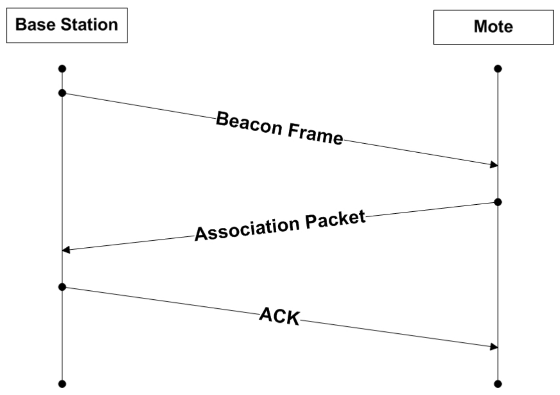


Figure 5.5: The Revised “Three-way Handshake” technique used for the motes association

## 5.4 Power Level Selection

Once the best channel is chosen and the two devices are connected, the last thing left to set is the Power Level. The aim of this operation is to obtain a low power but high efficient communication between the mote and the Base Station. The 8 power levels and their relative dBm and current draw values for the CC2420 radio chip have already been illustrated in Table 3.1.

The protocol is supposed to choose the lowest possible level that satisfy the QoS parameters requested. These parameters are usually well-defined by two values: the **PER** (Packet Error Rate) and the **Rssi** value of the received packets. The PER is the number of incorrectly received packets divided by the total number of received packets. Nevertheless, in this work PER means the ratio between the number of packets received by the Base Station and the number of packets sent by the mote:

$$PER = \frac{Packets\_received_{BS}}{Packets\_sent_{mote}}$$

The Rssi has already been explained in 5.1, but the command to call in order to get this value is different from the noise floor one discussed above. In this case, in fact, the `CC2420Packet` interface is used and the command syntax is the following:

```
async command int8_t getRssi( message_t* p_msg );
```

where *p_msg* is the received message.

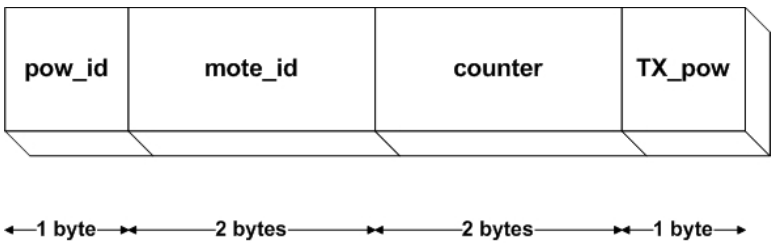


Figure 5.6: Power Test Packet Payload

However, as for the noise floor readings, this value needs also to be subtracted by 45 to obtain a meaningful Rssi value.

The Power Level can be set according to both these values or to just one of them. This protocol works checking only the first one: if the PER value calculated is below a chosen threshold, the power level is selected.

After that the ACK is received, the mote starts a periodic timer that fires every 50 ms. This interval has been chosen because it is the same one that will be used when the data packets will be sent, so it can return a reasonable estimate of the PER. Every time the `fired()` event connected to this timer runs, the Mote sends to the BS a packet which payload includes (Figure 5.6):

- one byte for the usual frame indicator value;
- two bytes for the mote ID;
- two bytes for the packet counter;
- one bytes for the power level which the packet is sent.

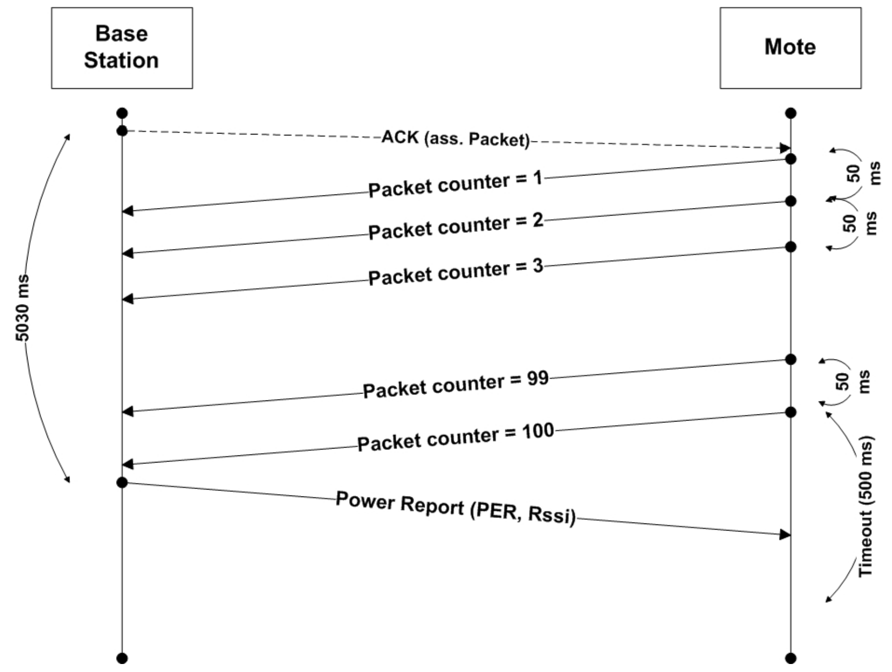


Figure 5.7: Example of Power Packets Transmission

The mote sends 100 packets to the Base Station: all packets are the same except for the *packet counter* bytes. This value increase every time a packet is sent, and it is useful for the Base Station to identify if there is any packet missing.

Right After the 100th packet is sent, the mote starts a Time-out, that allows it to wait for the Base Station's reply.

Meanwhile, the Base Station has started another periodic timer (using the `PowerSettingTimer` interface) that lasts 5030 ms. This timer has a purely computing meaning. It may happen, in fact, that the BS would not received the last packet (or any packet at all) from the mote because, for example, the power level is too low. Otherwise, every time the BS receives a power packet, checks its packet counter and gets its Rssi value[3]. When the timer fires, the Base Station sends to the mote its power report: this is a packet which payload contains, apart from an other frame indicator value, the power level and the node ID, one byte for the PER and one for the Rssi calculated (Figure 5.7).

On the other side, when the mote received the report, stops the Time-out and analyses the packet received. If the PER is lower than 10, the current power level is selected, otherwise the mote sets the power level to the next one available and repeats the same operations just described. If no one of the 8 levels satisfies the QoS requirements, it sends to the BS a packet telling that the power level is inadequate, and the whole protocol starts once again from step 1. On the contrary, when a power level is selected, the Network Protocol is finished, and the WSN can start transmitting data packets.

[3]Even if the Rssi value is not employed in order to select the Power Level, it was used in several tests which will be illustrated in Chapter 7

# Chapter 6

# Application

The second section of this work is dedicated to acquisition, processing and analysis of the data sampled by the integrated Freescale MMA7260Q 3-axis accelerometer (3.4.2) of the Shimmer mote. As shown in Figure 6.1, the Shimmer mote has to be placed on the barbell, while the Base Station can be connected to a PC in order to collect and save the data acquired.

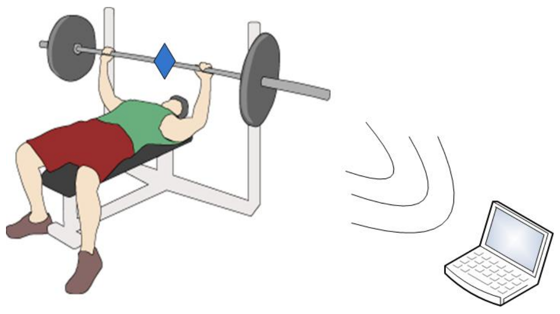


Figure 6.1: Example of application proposed for this WSN: the blue diamond shows where the Shimmer mote should be placed

This paragraph will describe both the TinyOS block code, which is in charge of acquiring the samples from the accelerometer and the Matlab [23] one, that computes these measurements and shows, through the aid of a GUI (Graphical User Interface), the final outputs of this WSN: the training profile of the user (Hill's and Power-Velocity curves) and the Power Analysis of some standard bench press practices. Before starting this dissertation, though, a brief introduction to the Hill Curve and the Power-Velocity curve will be presented, in order to make the outputs analysis clearer.

## 6.1 Training Overview

As stated before, the first output of the proposed work is to provide a training profile for the user. This profile is based on the **Hill's Equation**, a widely used formula based on the correlation between the Force and the velocity. Given the Hill equation, then, it can be obtained an other important curve to completely define the subject's training profile, that is the **Power-Velocity** curve. In this section will be explained the meaning, the fitting and the analysis of these curves.

### 6.1.1 Hill's hyperbolic equation

The Hill's curve is given by the equation:

$$C = (N + a) \times (V + b)$$

in which $N$ is force, $V$ is velocity, and $a$, $b$ and $C$ are constants with the dimensions of force, velocity and power respectively.

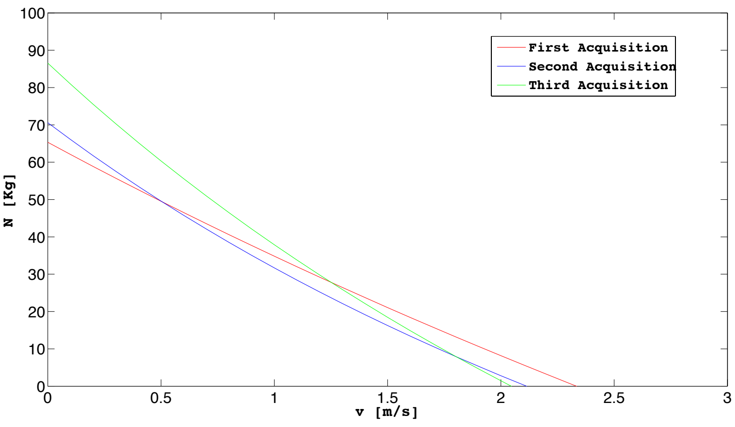


Figure 6.2: Training improvement through Hill's Curve: Strength

The Hill's equation was introduced by A.V. Hill to describe the relationship between the oxygen tension and the saturation of haemoglobin [24]. Followed by more detailed studies, it began greatly useful for describing how the muscle action is related to the velocity of shortening of movements. Indeed, the Hill's equation implies that velocity of muscle's contraction is inversely proportioned to the load.

According to this hyperbolic equation, athletes are able to monitor their training level and decide which feature they prefer to improve. The Hill's curve shows that the development of great strengths will not necessarily enhance the speed. Figure 6.2 and 6.3 show two examples of Hill's curve and how this one may change if the athlete would rather improve either his strength or his speed.

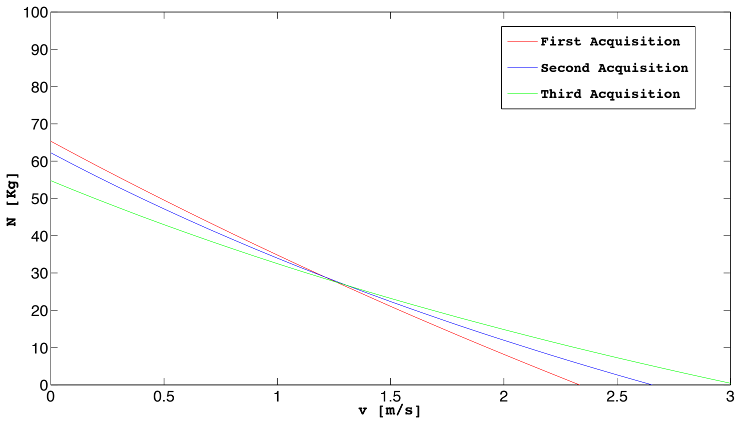


Figure 6.3: Training improvement through Hill's Curve: Velocity

### 6.1.2 Power-Velocity curve

The second important relationship used in this work is given by the Power-Velocity curve. It is directly extracted from the Hill's Curve by multiplying its $x$ and $y$ coordinates in order to obtain values with the dimensions of power. An example of this curve is illustrated in Figure 6.4.

This curve adds some useful information about the user's training profile, such as the maximum power the muscles are able to provide. Given that, indeed, it is possible to create a "power profile", and the user may choose a specific power range within he would rather practise.

## 6.2 Data Acquisition

Right After the Network Protocol described in Chapter 5 finished all its operations, the mote is almost ready to sense the data. Indeed, the last operation left to do is setting the Shimmer interfaces in charge of initializing the accelerometer.

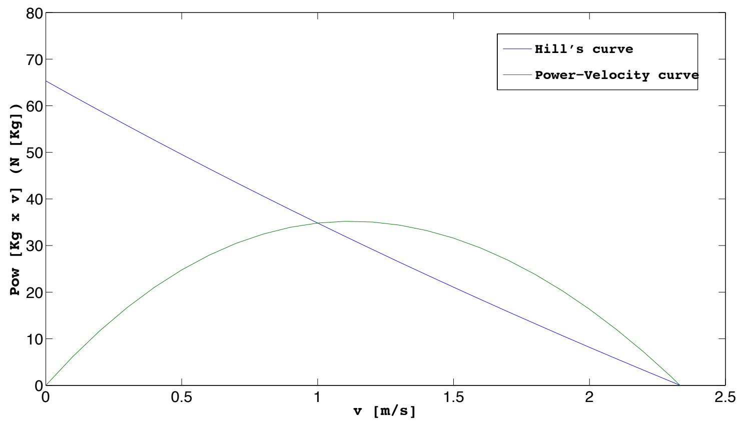


Figure 6.4: Power-Velocity Curve

The most part of commands are provided by the `shimmerAnalogSetup` interface that, for instance, sets the accelerometer sensitivity to 4g. It also includes the `triggerConversion()` command which starts the data acquisition. This command is in fact called every 10 ms (the sampling interval), using the known command of the `Timer<TMilli>` interface.

Each time this command is called, it activates the data transfer through the `Msp430DmaChannel` interface. However, the packets forwarding is not immediate, like it happens in the different steps of the Network Protocol, but it is handled by tasks. This allows the mote to collect five different readings before sending the packet to the Base Station. Figure 6.5 shows the simple structure of the Data Packet Payload:

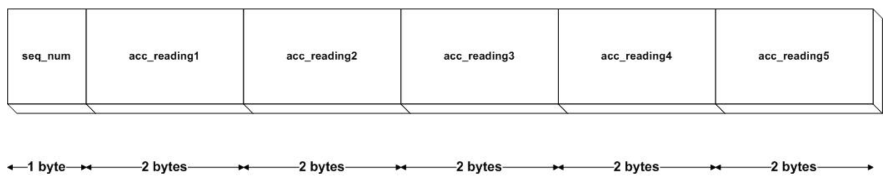


Figure 6.5: Data Packet Payload

where, apart from the five 2-bytes spots reserved to the accelerometer readings, there is one byte at the beginning of the payload containing the sequence number of the packet. Indeed, even if the mote is able to sample and send readings of all the accelerometer and gyroscope axis, in this work only one axis is used, so the data packet will only include the samples of the $x$ axis.

When the mote is ready to copy the data into the payload, the `copyData()` task is dispatched. This task simply saves the accelerometer readings inside the proper packet field and increments a counter in order to check how many samples have already been copied. Only after the fifth reading is completed, the `sendData()` task is computed. This task is in charge of sending the Data Packet with the usual `send - sendDone` split-phase operation performed by the `AMSend` interface. These operations keep running inside the mote until it switches off.

## 6.3 Data Processing

In this work, the software used for processing the accelerometer samples is **Matlab**. Matlab is a programming environment for algorithm development, data analysis, visualization and numerical computation [23]. A simple java application helps to move the data from the mote to this environment. Indeed, this application saves all the readings into *txt* files, which can be read by Matlab and saved into a vector or a matrix using the `textread` command. The Data processing consists of three main sections:

1. **Accelerometer Calibration:** the data acquired from the accelerometer are actually raw data and they are not in *g-force* units. The accelerometer must be calibrated executing some simple operations in order to convert the samples;

2. **Training Profile:** this section will illustrate how the Hill's Curve and the Power-Velocity curve can be drawn to define the user's training profile;

3. **Power Measurements:** the final step will describe how the power of each repetition can be calculated and how this information can be useful for the proposed application.

Eventually, the GUI used to show all the operations performed by the WSN will be presented.

### 6.3.1 Accelerometer Calibration

The accelerometers are devices that need to be calibrate as accurate as possible, because even very small biases may result in very significant drift changes. Usually, they should be calibrated every once in a while, but it is not mandatory if the environment in which they are used is always the same. Otherwise, they have to be calibrated when there are significant temperature changes, because the biases and the gain factors are very sensitive to these factors [25].

In this work, the calibration is executed externally, before the application starts running. The mote is placed in three different positions, at -1g, 0g and 1g. For each of this position, the accelerometer performs several readings that are saved into three different files using the java application mentioned above. A mean of these values is calculated and eventually, the calibration is achieved with the equation below:

$$x_{cal} = \frac{x_{sample} - x_{0g}}{(x_{1g} - x_{-1g}) \times 0.5}$$

where $x_{sample}$ is the raw accelerometer reading, $x_{0g}$, $x_{1g}$ and $x_{-1g}$ are the values read by the accelerometer when it is placed at 0g, 1g and -1g respectively. The $x_{cal}$ values obtained are in **g-force** units, where the g-force is the acceleration of gravity.

When the accelerometer is calibrated and the data are finally converted, it is possible to start the real processing in order to obtain the first output of the proposed WSN: the **Training Profile**.

### 6.3.2 Training Profile

As stated before, the first curve that must be considered in order to give meaningful information about the training level of the user is the Hill's curve. The algorithm used to draw it is the one performed by B. Wohlfart and K.A.P. Edman of the University of Lund, Sweden [26]. This method allows to fit the Hill's hyperbolic equation starting from experimental data using statistical techniques and regression analysis based on variables with the dimensions of force, velocity and power.

The algorithm does not force constraints on the maximum number of data collected; indeed the greater is this number, the better the algorithm works. Besides, the robustness of the algorithm increases if the $V_{max}$ and $F_{max}$ data are included in the data set, where these values represent respectively the maximum velocity that can be achieved with $F = 0$ and the maximum force with $V = 0$.

However, three is the minimum number of "$V$ and $F$ couples" needed in order to obtain a significant fitting, and it is also the number of acquisitions chosen for the proposed work (in line with most of the applications on the market).

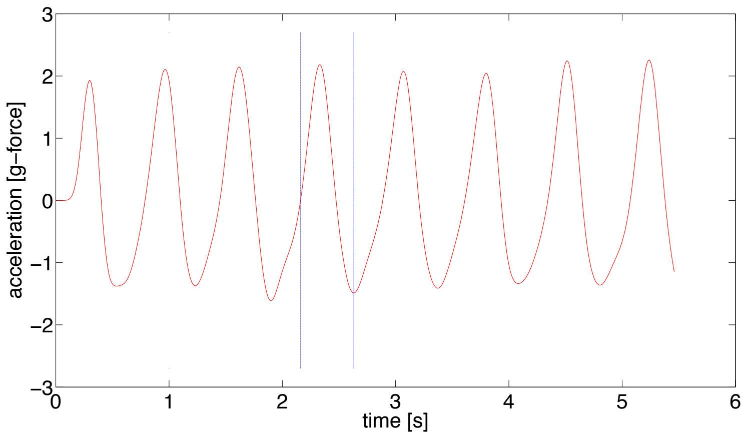


Figure 6.6: Window interval when a low-load practice is performed

The dataset acquisition works as follows. The user is asked to execute three different sets of the bench press exercise with respectively a low, a medium and a high load. For each set, 4 or 5 repetitions should be completed, trying to perform them as fast as possible. A Matlab function is created in order to convert the raw data of the Shimmer mote placed on the barbell from acceleration to velocity units. The goal of this function is to divide the accelerometer signal into as much windows as the number of the repetitions. Once the signal is fragmented, every window is integrated to obtain a velocity signal. The function then returns the mean of the peaks of each window.

Several attempts had been executed to decide where exactly each window should have started and stopped. Eventually, two different intervals have been adopted to obtain the best measurements (even if, as it will be widely explained in Chapter 7, these values are not really accurate) , according to the load employed in the practice. When low or medium loads are used, the interval is the one that starts each time the accelerometer signal has a transition from negative to positive and stops when the signal reaches a minimum (Figure 6.6). Otherwise, with high loads, it starts when the transition is from a positive to a negative value, and stops, as in the previous case, when the signal reaches a local minimum (Figure 6.7). However, these transition are kept into account only if the variance between adjacent samples is greater than a given threshold (more details will be provided during the performance evaluations in Chapter 7).

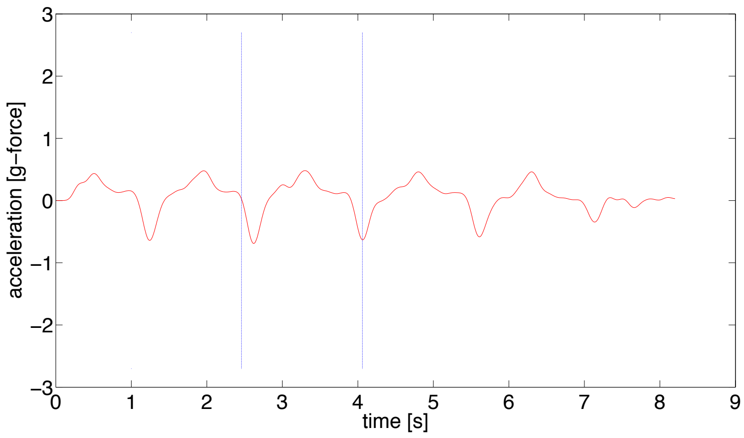


Figure 6.7: Window interval when a high-load practice is performed

Given the $V$ and $F$ values , the Wohlfart's and Edman's algorithm can be computed in five steps:

1. Three variables are defined: $X = V$, $Y = F$ and $Z = V \times F$;

2. Nine different sums are calculated (e.g. $S(X) = \sum_{k=1}^{3} X(k)$, where 3 is the number of acquisitions) for $X, Y, Z, XZ, YZ, XY, X^2, Y^2, Z^2$;

3. The values otained above are used for calculating the products of deviations from the mean (e.g. $[XY] = S(XY) - \left(\frac{1}{k}\right) \times S(X) \times S(Y)$) for the following products: $XY, XZ, YZ, X^2, Y^2, Z^2$;

4. The parameters $B_x, B_y, A$ are achieved with the following equations:

$$B_x = \frac{[XZ][Y^2] - [YZ][XY]}{[X^2][Y^2] - [XY]^2};$$

$$B_y = \frac{[YZ] - B_x[XY]}{[Y^2]};$$

$$A = \left(\frac{1}{k}\right) \times [S(Z) - B_x \times S(X) - B_y \times S(Y)]\,;$$

5. Given the Hill's equation in the form presented in 6.1.1, the parameters of the hyperbolic equation are set:

$$a = -B_x;$$

$$b = -B_y;$$

$$C = A + B_x \times B_y;$$

After that, the Hill's curve is finally determined. And, by simply multiplying the $x$ and $y$ vectors of the latter and plotting the correlation between the velocity and the values calculated, the Power-Velocity curve can be drawn, as already shown in Figure 6.4.

### 6.3.3 Power Measurements

Once the application created the user's training profile, it is possible to go through its second main output: the **Power Evaluation** of the bench press practice. A function very similar to the one used in the previous paragraph to determine the velocity peaks is employed. However this function, instead of returning the mean of all the peaks spotted, it returns a vector containing the velocity maximum for every repetition of the practice set. This is why the application then can be allowed to return the power measurements for every one of these repetitions, obtained by multiplying each peak by the load used in the set (in Kg units).

As it will be described in 6.3.4, the user decides the Power range within he wants to train by selecting a specific type of training (e.g. Explosive Strength) or just setting this range on his own. The power values acquired are compared to the ones of the Power-Velocity curve by converting them to percentage values, and the application returns the following output (see Figure 6.8):

- a **green** bar if the repetition is executed at the right velocity;
- a **yellow** bar if the execution is good, but not perfect;
- a **red** bar if the execution is poor.

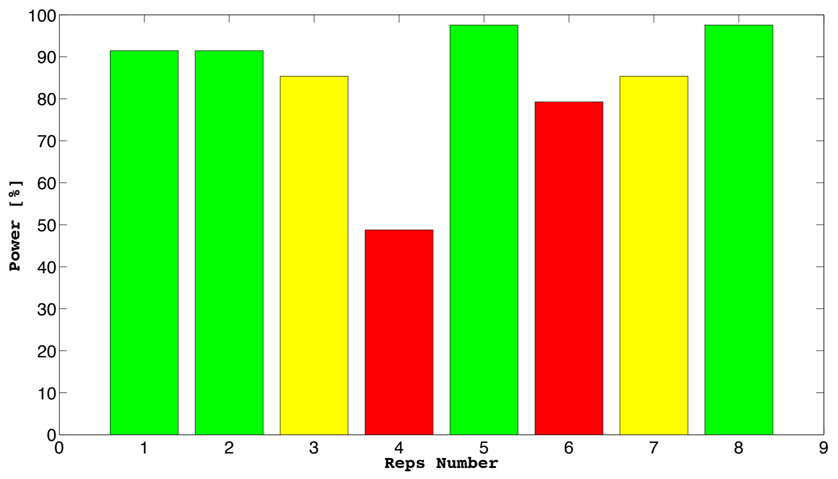


Figure 6.8: Example of Power Evaluation output

### 6.3.4 Graphical User Interface

In order to display the outputs obtained in a better and clearer way, a **GUI** (Graphical User Interface) is built. It contains button panels and edit text spaces that allow to run the Matlab functions just discussed in the previous section. The GUI consists of:

(A) Two axis employed to display the Hill's curve, the Power-Velocity curve (the first) and the Power evaluation bars (the latter);

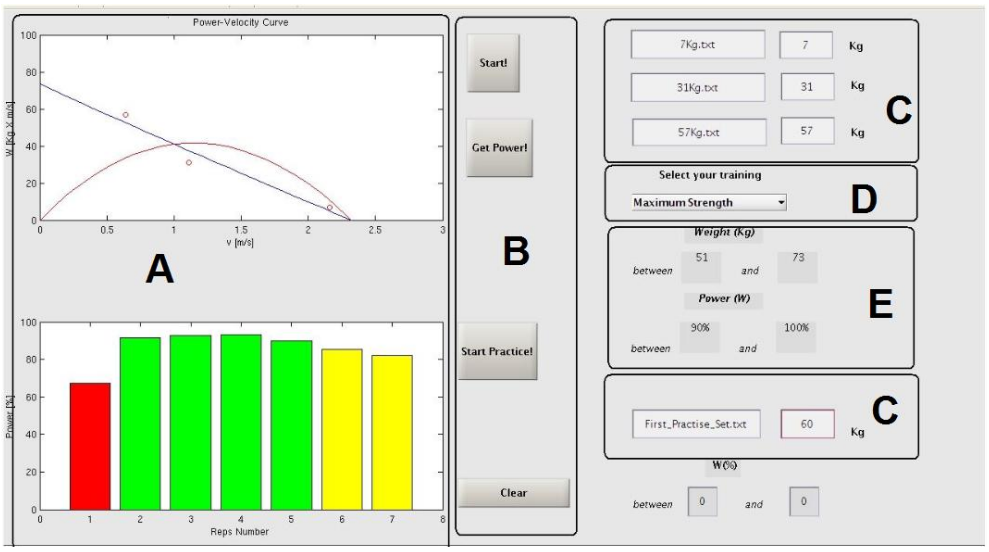
(B) Three Push Buttons for plotting the curve in these axis plus an other one in charge of clearing them;

(C) Four Edit Text spaces to upload the Text (.txt) Files containing the three acquisitions for the Hill's Curve and the one for the practice set (and the relative spots for the load values);

(D) One Pop-Up Menu icon which is used to select the desired type of training;

(E) Two Static Text spaces that show the Load and the Power ranges proposed for the selected training.

The execution order of the operations is the same discussed earlier. At first, the user inserts the three different acquisition and their relative loads and, by pushing the **Start** button, he will obtain his personal Hill's curve in the graph on the top. The Power-Velocity curve, instead, can be easily obtained with the **Get Power** button. After that, he can choose the type of training between 5 different options: **Maximum Strength**, **Explosive Strength**, **Hypertrophy**, **Quick Strength Endurance**, **Resistance Endurance**. According to the training selected, the GUI will show the Load and the Power range suggested in order to perform as good as possible. If none of these options are selected, the user can set his own personal Power Range in the corresponding spaces next to the graph on the bottom. This graph is in charge of showing the Power Evaluation of the practice set by adding the file with the acquisition in the suitable space and pushing the **Start Practice** button. The user, then, can obtain power estimation of new sets only by changing the uploaded file with the one wanted.



7Kg.txt 7 Kg
31Kg.txt 31 Kg
57Kg.txt 57 Kg

Select your training: Maximum Strength

Weight (Kg) between 51 and 73
Power (W) between 90% and 100%

First_Practise_Set.txt 60 Kg

W(%) between 0 and 0

Figure 6.9: An Example of the GUI employed

# Chapter 7

# Performance

The network protocol and the application have been tested to check both their behaviour in some realistic environments (such as a crowded gym) and their accuracy in the data processing. The first one, indeed, needs to ensure the lowest PER possible even if a lot of devices are used at the same time and in the same place. The performance of the latter, on the contrary, is not affected by the noise or the packet traffic, but several measurements have been tested to obtain velocity values as precise as possible.

The aim of this Chapter is to describe the strengths and the weaknesses of the proposed work, focusing first on the Network Protocol and then on the Application performance.

## 7.1 Network Protocol

Setting a suitable power level is fundamental to achieve efficient data transmission. Indeed, the Transmission Power behaviour is highly linked to features like, for example, the distance between the devices. In order to make it clearer, some experimental data have been acquired. Two motes, one acting as a Base Station and one as a transmitting mote, like in the WSN proposed in this work, have been used. For all the duration of the experiment, the Base Station remained still in its position while the mote have been gradually moved away from it. The motes have been placed in LOS (Line of Sight), 0.5 meter above the ground (Figure 7.1). For every distance examined, the mote sent 100 packets to the BS, and this operation have been repeated three times.

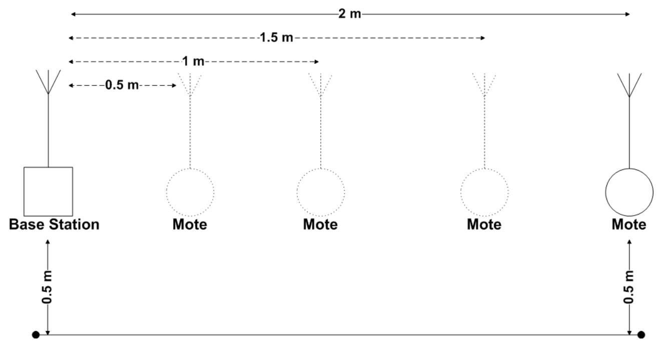


Figure 7.1: LOS Transmission between the Base Station and the mote

The graph below shows the relationship between Distance and PER (Figure 7.2) when a −25 dBm transmission power is used.

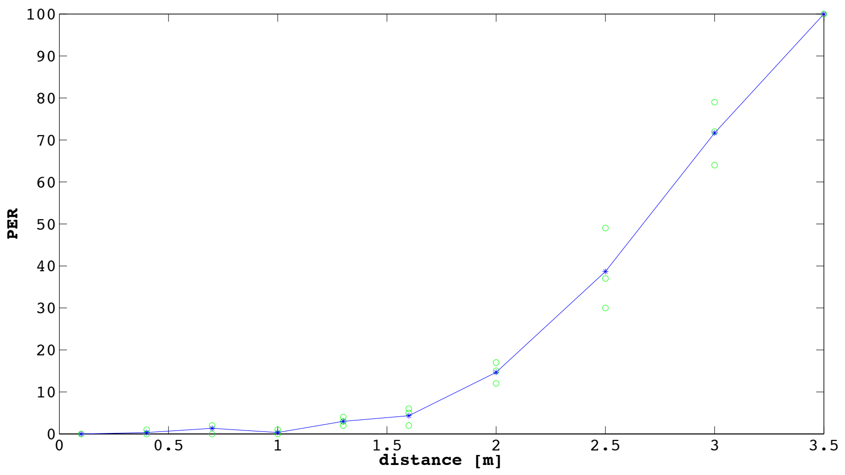


Figure 7.2: Distance/PER relationship with Tx Power = -25 dBm

The green circles represent the measured values while the blue curve describes the course of the means of each set of samples. As it might be seen, the PER becomes already consistent for distances over 2 meters, and the transmission fails (PER = 100) when the mote is 3.5 meters far from the Base Station. The Rssi of the received packet is also correlated with the distance. Figure 7.3 illustrates how this correlation works: the further is the mote from the Base Station, the worse is the Rssi value (that is, the more negative).

The Rssi value has been evaluated only on the received packets: this means

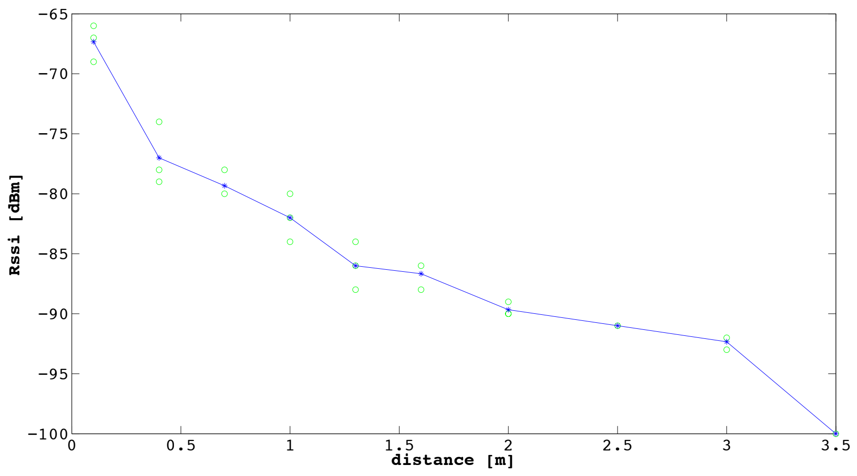


Figure 7.3: Distance/Rssi relationship with Tx Power = -25 dBm

that, if the Base Station does not receive, for instance, 5 packets, the Rssi mean is only based on 95 values out of 100. Hence, it is fair to assume that if the PER is not consistent, the estimation is quite near to the real Rssi average (that is the one that takes into account also the missed packets). Otherwise, the bigger is the PER, the less meaningful is the Rssi value (it gets underestimated). If no packet is received (at 3.5 meters in Figure 7.3), then the Rssi value is set to $-100$ dBm.

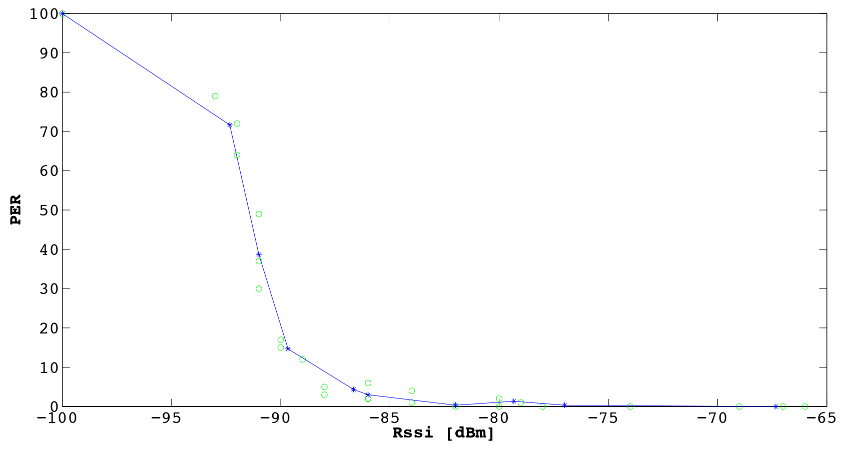


Figure 7.4: Rssi/Per relationship with Tx Power = -25 dBm

Another useful information can be extracted from these measurements if the relationship between the PER and the Rssi is analysed. Although the PER values do not depends on the Rssi ones, from the graph in Figure 7.4 an useful rule in defining the Power Transmission parameters can be detected:

- if Rssi < -95 dBm then PER = 100% ;
- if -95 dBm < Rssi < -85 dBm then 10% < PER < 100% ;
- if Rssi > -85 dBm then PER < 10%.

In order to show how the quality of the motes' connection increases along with the Power Transmission chosen, the same measurements have been collected for 3 more different power levels: $-15$ dBm, $-7$ dBm and 0 dBm. The settings of the experiment are the same of the one illustrated before in Figure 7.1. The following graphs (Figures 7.5 and 7.6) show how the Distance/PER and the Distance/Rssi relationship change according to the power level employed.

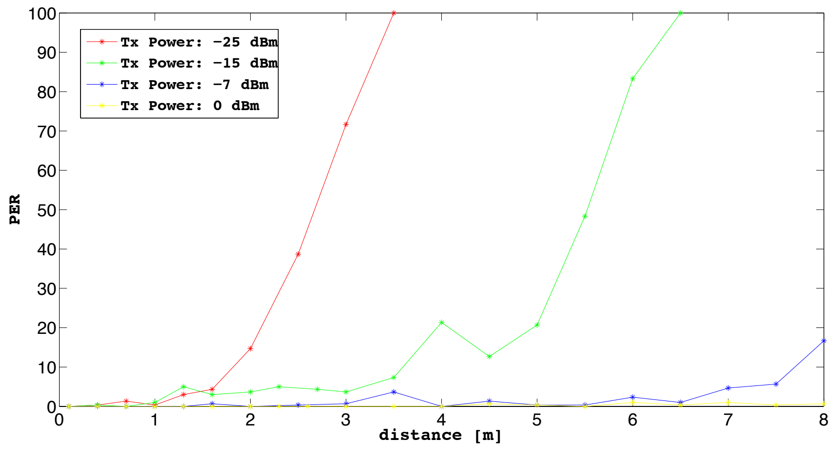


Figure 7.5: Distance/Per relationship with different Tx Power Levels

The experimental data show what it has been expected. For a given distance, the higher is the Tx Power, the lower is the PER and the less negative is the Rssi. This explains why the Network Protocol presented in Chapter 5 is mainly focused on the selection of a suitable Tx Power: even for little distances, the difference between the PER at a power level $i$ and the one at a power level $i+1$ can be significant. This can be noted, for example, in Figure 7.5, where, at 3

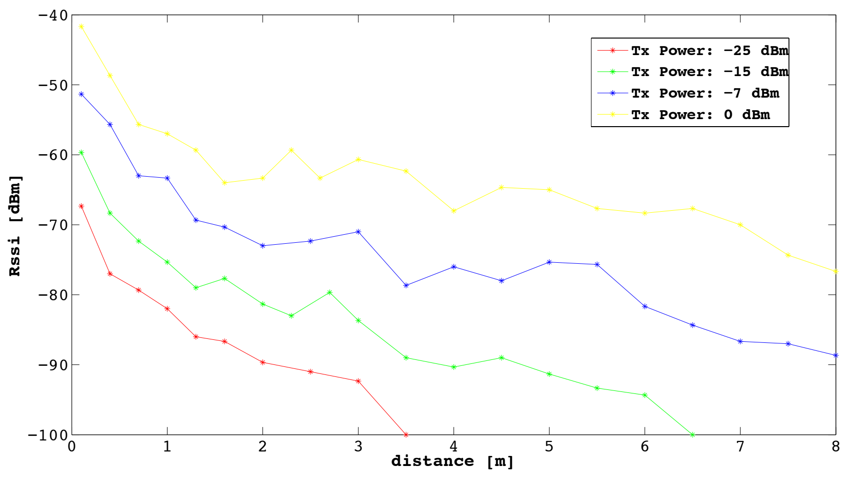


Figure 7.6: Distance/Rssi relationship with different Tx Power Levels

meters distance, the PER of a −15 dBm transmission is almost seven times less than the one of a −25 dBm transmission.

Anyway, the performance of the Network Protocol proposed in Chapter 5 is not just a matter of distance between nodes, but also a matter of traffic and noise in the channel. In order to show the robustness of this protocol, another experiment has been set. As for the previous one, the aim of this test was the evaluation of the PER and the Rssi over 100 packets sent from the mote to the Base Station, but the working hypothesis changed. At first, given the channel, the Tx power and the distance between the motes, the parameters just mentioned have been calculated gradually increasing the number of motes transmitting on the same channel, in order to simulate a "crowded gym" scenario. Every measure has been repeated four times. Secondly, the same measurements have been collected adopting the network protocol used in this work. Basically, the Tx power and the distance remained the same, while the channel has been selected with the new protocol. The results of this experiment are shown in Figure 7.7.

The improvement from the "One-Channel Transmission" to the "Propose Method" is highly remarkable, especially when $BS_{number} \geq 4$. The first one (red curve), indeed, grows as an exponential-like function, while the new protocol one (blue curve) is still around PER = 0%. The experiment has been stopped when nine different motes were simultaneously transmitting, because no packet arrived to

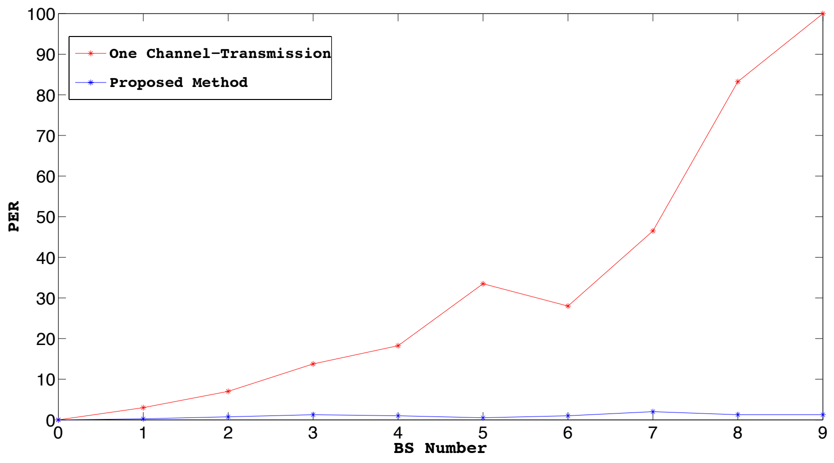


Figure 7.7: BS Number/Per Transmission Methods Comparison [Dis: 1m, Tx Power: -25dBm]

destination when all these motes where transmitting on the same channel.

However, a very interesting result can be obtained from this graph. The red curve shows that, according to the PER threshold proposed in Chapter 5 (around 10%) only 2 BS can be used with the "One-Channel Transmission" method. Otherwise, it is fair to suppose that this number would be quite larger if the new protocol is used. Indeed, Figure 7.7 shows that the blue curve is not influenced by the number of BS transmitting, because they are working in different frequencies. Since the permitted channels are 16, this allows to reckon that this behaviour must be the same until 16 different BS (one per channel) are transmitting.

## 7.2 Application

As already mentioned earlier (6.3.2), the Application presents some inaccuracies in the extraction of the velocity peaks. The main issue in evaluating these values is the course of the accelerometer signal. This is due to the structure of the latter (3.4.2): a positive acceleration is followed by a transition to a negative peak before coming back to zero and the other way round. Besides, when the acceleration variations are close to each others, this transitory may be hidden within them.

In order to simplify this idea, Figure 7.8a and 7.8b illustrate respectively the accelerometer signal for a “Down-Stop-Up” and a “Down-Up” movement of the arm. It can be easily noticed that the sum of the two signals in Figure 7.8a is not equal to the one in the Figure 7.8b, for the reasons explained above.

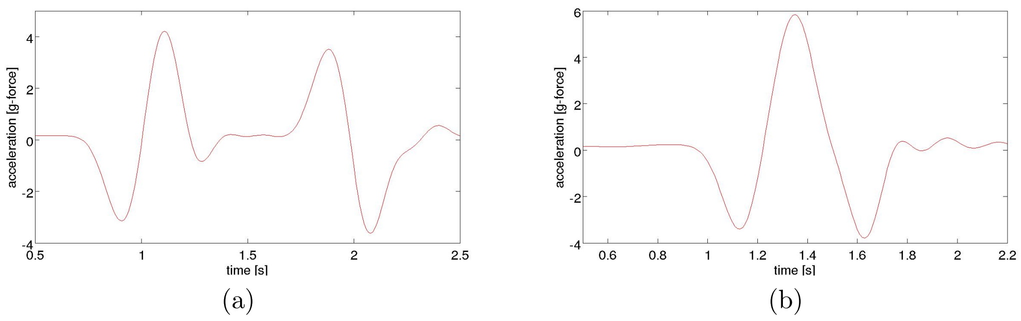


Figure 7.8: Example of “Down-Stop-Up” (a) and a “Down-Up” (b) movements of the arm.

Given that, the signal has been divided in different windows, one for each repetition (6.3.2), to obtain a better estimation of the velocity peaks. Several experiments have been performed in order to select the best window possible and, as described in Chapter 6, two intervals of integration have been chosen. The selected intervals are different between the low loads and the high loads because in the first one the transitory is highly influential, hence only the “Up” movement is detected. In the latter, on the contrary, the transitory of the previous repetition is almost non influential for the next one, so all the “Down-Up” movement can be detected.

Nevertheless, the velocity values were still not so accurate. The last problem left to solve, in fact, was that the peaks achieved were greater than the values expected. This has been noticed because it is known that the maximum velocity that the human arm can reach is around 2.6 - 2.8 m/s [27]. All the values concerning the low loads and calculated with the algorithm proposed, though, were 1.5 m/s higher than the ones expected. In order to avoid this overestimation, it has been decided to subtract this number to all the peaks, resulting in a good but not perfect estimation.

On the other hand, the algorithm in charge of drawing the Training Profile curves performs really well even if some data are miscalculated. As an example, Figure 7.9 shows the Hill’s curve given by this algorithm when the acquisition of the high load has been forced to be wrong. It can be seen, in fact, that despite

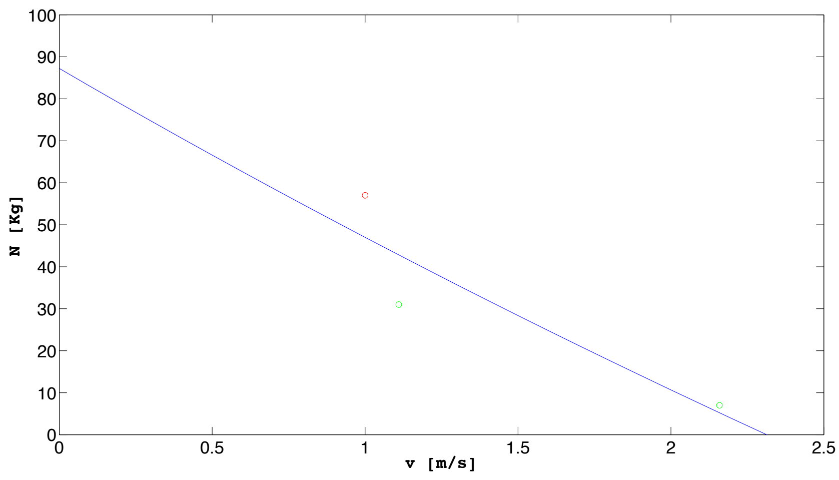


Figure 7.9: Hill's Curve fitted with incorrect data

this value (red dot) is almost the same of the middle load one (first green dot), the fitting of the curve still works properly.

# Chapter 8

# Conclusions

As widely explained in Chapter 2, a Wireless Sensor Network is a group of low-cost low-speed and low-power consumption devices, also known as **motes**, capable of sensing data from the environment and transmitting them through wireless links. Because of its features, this technology greatly expanded in the last decade finding easy and feasible applications in different categories such as Military, Home, Healthcare and Sport.

The use of such networks in Sport totally transformed the way athletes, both the professional and the amateur ones, train. Indeed, the opportunity to take advantage of a virtual trainer in order to improve some skills or learn some basic training movements (depending on the user's level) was successfully welcomed by the consumers. The elite athletes, in fact, have been finally able to train in their usual context but, in the same time, to use a technologically advanced device that allows them to obtain better performance and better result without being forced to have recourse to outdated and inadequate laboratories. Indeed in sports like, for instance, swimming, baseball or tennis (Chapter 4) the differences between a lab environment and an usual training pitch or court is really consistent, and the employment of these devices made a massive impact on athletes' performance.

On the other hand, even people who sporadically play sports or go to the gym can highly exploit various types of virtual trainers: learning basic skills of a sport, preventing injuries or simply replacing "real" trainers when they are not available (e.g. peak-times in a gym) are only some of the common applications that also a non-elite sportsman can make use of.

This dissertation focused on a WSN used as a virtual trainer in the Bench

Press practice. This WSN consists of two motes, a Base Station attached to a PC, where it can download and process the data collected, and a Shimmer mote placed right on the barbell in charge of acquiring the data. The work was divided in two main sections: at first, a Network Protocol was developed in order to optimize the transmission between the motes and the coexistence of many similar WSNs. This Protocol consists in 4 steps and works as follows. When the BS switches on, it starts scanning all the 16 different radio channels in order to detect the Noise Floor of each one. At the end of this process, the BS sets the channel with the best (more negative) Noise Floor value. Right after that, it begins sending broadcast beacon packets in the selected channel, waiting for the mote's reply. Meanwhile, when the mote switches on, it starts listening periodically to all the radio channels as well, but without sending any packet. When it intercepts the beacon message that contains some known values (such as the Base Station ID and the beacon frame indicator), it stops scanning the channels and sends ad association packet to the BS, which replies with an Acknowledgement (ACK). When the two devices are connected, a power selection procedure (last step of the protocol) starts in order to choose the lowest power level possible that satisfies the Quality Of Service (QoS) parameters of the network.

The second part of this Thesis concerns data acquisition, processing and analysis. Once the mote are connected and the power level is set, indeed, the Shimmer mote starts sending data packets to the BS containing accelerometer readings. The accelerometer samples every 10 ms, and the packets are sent to the Base Station every 50 ms (five readings for each packet). These samples are collected by the BS in txt files using an appropriate java application and, when all the data are collected, it can start processing and analysing them using the Matlab environment.

There are two different outputs for the proposed application:

1. The user is asked to perform, as fast as he can, three different acquisition sets of about 5 repetitions each one, with a low, a medium and a high load respectively. The velocity peaks are extracted from these sets and the Training Profile of the user is shown using the Hill's Curve and the Power-Velocity curve (6.1.1);

2. The user, then, can start training performing as mush practice sets as he wants. For each set, according to the Training Profile and the type of

training selected by the user, the application provides a power evaluation of every repetition of the set showing its percentage value compared to the maximum one obtained in the Power-Velocity curve. If this number is within the power-range selected, a green bar (excellent) is visualized. Otherwise, the color of the bar will be yellow (good) or red (poor).

Eventually, the performance of the two different sections have been estimated. The results of several tests show that the Network Protocol proposed, where a Base Station decides to transmit on the best channel possible, can be up to **16** times better than a scenario where all the BS sends and receives packets on the same channel. This is actually very helpful in crowded environments, such as the gyms. Adopting this Protocol, a large number of WSNs can work simultaneously without interfering with each others.

The Application algorithm presents, instead, some inaccuracies on the evaluation of the velocity peaks, due to the nature of the accelerometer signal, which presents a long transitory after that an acceleration variation is detected. This may affect the proper evaluation of quick movements such as the one of the barbell during the bench press exercise. However, this error is partially smoothed by the Hill's curve fitting algorithm that, as it was proved in 7.2, works properly even if some data are miscalculated.

## 8.1 Future Developments

The work of this Master Thesis may be seen as a "starting point" both for improving some of the limitations underlined by the performance analysis in Chapter 7 and for implementing new suitable features. The following paragraph will present some of these possible developments.

#### Velocity Peaks Accuracy

The main issue highlighted by the Application algorithm performance was the partially inaccuracy of the velocity peaks calculated. The reasons of this error have already been explained in 7.2, nevertheless it might be possible to introduce some improvements to the existent algorithm which may result in better velocity estimations. For instance, the application of the Kalman Filter or the Particle Filter may allow to reduce the miscalculation.

### Real-Time Outputs

The Matlab Graphical User Interface employed is also able to directly connect with the motes and acquiring the data without the aid of the java application used in this work. In the old version of TinyOS 1.x (3.1) this interaction was simpler and the `connect` function was used in order to let the Matlab environment connect to the Serial Port. Java classes were still used, but not for saving the data in txt files as mentioned in Chapter 6. Indeed, they were only in charge of generating Java representation of the TinyOS packets, which were eventually sent through the serial port to the Matlab function connected.

In TinyOs 2.x, although it includes a totally new software, this interaction is still allowed, so it would be possible to draw the Hill's curve dynamically and to obtain a real-time estimation of the power repetition by repetition during the set, and not just at the and of it.

### Accelerometer Calibration

As mentioned in 6.3.1, the Accelerometer is a device that needs to be calibrated from time to time, especially when biases and gain factors changes can be consistent, due for example to temperature alterations. In the WSN proposed, the calibration of this device has been performed externally before beginning the data acquisition, but a simple program can be added to the software in charge of requesting a calibration, for instance, every $x$ days. This may slightly increase the accuracy of the measurements and the outputs.

### New Exercises Application

The proposed work can also be configured for other types of trainings, such as the Leg Press, Leg Curl or Lat Machine exercises, just to name a few. The network protocol will be exactly the same, while the application (and the relative GUI) will change in order to fit the specific training requirements.

# List of Figures

# List of Tables

# Bibliography

[1] J. A. Stankovic (2008) “Wireless Sensor Networks” - *IEEE Wireless Communications*, 92-95

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci (2002), “Wireless sensor networks: a survey” - *Computer Networks*, 38:393-422

[3] G. Lu, B. Krishnamachari, C.S. Raghavendra (2004), “Performance Evaluation of the IEEE 802.15.4 MAC for Low-Rate Low-Power Wireless Networks” - *Performance, Computing, and Communications, 2004 IEEE International Conference* , 701-706

[4] E. Callaway, P. Gorday, L. Hester, J.A. Gutierrez, M. Naeve B. Heile (2002), “Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks”

[5] University of california at berkeley. TinyOS website, http://www.tinyos.net/.

[6] V. Handziski, J. Polastre, J.H. Hauer, C. Sharp, A. Wolisz, D. Culler, D. Gay (2004) “Hardware Abstraction Architecture”

[7] MEMSIC, “MICAz Datasheet”, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148%3Amicaz

[8] ChipCon Products from Texas Instruments, “2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver”

[9] Shimmer website, http://www.shimmer-research.com/

[10] Roving Networks, Bluetooth Module RN-46, http://www.rovingnetworks.com/modules.htm

[11] B.P.L. Lo, S.Thiemjarus, R.King and G.Z. Yang (2005) "Body Sensor Network: a wireless sensor platform for pervasive healthcare monitoring"

[12] C.Otto, A.Milenkovic, C.Sanders, E.Jovanov (2006) "System Architecture for a wireless body area sensor network for ubiquitous health monitoring" *University of Alabama in Huntsville*

[13] U.S. Census Bureau, U.S. Interim Projections by Age, Sex, Race, and Hispanic Origin, http://www.census.gov/ipc/www/usinterimproj/

[14] D.A. James, N. Davey, T. Rice (2004) "An Accelerometer Based Sensor Platform for Insitu Elite Athlete Performance Analysis" *IEEE Wireless Communications*

[15] M. Lapinski, E. Berkson, T. Gill, M. Reinold, J.A. Paradiso (2009) "A Distributed Wearable, Wireless Sensor System for Evaluating Professional Baseball Pitchers and Batters"

[16] H. Ghasemzadeh, V. Loseu, E. Guenterberg, R. Jafari (2009) "Sport Training Using Body Sensor Networks: A Statistical Approach to Measure Wrist Rotation for Golf Swing"

[17] A. Ahmadi, D.D. Rowlands, D.A. James (2006) "Investigating the translational and rotational motion of the swing using accelerometers for athlete skill assessment" *IEEE 2006, Exco, Daegu*

[18] Freescale MMA7260Q Accelerometer Datasheet, http://www.freescale.com/

[19] Sensorize web site, http://www.sensorize.it/

[20] TinyOS wiki, Modules and the TinyOS Execution Model,Split-Phase operations, http://docs.tinyos.net/tinywiki

[21] J, Yu, Q. Tian, T. Rui, T. Huang (2007) "Integrating Discriminant and Descriptive Information for Dimension Reduction and Classification" *IEEE Transaction on Circuits and System for Video Technology*

[22] Q. Li, J. Ye, C. Kambhamettu (2004) "Linear Projection Methods in Face Recognition under Unconstrained Illuminations : A comparative study" *IEEE Computer Society*

[23] Matlab website, http://www.mathworks.it/products/matlab/

[24] S. Goutellea, M. Maurinc, F. Rougierb, X. Barbautb,L. Bourguignona, M. Ducherb, P. Mairea (2008) “The Hill equation: a review of its capabilities in pharmacological modelling”

[25] S.H. Peter, W. Golnaraghi, F. Golnaraghi (2010) “A Triaxial Accelerometer Calibration Method Using a Mathematical Model” *IEEE Transactions on Instrumentantion and Measurement*

[26] B. Wohlfart, K.A.P. Edman (1993) “Rectangular Hyperbola fitted to muscle force-velocity data using three-dimensional regression analysis”

[27] G. Dintiman, B. Ward (1988) “Sport Speed”