Università degli Studi di Cagliari
Dipartimento di Matematica e Informatica

Corso di Dottorato in

Informatica

Movements assessment and analysis for IMU based wearable networks

Tesi di Dottorato di

Gary Steri

Supervisor:

Prof. Gianni Fenu

Anno Accademico 2009/2010

Contents

List of Figures

List of Tables

Listings

Introduction

During the last few years increasingly smaller and cheaper devices have pervaded our houses, workplaces, cars and, more in general, many of our daily activities. The term *ubiquitous* or *pervasive computing* is becoming more and more popular, gaining the attention of important conferences especially with sensor networks as main topic.

Sensor networks have considerably arisen thanks to the important progress made in micro-electromechanical systems, digital electronics and also in wireless communications. The combination of these elements has allowed the development of tiny and versatile devices able to observe and monitor, very closely, an environment, an activity or anything else to be measured. We have examples of these devices employed in many kinds of applications, e.g.:

- environmental parameters survey: for monitoring seismic activity, weather conditions (temperature, wind, etc.), optimal conditions of cultural heritage (light exposure or humidity for paintings);

- military purposes: intelligent and self-organizing weapons;
- car automation and security: for offering even more comfortable functionalities or detecting dangerous situations;
- sports and medical applications: for measuring human body's activity, parameters and movements.

The last application is also called *wearable computing* and the devices used to create the (wearable) network are a particular kind of sensor such as accelerometers, gyroscopes and magnetometers: combining these sensors with micro-controllers, memory modules, serial and radio transceiver we obtain an *Inertial Measurement Unit* (*IMU*).

After a general study of (wireless) sensor networks, the focus of my PhD initially moved to the first kind of application listed above, in particular for security related problems and communication protocols [12], and then moved to the last point. Wearable computing and applications has been the main topic of my research and we have had the possibility to perform many tests using IMUs composed of triaxial digital accelerometers and gyroscopes.

My efforts have focused on movement analysis and evaluation in order to assess the correct execution of an athletic skill for sport performance or therapeutic rehabilitation. The approach that we used differs from the most used approaches in the way we utilized the sensors output. Generally, in applications for diagnosing specific diseases [4] or for assessing

the progress made during medical treatment [18, 5], sensors output is processed as a waveform in which a pattern can be recognized or the initial and the final positions of the movement detected.

We have attempted to view the sensors output as the basis for tracking the movement in three dimensions in order to obtain the actual trajectory of the whole movement. That is similar to visual approaches used, for example, in [25] or in [16], but we have only used IMUs without any optical devices or external assistants like GPS.

Using this kind of approach the main problem is to calculate displacements from every single variation of data provided by sensors; since sensors are electromechanical devices, they are prone to noise and measurement problems such as inversion and error propagation. Thus, measuring and tracking the entire movement is not a trivial problem if we want to achieve a good degree of precision for the aforementioned uses. In addition, we should provide a proper representation of the movement, even graphical, which can be efficiently analyzed and compared with other ones.

In this thesis we will describe the problems studied during the PhD course, the experiments performed and the solutions proposed, even comparing them with other solutions proposed in literature. In the first chapter we will give a general introduction to sensor networks with particular attention to the structure of the network, nodes organization, communication schemes and security issues. The second chapter will introduce wearable networks and will expose the techniques used to represent the human body

and its movement so that it can be easily visualized, stored and evaluated.

In the third and forth chapter the aspects related to the movement measurement will be covered: starting from the set up of the IMUs, sensors integration and programming included, we will show the experimental activities done and the sensors data processing. The fifth chapter, the last one, will discuss the results obtained and the final considerations.

Many results and issues explained in this thesis have been published in [12], [13] and [14].

Chapter 1

Introduction to sensor networks

Sensor networks can obviously be both wired and wireless, but the most part of their applications require a wireless network. Therefore, in this chapter we will immediately introduce Wireless Sensor Networks (WSNs), giving basic definitions and an example of environmental application for which we have proposed a tree structured communication and key distribution scheme.

1.1 Definition and main applications

A Wireless Sensor Network is a set of sensor nodes arranged within or very close to the phenomenon to be observed [1]. It can comprise different kinds of sensors (e.g. seismic, magnetic, infrared, acoustic, radar) useful for monitoring many environmental conditions [11]. Sensors are able to "collaborate" and preliminarily process data, sending not just coarse data to collector nodes [1]. Thanks to this characteristic, sensor networks are suitable for many kinds of applications, such as medical, military and security.

For example, in the medical field, we can find sensors able to measure blood pressure, heart activity or blood oxygen saturation in real time. It means that, these monitorings can be continuously done with an high level of reliability and without the presence of the patient in a medical structure. Textile and organic versions of this kind of sensors are employed in spacesuits or in military equipments for monitoring the health conditions of astronauts and soldiers. Moreover, similar overalls have been proposed for emergency operators in catastrophic events [21, 6].

The creation of these networks can require some techniques originally developed for wireless ad hoc networks. However, the protocols and algorithms directly derived from ad hoc networks do not fulfill sensor network purposes. We can argue this point outlining the main differences between these two kinds of networks [22]:

- the number of nodes in a sensor network can be much greater than in an ad hoc network;
- sensor nodes are deployed with high density;
- sensor nodes are prone to frequent failures;
- sensor network topology can change frequently;
- sensor nodes mainly use broadcast communications, whereas ad hoc networks are based on point-to-point communications;
- sensor nodes have very restricted power, computing and storage capabilities.

The last point is one of the main constraints of sensor networks; sensor nodes are equipped with limited power supply units and, generally, it is not possible to recharge or change them.

1.2 Architectures

First and simplest implementations of WSNs had a flat architecture composed of a limited number of sensor nodes for monitoring a single parameter in the all area; in this kind of scenario, a single sink node collects the all data and send them to a processing center.

The increase of number and typology of sensors has required more sophisticated architectures and communication protocols. In general, we

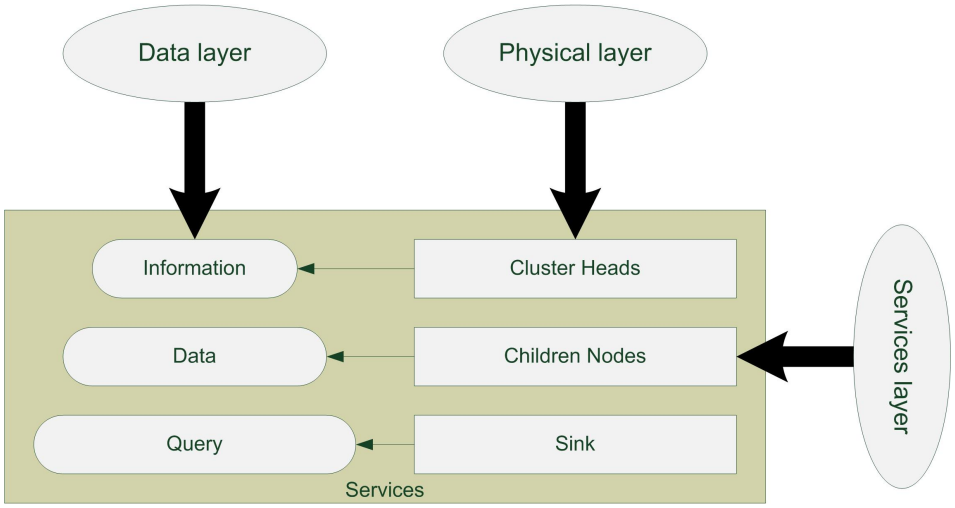

Figure 1.1: WSNs three tier architecture

can identify a three tier architecture divided into services, data and physical layer as shown in figure 1.1 [20]. Queries on the network are propagated from sink nodes to children nodes, which sense the data and send them to a cluster head. The latter transmits the result of the query to the sink node.

1.3 Tree structured model

We have proposed a model for a tree structured architecture ([12]) of a sensor network conceived to obtain weather conditions by monitoring temperature, wind speed and luminosity at the sea surface close to a beach or a coastal area. It uses a multi-hop communication scheme with sensors

identified by a unique ID and one sink node always active. Each node should be able to measure the three weather parameters and send them to the sink node, which should be able to determine the situation throughout the entire monitoring area. Again, each node lies within a rigorous level architecture defined over the stretch of sea to be monitored. The upper bound of the area (landwards) is delimited by the sink node and the lower bound (seawards) is defined by the number (indefinite) of levels. So the total number of levels is given by the number of nodes and their transmission range, which determines the level's height and width.

The operating area of the network can be represented as a series of overlapping rectangles, as shown in figure 1.2. Level dimension has been chosen so that transmission range of devices allows communication only between nodes at contiguous levels, in other words it is not possible to skip a level. The height of a level is a little more than half-transmission range and is equal to 5/4 of the width. Increasing level width risks compromising communication between a node of level n and a node of level n±1. By virtue of this structure, each node resides only in one level (there are no overlapping areas) and has well defined coordinates within the area.

1.3.1 Construction and maintenance of the network

The tree structure is built up along the levels of the monitoring area. The start-up of the entire survey system envisages a downward network

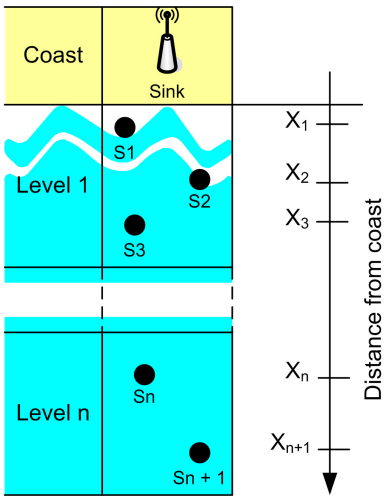

Figure 1.2: Level architecture of the tree structured model.

growth starting from the sink node; once Sink has been started, network construction begins turning on a sensor node in the first level. The new node notifies its presence sending a request containing its position, the sink receives replies communicating its identity, adds the node to its children list and marks it as "preferred" child.

Switching on another node in the same level, will produce nearly the same operation as above, but the first node does not reply to the request (level n's requests are handled at levels levels n-1 and n+1) and the second node will be the new preferred child of the sink node if it has a battery level higher than that one of the first node.

Tree construction proceeds in the same way for the other levels, but

from the second level onwards each node can choose between different roots. The choice of "preferred" root, as happens for children, is based on battery level, and variations therein can cause root changes.

The network changes continuously; node battery level varies constantly, a few nodes can switch off or be destroyed and new nodes can join the network. So, a tree maintenance and convergence policy is required, that always guarantees the delivery of active nodes data to Sink. Every T seconds each node sends to its neighbors nodes a refresh notification containing battery level in addition to surveyed parameters. It is important to specify that T period starts a few instants after the node is switched on, so notifications are not synchronized. Obviously simultaneous notifications can occur but co-channel interference problems only arise for contiguous levels. The destination node receives the essential information indicating that its neighbor is still active and updates its lists using just received data, perhaps modifying preferred root and child.

Eventually some nodes fail; this means that it will no longer send refresh notifications. If a node stops sending notifications it will be deleted from its neighbors membership lists; if the node was a preferred node then preferred root and child must also be updated according to battery level of still active nodes. This mechanism has been implemented using a timer started at the same time as the last notification acceptance.

Should a node switch off or turn on before the timer has expired, probably no one would realize, except for the fact that when turned on again it

sends a request to detect neighbor nodes. In any case this request would be discarded (could be a duplicated ID) and the node would resume working as though nothing had happened.

1.3.2 Routing and querying on the tree

The tree construction described in the previous sections places much emphasis on node battery level and thus on estimating node life time. The choice of preferred roots and children is fundamental in determining routing paths used for data flow toward Sink.

From the above described rules it should be obvious that no node has a full view of the tree. Each node knows nodes on contiguous levels (its children and roots) but does not know what lies beyond. Only the sink node can, at any time, perform a query on the tree and know all its details. Underlying nodes can use the query to know next levels but they do not know anything about the previous ones.

In the previous section we stated that refresh notifications are not synchronized. This means that not every node has the same view of the tree at every instant. This problem is solved when it is required to know precisely the network status, that is when Sink requires a tree snapshot. This event triggers a data update of all sensors reachable from Sink, giving a full view of the monitoring area.

Tree tracking, like most of the operations performed on the tree, is

recursive and is based on two simple steps:

1. the sink makes a refresh call to all its children which reply with a notification containing all data; the sink updates its children list and, if it is not empty, adds it to the procedure result;

2. the procedure is propagated recursively until the lower level (or until a tree interruption occurs) and the results returned to the upper level at each step.

This approach is not purely recursive because each root receives the first results simultaneously with propagation to next levels, not after reaching the end of the tree.

We defined 4 modes to query the nodes. Queries do not require an immediate update of the lists: each node replies with data obtained during last refresh, such that no inconsistent data are sent. In fact, in these cases only one node is responsible for data sent to upper levels. Furthermore, the data provided are at the most T seconds "old" and this can be neglected for T values within reasonable ranges. Anyway instantaneous data can be obtained performing a refresh, as previously described, before querying the tree.

All query modes are based on checking children lists owned by each root and on query propagation to preferred child. Also in these cases the recursive technique is adopted with the same specification described in the previous section.

The first query mode involves all nodes and is useful for obtaining data for all sensors contained in the network. The results can be used, for example, for calculating average values of parameters for the whole tree. The preferred root of each level is responsible for the data transmitted to the sink; this query is able to detect possible duplicated nodes on the network.

The query on one node, detected using its ID, returns all data surveyed from that sensor together with the level where the node is situated, its coordinates and its battery level. The presence of a duplicated ID is not detected because the procedure stops at the first matching node; however, retrieved information can be used to perform a consistency check with initial node distribution.

Another mode is the level query, which returns all data of the nodes in the level specified as query parameter. It can be used to verify if a level is reachable and detect tree interruptions.

Finally, the query on a point allows survey data in a particular point of the monitoring area, specified by its coordinates to be obtained. Obviously not necessarily does a node exist at the exact point specified, therefore the data obtained pertain to the sensor closest to that point. Unlike the other queries, propagation does not follow the preferred child but the child closest to the requested point; along with parameter values survey distance is also determined.

1.4 Key distribution and security

1.4.1 The "main scheme"

From what was said earlier, we may state that the tree structured network has the characteristics of a distributed sensor network, because its scale can be very large, it is dynamic and it may be deployed in a hostile area. In addition to communication security constraints due to hardware limitations of sensor nodes, this kind of network is prone to key management constraints.

The variable topology of the network, the low communication range and the frequent failures of the nodes do not allow the use of trusted third parties to exchange and distribute keys in the network. As a consequence, the only solution is to install the keys in the sensor nodes before their deployment in the network in order to guarantee secure communication with their neighbors. This technique is known as *key pre-distribution* and one of the best known schemes is the so-called "main scheme" [10].

The main scheme requires less memory storage than pair-wise key distribution schemes which need to store $n-1$ keys in each node ($n(n-1)/2$ in the whole network) quickly reaching the limit of memory space for a few thousands of nodes. It stores from few tens to a couple of hundred keys relying on probabilistic key sharing and it is divided in the three phases: key pre-distributions, shared-key discovery and path-key establishment.

In first phase a key ring is established for each node randomly choosing

a set of k keys from a large pool of P keys ($2^{17} - 2^{20}$) previously generated together with their identifiers. In the shared key discovery phase each node broadcasts their key identifier in order to discover the neighbors which share a key. At the end, in the third phase, a *path-key* is assigned to pairs of nodes which do not share a key but are connected by two or more links.

The strong point which allows the reliability of this technique is the achieved probability for two nodes to share at least one key. The random distribution, in fact, guarantees an high probability such that, for example, only 75 keys need to be distributed to have the probability $p = 0.5$ that two nodes share a key in their key ring using a pool size (P) of 10,000 keys. Next formula expresses the probability to share at least one key ($p' = 1 - Pr[\textit{two nodes do not share any key}]$) as a function of P and k.

$$p' = 1 - \frac{((P-k)!)^2}{(P-2k)!\,P!} \tag{1.1}$$

A plot of the probability functions for different values of P is shown in the figure 1.3. Actually, the plotted function is obtained using Stirling's approximation for factorial [1] because P is a very large number; the function

[1] $n! \approx \sqrt{2\pi}\, n^{n+\frac{1}{2}} e^{-n}$

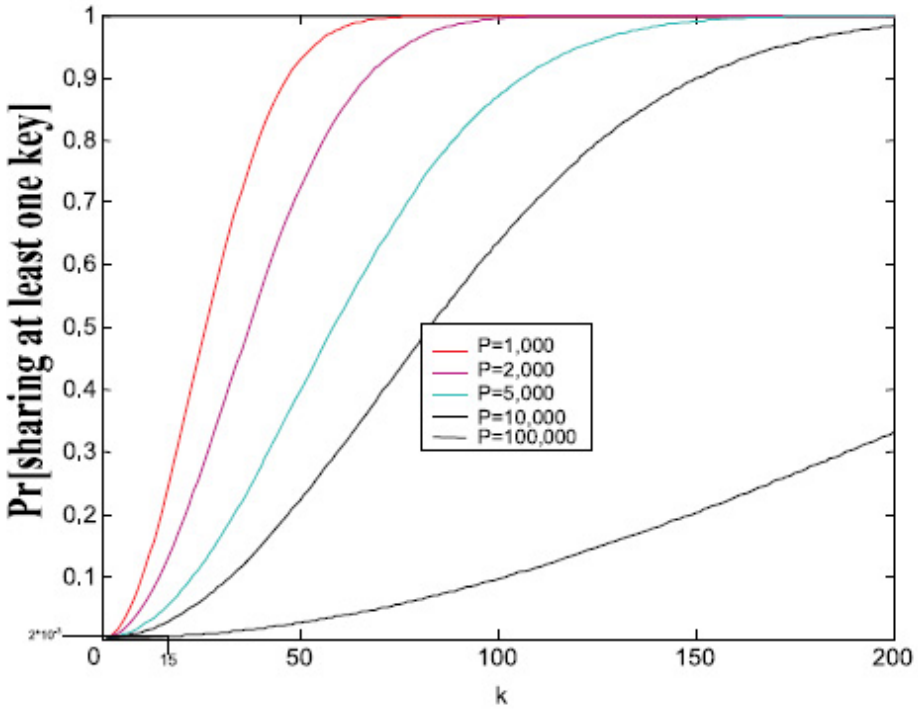

Figure 1.3: Probability of sharing at least one key when two nodes choose k keys from a pool of size P [10]

becomes:

$$p' = 1 - \frac{(1 - \frac{k}{P})^{2(P-k+\frac{1}{2})}}{(1 - \frac{2k}{P})^{2(P-2k+\frac{1}{2})}} \tag{1.2}$$

1.4.2 Nodes initialization on the tree

Nodes need a key to authenticate communications directly addressed to the sink node and another one to initialize a hashchain with its preferred root in order to send measured parameters.

As pointed out in the main scheme, the initialization of a node is done during its installation, when it can load in main memory a key from a trusted source and use it to communicate with the sink node (which obviously knows keys associated to each node) without saving it in a secondary memory. But in this way, only the first initialization of the node can be done and only being able to physically reach the node. Subsequent activations may be necessary in the event of malfunction, temporary exclusion from the tree or after attack by extraneous nodes.

In order to obtain this functionality each node has to be able to execute a hash function (e.g. SHA-1) and a secure deletion algorithm (e.g. Gutmann, Schneier, DoD-3 or other simpler ones). Moreover, the node has to store a list of keys to be used for subsequent activations (number of possible activations depends on number of keys but, then again, the device's life is limited by its battery duration).

After the first initialization, a node that has to be remotely re-activated expects to receive from the sink node the hash for the first key on the list (only the sink can compute it). As soon as this is received, the node computes the hash for the key and matches it with the one received, verifying

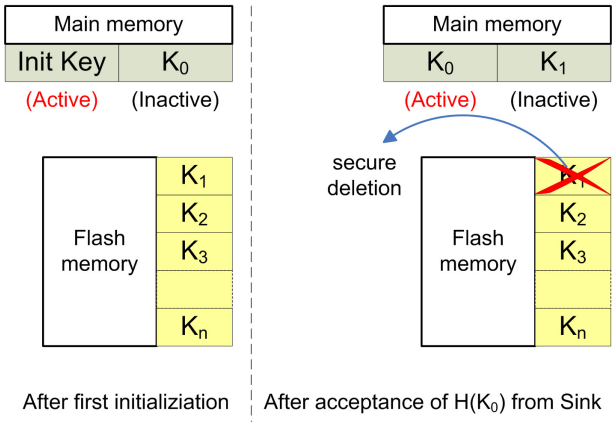

Figure 1.4: Keys activation sequence.

that activation was actually started by the sink. At this point, the node loads the second key of the list in the main memory deleting it in secure mode from flash memory. The first key (which remained in main memory since first initialization) is now active and can be used to authenticate the messages directed to Sink. The activation sequence is shown in figure 1.4. In order to initialize the chain, each node, possible root, stores the hashes of a set of initialization keys owned by underlying nodes. The pool of keys is loaded in main memory as explained in the previous section for activation keys, while the hashes of the keys can be stored in the flash memory.

The chain is started by sending the key's hash to the preferred root together with first data hash; the next block contains first data, the hash of the second and so on. The chain can be interrupted and subsequent re-initializations cannot be done using the same hash. To solve this problem

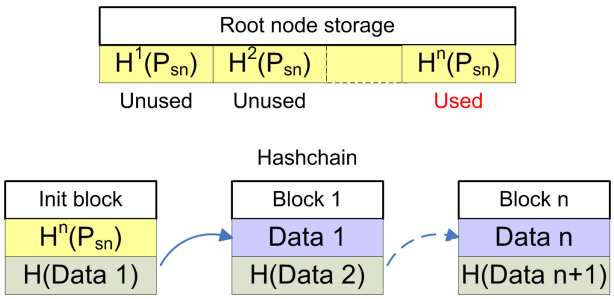

Figure 1.5: Hashchains and one-time passwords.

each root can store the hashes of different keys for each node, otherwise it can expect to receive the nth hash of the key at first initialization, the n-1th at the second and so on, creating a one-time password authentication scheme. Root node storage and hashchain are shown in figure 1.4. Using this mechanism a potential attacker wanting to enter the chain should be able to retrieve measurement data of next block using the hash of current block. Observing enough transmissions, an attacker will notice that some measurement data recur in the chain and analyzing the hashes can try to guess the content of the next block. To avoid this kind of attack, means ensuring that identical survey data generate different hashes, introducing additional data in the computation process. Introducing values obtained by a pseudo-random number generator (PRNG) can be useful but this does not significantly enhance security; inserting values obtained by a true-random number generator (TRNG) is the best solution. In simulations, the true-random values has been generated using parameter measurement

intervals: the expiry time between one measurement and another is not extremely rigid and is influenced by hardware delays due to measurement devices and clock. The delays and the total time from last measurement are measured with an accuracy higher than interruptions (e.g. milliseconds or nanoseconds) and added to data packet (which also contains unique IDs for each node). The probability that an attacker guesses the exact instant of data measurement and sending is really low.

1.4.3 Capture resilience

The most difficult problem to solve in this kind of application is the physical protection of the nodes. An attacker can steal them, copy their content, clone them or exchange them with other devices. In order to protect the nodes we need to ensure that even if the attacker does steal the node, he is unable to extract useful information for accessing the network.

In the nodes activation procedure and for using the keys for measurement data sending, we have chosen to store the active key and the next key of the activation pool in the main memory (RAM) alone, performing a secure deletion from the secondary memory (flash memory) of the loaded keys. The latter step is very important because, even after a flash memory erase operation, the information can be restored [26].

We also have to prevent the attacker from accessing the main memory content, for example by creating tamper-protection for the node which

forces node shutdown in the event of intrusion. But shutdown in itself is not sufficient, because RAM memories are also prone to data retention problems: using a key for a long period of time can cause long-term retention effects in the memory [15], leaving useful traces for restoring information. Therefore memory cells should not store data for too long a period of time (e.g. performing a bit flipping) and in the event of tamper detection it is necessary to apply a reverse current which reverses the electromigration stress due to long-term retention [15]. In this way node theft and access to its content render the node unusable for joining the network.

Chapter 2

Wearable networks and movements representation

In this chapter, after the illustration of general characteristics of wearable sensor networks or, more in general, Body Sensor Networks (BSNs), we will explain the methods we used to analyze body parameters and compare its movements.

2.1 General characteristics

The miniaturization of sensor nodes prompted the idea of applying them to the human body to monitor and analyze vital parameters or movements for a variety of applications. In this way it is possible to create a Body Sensor

Network which is the basis of a wearable computing system together with a software layer for analyzing the data according to a specific logic and a platform which enables (distance) cooperation between the network and a server for data collecting and processing.

Depending on the parameter to analyze and on the kind of sensor, nodes can be applied to different parts of the body. In a system for body movements detection a series of sensor nodes can be placed at the main joints, i.e. at those points that allow the detection of limb position and movements, as shown in figure 2.1. In case of wireless link, sensor nodes create an ad hoc network with a topology dynamically determined not only by the the relative position of the body limbs but also by the quality of the link itself.

In order to obtain movements representative data from sensors applied to the body, we need to monitor movements representative parameters; generally, this problem is addressed using IMUs which measure acceleration and rotation changes. The details of the hardware employed end the technique used to obtain this data will be given in the next chapters; next sections instead, will described the techniques we chose to represent and compare movements obtained from this kind of data.

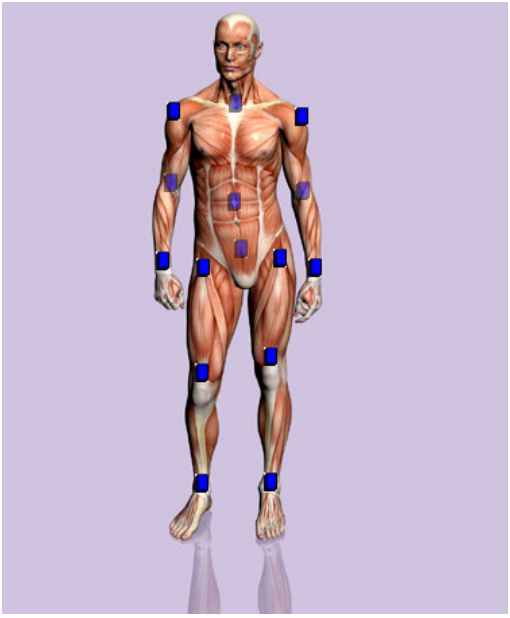

Figure 2.1: Position of sensors on the body.

2.2 Movements representation

2.2.1 The approach

Movements monitoring and representation using accelerometers and gyroscopes is a common feature for many kinds of applications, such as industrial, videogames, sports and medical. Apart from the use of IMUs, the important issue in these applications is the movement analysis technique, that is how IMUs are used. In most approaches, reading sensor data, as well as the analysis and the comparison of relative movements, is based

on the output waveform in which a pattern indicating a certain movement can be identified and evaluated. An example of waveforms from accelerations sampled during different activities is given in figure 2.2. The whole process of acquisition and assessment of a movement is substantially based on the creation and the comparison of graphs representing trends in acceleration and rotation along a certain axis during a given movement. The recognition of specific patterns in a movement performed by a reference subject forms the basis for the comparison and assessment of other subjects; this approach yields results that are not readily understandable, especially visually.

Another movement analysis technique is based on pose detection; in this case IMUs and data coming from them are used to deduct the position of a standing body, even after a movement but without track it. Sensors are mainly employed as inclinometers (tilt meters) and movements are analyzed in an attempt to deduce the actual position of the body but not the whole movement, in other words trying to identify initial, intermediates and final positions of a movement from the angles between limbs. The representation of poses is, without any doubt, easier to understand than the waveform analysis.

In our approach, we tried to view the sensor output not just as a signal in which specific patterns were identified but as a representation of the movement in space, in other words as the basis for tracking the movement in three dimensions. To achieve this, similarly to techniques that rely on

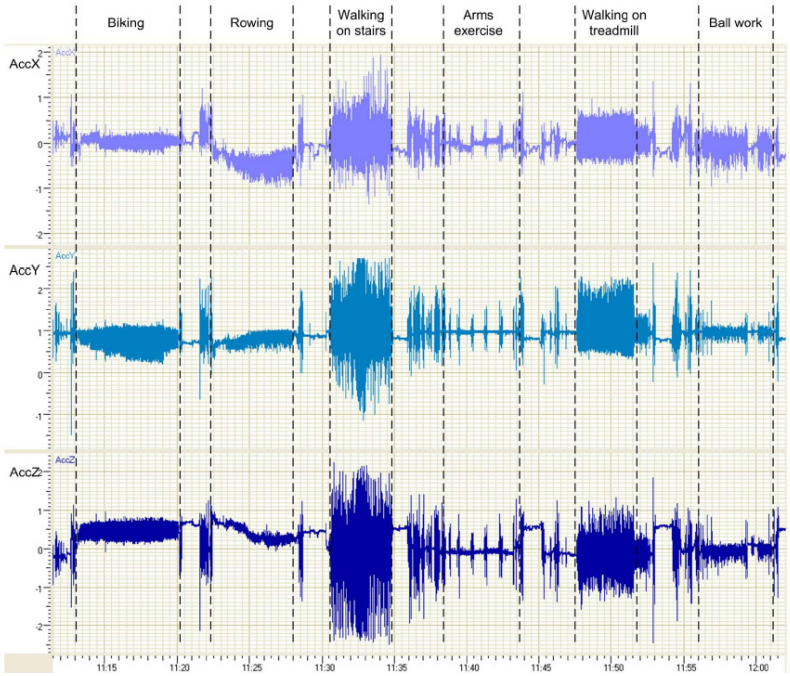

Figure 2.2: Acceleration waveforms for different activities [5] .

optical devices but, obviously, without using them (see paragraph 2.4), we created a system and comparison techniques based on visual representation of the movements, in order to provide an easily readable and understandable result that represents, as far as possible, the real movement.

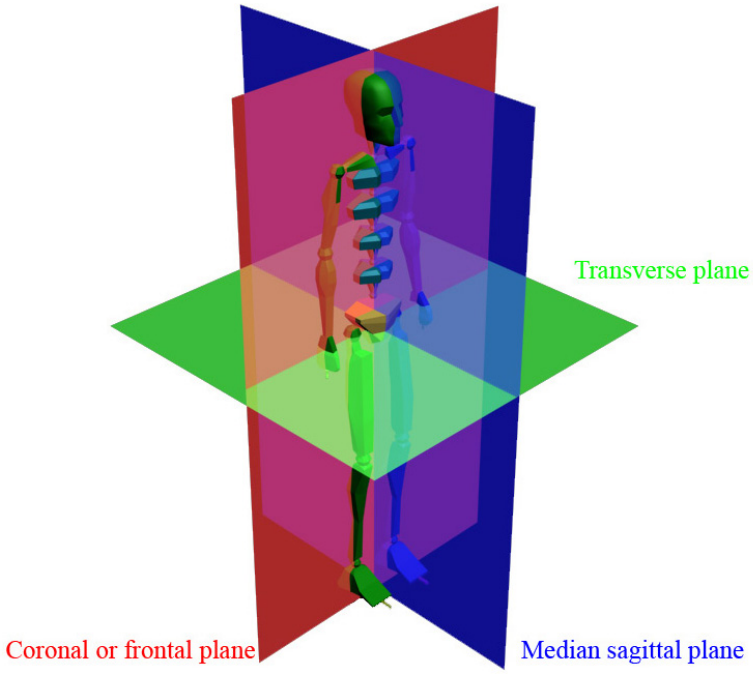

Figure 2.3: Main planes of the body.

2.2.2 Parameters

The analysis of positions and movements of a human body requires observing them from different perspectives in order to obtain a full view; we decided to set the body (the sensors network) in the anatomical planes and take measurements observing from the coronal (frontal), sagittal and transverse plane, as shown in figure 2.3.

We also decided (and needed) to define two parameters helpful to formally describe positions and movements of the body in a specific plane; these two parameters are imprint and trace, defined as follows:

- *imprint*: we define imprint of sensors in a given anatomical plane, the set of points given by the orthogonal projection of the sensors on that plane and the set of segments given by their conjunction;
- *trace*: we define trace of a sensor in a given anatomical plane, the set of segments given by the conjunction of the imprint points left by the sensor on that plane at subsequent instants.

Imprint and trace can be defined starting from a given position and from a given movement of the body at different time instants. By sampling a movement at a certain rate, we can define an imprint for each sample and then the trace left by the body during the whole movement or parts of it.

A graphical example of these parameters can be obtained starting from the athletic movement depicted in figure 2.4, showing initial and final position of the skill.

This movement is symmetric so we can focus only on on arm (the right one, for example) and view the body on the three anatomical planes as shown in figure 2.5. In these views we can highlight the main joints (the places were sensors are applied) and extract relative imprints at starting and finishing positions. In the same way we can obtain the trace of the full movement viewed at two subsequent time instants. Taking as an example

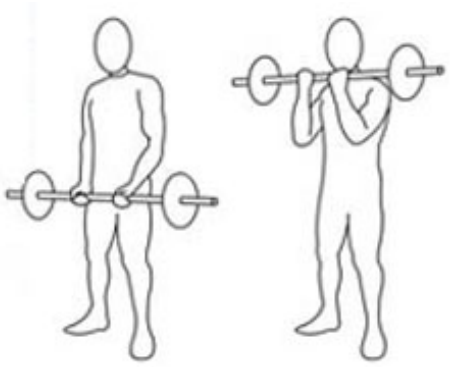

Figure 2.4: Example of athletic movement (arm curl).

the view on median sagittal plane the two imprints and the trace are shown in figure 2.6.

It can be seen that only two sensors describe a movement; in the trace they are represented with different colors to better distinguish from the others. In the same way we can define imprints and trace on the three anatomical planes obtaining simple images (they can be represented as black and white images) that can be easy compared to reference images. Next section will present the methods we used to perform this operation.

2.3 Movements comparison

Using the representation described before it is possible to efficiently compare a reference movement with a test movement with good reliability. First of all, the movement is sampled at a predetermined frequency (de-

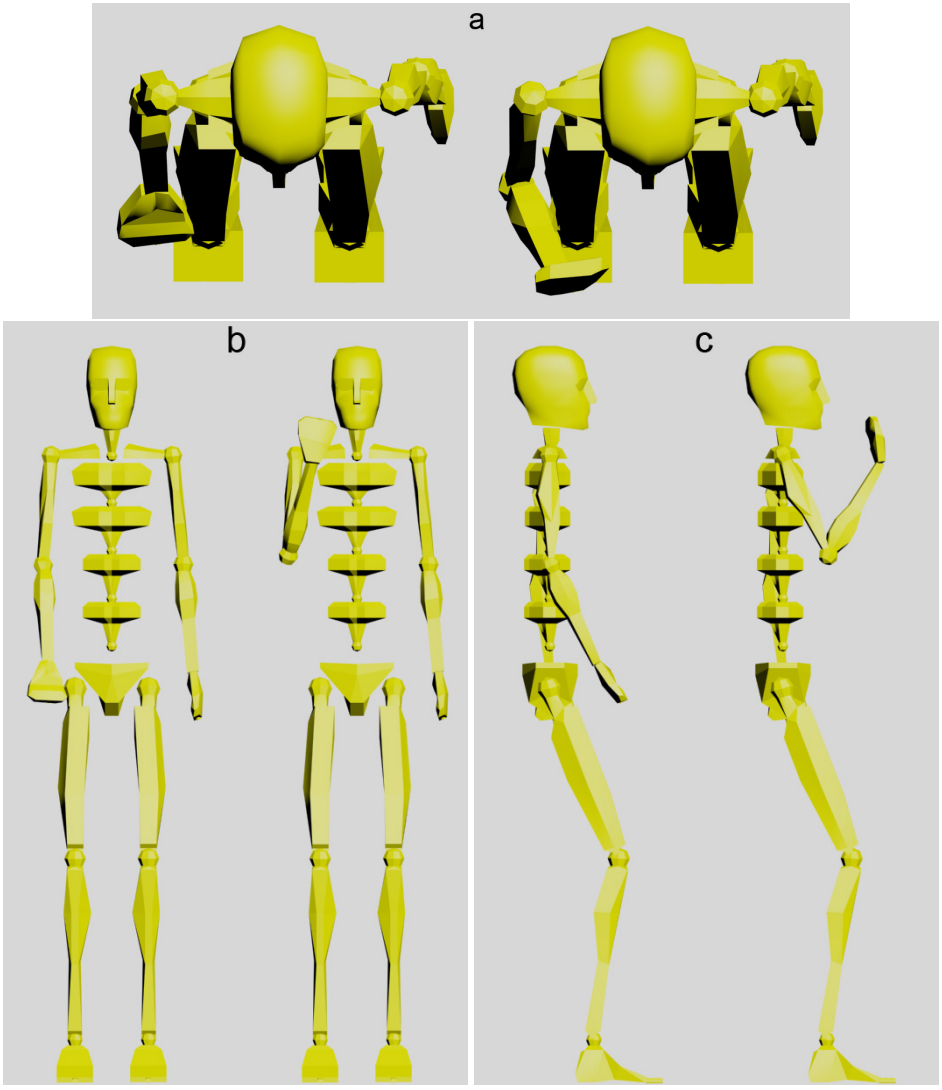

Figure 2.5: Transverse (a), frontal (b) and sagittal (c) view of the movement of figure 2.4 (right arm only).

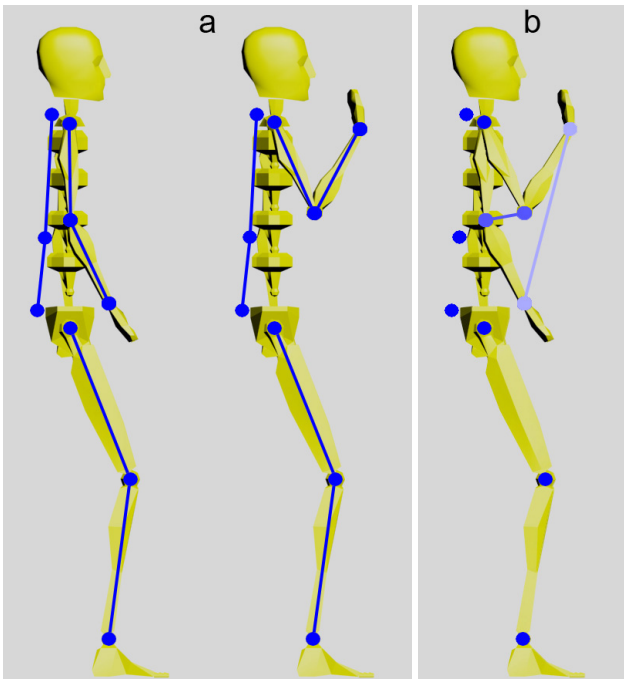

Figure 2.6: Imprints of starting and finishing positions (a) and trace (b) on median sagittal plane (right side).

pending on duration and velocity of the execution) and for each sample imprints and traces are extracted. The same procedure is followed for the reference movement.

In this way we can compare a test movement to a reference movement over the three anatomical planes or just over the movement plane (if single), giving both a numerical score and a graphical representation that is easier to understand than the waveforms produced by the sensors. We defined this procedure *execution assessment* and it will be described in the following subsection; we also defined a procedure for the calculation of the *best execution*, which calculates the ideal best execution of a certain movement for a specific subject (2.3.2).

2.3.1 Execution assessment

The first step of this procedure is the translation and scaling of the images obtained from movements sampling: dimensions need to be standardized with respect to those of the reference imprints. Hence, the imprint is translated such that the position of the sensor placed in the lumbosacral area coincides with its counterpart in the reference imprint. The imprint is then scaled and rotated so that it too coincides with the sensor positioned in the nape of the neck. By so doing, the segment representing the vertebral column is made to coincide to the greatest extent possible.

The algorithm

Since imprints can be represented as black and white images, they can be compared simply by pixel subtraction and they do not need special pre-processing like edge thinning or thresholding, often used in this kind of application on grayscale images. In simulations and tests, we calculated the distance between imprints as absolute value distance and quadratic distance, the latter yielding better results. The value of the distance provides a score index for the evaluation of single imprints and thus of the athletic movement as a whole.

The imprint to be analyzed can be compared with a single imprint, extracted from a reference physical exercise whose execution is considered ideal, or with a set of reference imprints. In the second case we can obtain a kind of reference training set to which classification algorithms can be applied as in a recognition problem. In this case, the aim is not to perform a recognition but to evaluate the distance from a set of samples already classified, evaluated and considered as reference models: using the K-Nearest Neighbor algorithm we obtain a classification in ascending order of distance from samples to which a score index is associated (perfection of the execution). An example is given in table 2.1.

Starting from this kind of classification, different possibilities exist for calculating the score of an imprint. The first is to decide a K value for the K-NN algorithm, keeping the similarity percentage or the score index as

Table 2.1: Evaluation classification of an imprint.

Position	**Similarity**	**Reference sample**	**Score index**
01	93.9131%	n. 12	96/100
02	93.8016%	n. 05	94/100
03	93.5187%	n. 09	98/100
04	93.2145%	n. 17	91/100
05	92.9734%	n. 03	89/100
06	92.7559%	n. 21	92/100
07	92.4589%	n. 25	88/100
08	91.6753%	n. 10	90/100
09	91.1267%	n. 07	86/100
10	90.3426%	n. 16	87/100

score of the imprint to be analyzed. Using this method, the best choice in tests was found to be a K value equal to one and to use the score index as evaluation score: the results obtained proved to be the most consistent with the assessment of a hypothetical board of examiners.

As an alternative we can calculate the average percentage of similarity and average score index for the first K samples of the classification and use these values for assigning the score. In this way we obtain an evaluation that accounts for the different reference samples as far as similarity percentage is concerned and are not dependent upon possible errors in

samples index score assigning.

Assessment correctness

In order to verify the correctness of the evaluation computation, we performed a classification procedure of the imprints to evaluate using reference samples as training set.

As usual, we used the K-NN algorithm and the best value of K proved to be 1. The results obtained using this procedure proved to be the larger training set dimension the more accurate, and they allowed us to establish whether a certain imprint (already classified and evaluated) was classified as a training set imprint of the same class and the same score. For small training sets (less than 100 samples) correct classification was over 80%, reaching maximum values (over 98 %) for about 1000 samples.

2.3.2 Best execution

This analysis method calculates the best execution for a given person based on the starting and finishing positions of the athletic movement (made as similar as possible to the positions of an ideal execution) and a set of parameters and constraints to be observed during execution. The result of this calculation constitutes the reference for evaluating the real movement of the same subject.

The reasons for implementing and using this method is the impos-

sibility, in some cases, to correctly compare sensor movements and their relative positions because proportions of bodies on which sensors are worn can differ, even after scaling and rotation of the imprints. These differences can result in disadvantages already in the starting and finishing positions of the athletic movement. In this way the computation results are compared with the execution examined which can be evaluated in its entirety and independently of possible physical limits.

In the computation process we have to observe constraints imposed by execution specifications or physical constraints imposed by the impossibility to perform certain movements. This kind of constraint has been imposed by verifying relative positions between sensors and by imposing that at given instants given sensors lie in specific positions of the plane and by verifying that the distance between adjacent sensors (that can represent the length of a certain limb) do not change during execution. Once all constraints are satisfied, the computation of the best execution is simply a problem of minimization on graphs (traces and trajectory previously defined) and the result can be easily compared with others. In order to render the trajectory defined by the computation as natural as possible and to compare it with the trajectory of a real athletic movement, the sampling frequency needs to be sufficiently high to delineate a trace using linear interpolation.

2.4 Other approaches

The other major approaches used to solve the problem of human body representation are the marker-based optical motion capture and the contour finding (marker-free) techniques. In the first one the body is fitted with optical markers and represented similarly to the description given before; in the second one, a cameras-based system captures the image of the body from multiple viewing positions and tracks it without any marker applied on the body. Both use cameras or video sequences to capture the image of the body but marker-based approaches can use simple video acquisition (monocular or stereo), targeted to just capture the marker position, instead of a fine body representation from multiple views needed by marker-free approaches.

The main advantage of these approaches compared to sensor based techniques is the precision of the representation that can be achieved: sensors are prone to errors, need proper calibration and the final result is not always fine. However, also optical techniques have to counteract different problems. First of all, the video capturing procedure requires the absolute absence of objects between cameras and the body: this can be a great limitation in many environments or applications.

Marker-based approach generally uses infrared reflection in order to track the movements but retro-reflective markers are indistinguishable, so the first fundamental problem is to *identify* the markers and associate

them to the limb which they are attached to. Moreover, to reconstruct the three dimensional position of a marker, it has to be visible from at least two cameras [16]; the *occlusion* is the second fundamental problem for marker-based techniques and it needs methods to reconstruct the position and orientation of the limbs even if a significant number of markers is occluded.

In the contour finding techniques the aim is not only the reconstruction of human body positions but also the whole body's silhouette, its surface including skin, muscle deformation and garments. This approach can achieve a more realistic representation of the subject but there are many limitations in positions and movements that cannot be detected due to insufficient visibility. In addition the body reconstruction and motion estimation processes are done off-line, making the technique unsuitable for real time or live broadcast applications. Also the surfaces and textures representation is not always sufficiently realistic because it requires the exact object geometry for an unblurred view-dependent texturing [7].

In conclusion we can state that sensor-based approach, in spite of the impossibility to represent graphical details noticeable only by video capture, allows a real time movement representation with a good level of accuracy which can be modulated depending on the technique of data processing applied to the sensors themselves as we will see in the next chapters.

Chapter 3

IMUs set up

In this chapter, we will describe, step by step, the operations we did to obtain 6 degrees of freedom Inertial Measurement Units starting from a board equipped with a microcontroller, a digital tri-axis accelerometer and a digital tri-axis gyroscope.

3.1 The ADXL345 evaluation board

The hardware platform we used to obtain our IMUs is the ADXL345 evaluation board produced by SparkFun™ Electronics. The board, as the name suggests, is equipped with the accelerometer ADXL345 by Analog Devices but also with a microcontroller ATMega328p by Atmel® Corporation and other important components. Figure 3.1 shows a full-size image of the

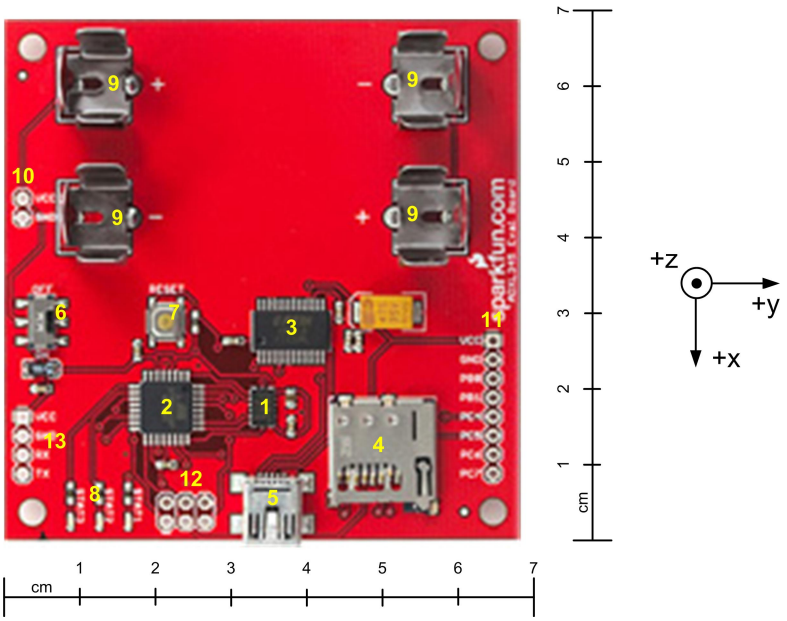

Figure 3.1: The ADXL345 evaluation board.

board (68 mm × 69 mm) with the indication of its components and the axes orientation of the accelerometer; the description of each component is given below.

1. ADXL345 accelometer: it gives the name to the board and it will be deeply described in section 3.2;

2. ATMega328P microcontroller unit: it allows functioning and coordination of all components in the board. More details will be given in the next subsection;

3. FTDI FT232RL USB UART interface Integrated Circuit: it is a single chip USB to serial UART interface which provides entire USB protocol handling without the need to load a specific firmware in the ATMega microcontroller. It also integrates clock generation and support multiple transfer rates (up to 3 Mbaud) at 7 or 8 bit of data;

4. microSD card socket: it supports FAT32 formatted microSD cards for data logging and storage purposes;

5. mini-USB type B port: connection port for serial communication;

6. power switch: to switch on/off the board;

7. reset button: it switches off and immediately restarts the board clearing the registers;

8. three status LEDs (from left to right, LEDs STAT3, STAT2 and STAT1): they have a different behavior depending on the status of the board. All three blink in sequence at power up and then, if a microSD card is present datalogging is enabled and they blink on and off four times, otherwise two times. When datalogging is started, the first LED (STAT1) signals writing operations on the card. STAT2 and STAT3 LEDs are used to indicate interrupts on INT1 and INT2 pins respectively.

9. AA battery clips: to hold two AA batteries;

10. power header (from the top to the bottom, pins VCC and GND): as an alternative to AA batteries, the board can be powered connecting a source to these pins without exceeding 3.3 Volts. They can also be used as power source to external devices;

11. external I/O header (from the top to the bottom, pins VCC, GND, PB0, PB1, PC4, PC5, PC6 and PC7): these pins provides access to the board's power and ground signals and, most importantly, to the "spare" pin from ATMega328P. Using these pins it is possible to connect external hardware and interact with it by writing a specific firmware;

12. ISP header (from left to right and from the top to the bottom, pins VCC, MOSI, GND, MISO, SCK and RESET): a 6 pins header which allows standard programming methods using ISP (In System Programming) devices (e.g. AVRISP programmer);

13. UART pins (from the top to the bottom, pins VCC, GND, RX and TX): using these pins it is possible to access serial communication at TTL level.

3.1.1 The ATmega328P MCU

The ATmega328P microcontroller unit (MCU) is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture [3]. It

embeds a 8 bit AVR cpu, a 32 KB self-programming Flash Program Memory, a 2 KB SRAM, a 1KB EEPROM and a 8 channel 10 bit A/D converter. Its operating voltage is between 1.8 and 5.5 V and it can reach up to 20 MIPS throughput at 20 MHz. The device mounted in our boards is supplied in the 32A package type, with a body size of 7 mm $\times$ 7 mm and a body thickness of 1.0 mm; the block diagram of the microcontroller is shown in figure 3.2.

The role of ATmega328P in the ADXL345 evaluation board is to coordinate the functioning of each component of the board by executing tasks and operations specified in the firmware loaded into the 32KB Flash Program Memory through the ISP header or via USB using an Arduino IDE and the Arduino Serial Booloader installed in the evaluation board. The board also comes with a firmware loaded into the ATmega328P's flash memory which performs startup operations, datalogging and provides a command interface to modify the registers on the ADXL345 accelerometer (by connecting the board to a terminal program via a Virtual Com Port and the mini USB connector).

The interaction between the MCU and the other components is done through the MCU's data registers. Some bits of these registers also correspond to a specific pin in the microcontroller and to a spare pin in the board. The figure 3.3 shows the pinout of the ATmega328P. Port B, Port C and Port D registers (whose pins are listed as PBx, PCx and PDx in figure 3.3) are registers of the I/O ports, other registers like SREG, UCSR or ADCSR be-

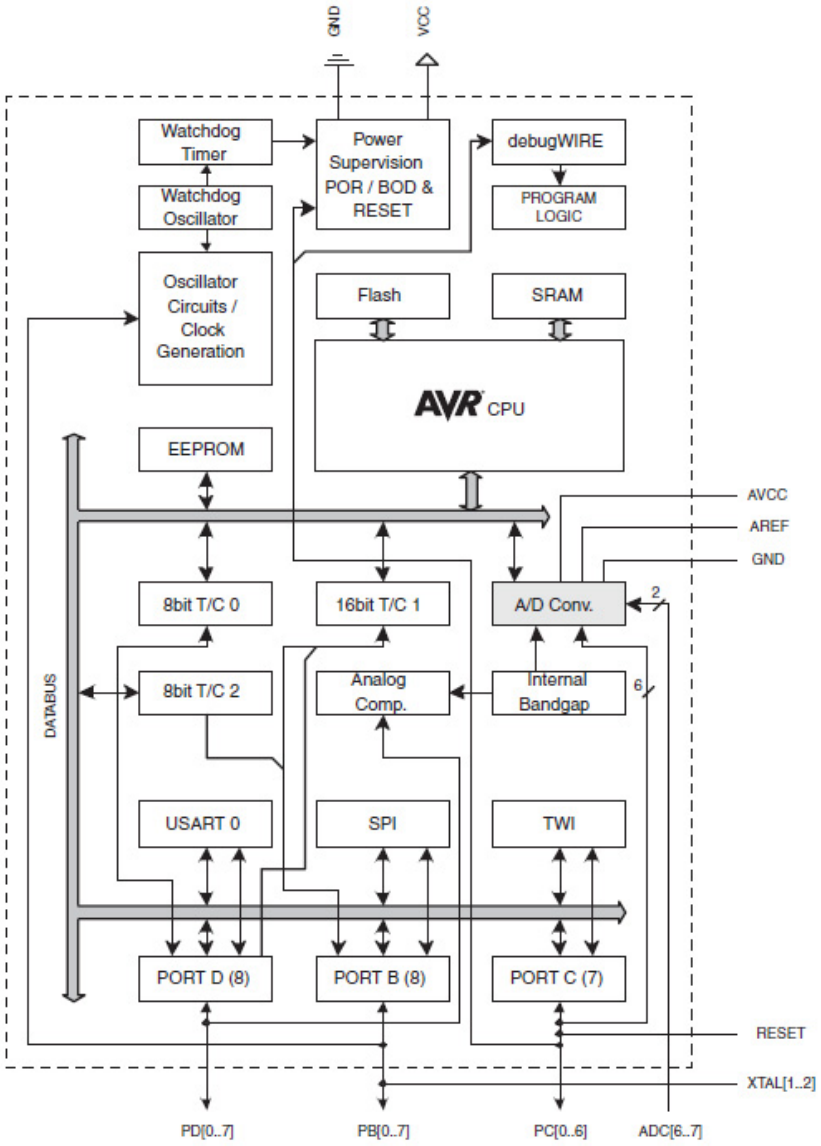

Figure 3.2: The ATmega microcontroller block diagram.

long to other blocks of the MCU. As can be seen in the pin configuration, I/O ports can have different functions: in the ATmega microcontroller the functions of some pins can be alternated by ovverriding the signals. For example, the PB0 pin (corresponding to bit 0 of PortB register) can act as input capture for Timer/Counter1 (ICP1 function), as output for the system clock (CLKO function) or as source for the interrupt 0 (PCINT0 function). This characteristic of the ATmega328P was very useful for our purposes and allows us to successfully connect the ITG3200 gyroscope using the spare pins on the board. The details of these operations and the functions we used will be given in section 3.4.

Although the ATmega328P allows frequencies up to 20 MHz, we equipped our boards with two AA batteries with a total amount of 3.0 Volts. As can be seen in figure 3.4, this voltage imposes a limitation on the maximum reachable frequency for the processor: the limit at 3.0 Volts is around 12 MHz, that is 12 MIPS (Million Instructions Per Second[1]) throughput. However, the ATmega328P allows selecting among 12 predefined frequency values and the closest to 12 MHz is 11.0592 MHz (the next is 14.7456 MHz).

In order to come through this limitation, we could have powered the board with a higher voltage but the ADXL345 does not support more than

[1]Assembly instructions. The ATmega328P has a rich instruction set composed of 131 instructions. Most of them require one or two clock cycles, but other ones require three or four cycles

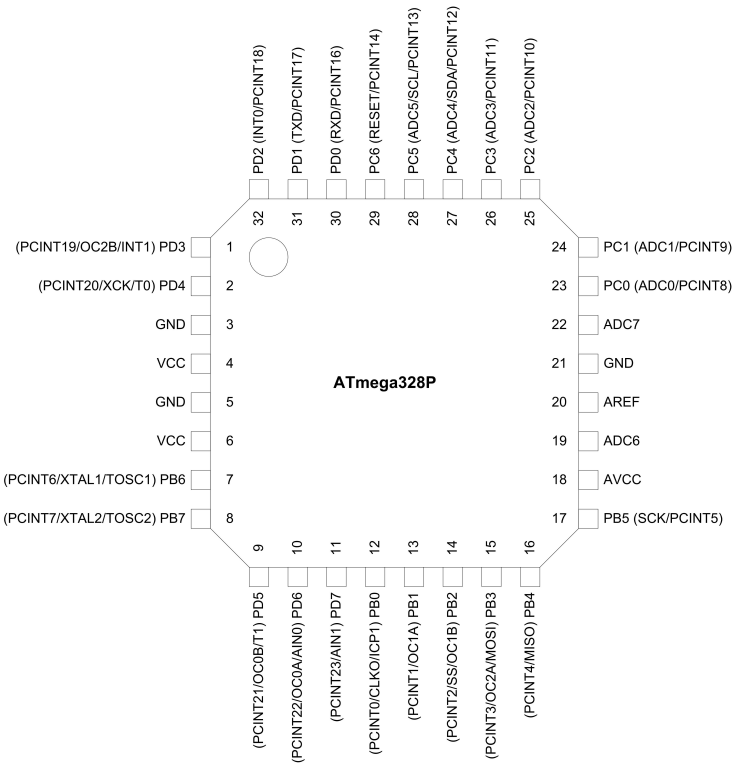

Figure 3.3: Pin configuration of ATmega328P (top view).

3.6 Volts, so the limitation is also due to this fact.

3.2 The ADXL345 accelerometer

Every component in the evaluation board acts for it. The ADXL345 is a high resolution digital triaxial accelerometer; its measurement range is between $\pm 16g$ at 13 bits (default ±2g at 10 bits). The plastic package in which the

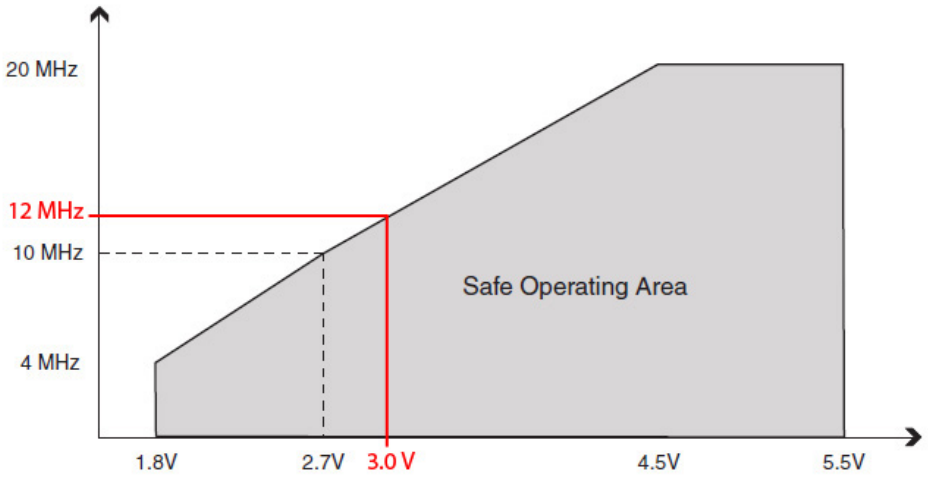

Figure 3.4: The ATmega328P's speed grades.

accelerometer is supplied is a 14-Terminal Land Grid Array (LGA) with a size of 3 mm × 5 mm × 1 mm (thickness); these small and thin dimensions, together with a supply voltage range from 2.0 V to 3.6 V and a temperature range from −40 °C to 85 °C, allow the use of the ADXL345 in tiny and low power devices and in a large number of applications. Figure 3.5 shows its functional block diagram.

As can be deduced from the name of the serial I/Os, the digital output of the accelerometer is accessible using either a SPI (Serial Peripheral Interface) or I^2C (Inter Integrated Circuit) interface; the output is formatted as 16 bits two's complement, where the least significant bit (LSB) corresponds to 3.9 mg: this high resolution allow the device to detect inclination changes

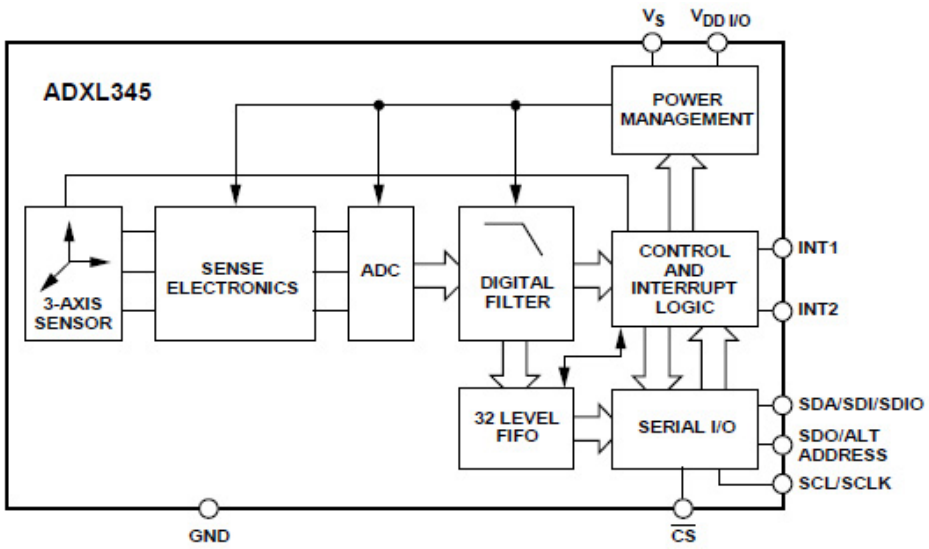

Figure 3.5: The functional block diagram of the ADXL345 accelerometer.

less than 1°. There are different resolution levels the user can choose from.

The acceleration sensor is a polysilicon structure suspended by springs over the surface of a silicon wafer; differential capacitors measure the deflection of the structure caused by both static (gravity) and dynamic (motion or shock) acceleration; this simple structure, together with the sense electronics provides other sensing functions like free-fall detection, tap sensing (single and double taps) or activity/inactivity sensing. The 32 level FIFO buffer stores data and optimizes the transfer to an external component also reducing the power consumption.

Another important feature of the ADXL345 is the output data rate; it

can be selected by the user from 0.1 Hz to 3200 Hz by setting the 8-bit BW_-RATE register (address 0x2C). The value 1 in the bit 4 of this register enables the low power mode, in which it is possible to choose only from six rates (12.5, 25, 50, 100, 200 and 400 Hz). The output data rate also determines the bandwidth, which corresponds to its half, and the power consumption which automatically scales with the bandwidth (from 23 μA to 140 μA in normal power mode); the default rate is 100 Hz, corresponding to a bandwidth of 50 Hz and a consumption of 90 μA in normal power mode (50 μA in low power mode). In the ADXL345 evaluation board, the default firmware limits the maximum frequency to 400 Hz, due to the bit rate imposed by the UART: in our tests we bypassed this problem by removing some printing operations that constrained the data rate.

Figure 3.6 shows the pinout of the ADXL345; the description of each pin is given in table 3.1. From these descriptions it is possible to deduct the connection between the ADXL345 and the ATmega328P and also between the ADXL345 and some functional part of the evaluation board. The interrupt pins (8 and 9) are used for driving interrupts and their functions (active high) are enabled setting the appropriate bit in the INT_ENABLE register (address 0x2E); the mapping to either the INT1 or the INT2 pin is done through the INT_MAP register (0x2F), where any bits set to 0 send their respective interrupts to INT1 pin and bits set to 1 send their respective interrupts to INT2 pin. The eight available interrupts are DATA_READY, SINGLE_TAP, DOUBLE_TAP, Activity, Inactivity, FREE_FALL, Watermark

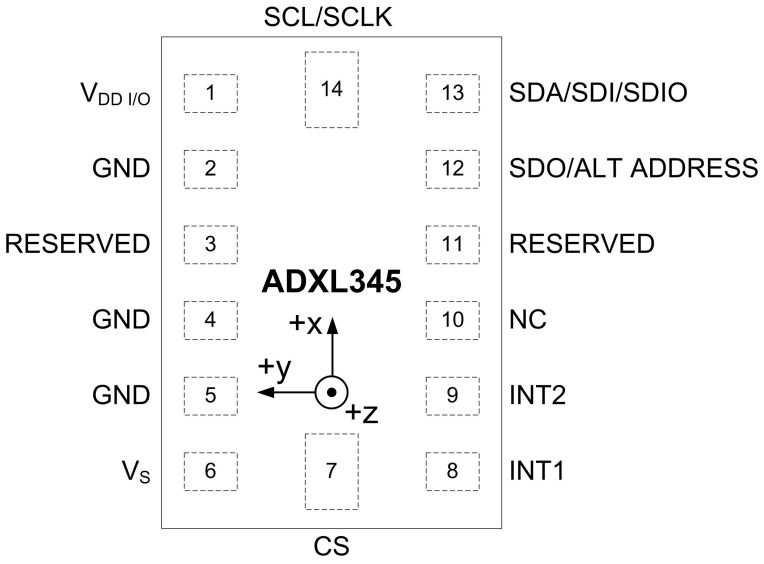

Figure 3.6: Pin configuration of ADXL345 (top view).

and Overrun; in our boards the DATA_READY interrupt (set when new data is available and cleared otherwise) is mapped to INT1 pin.

Pins 12, 13 and 14 can be used for both SPI and I^2C serial communication protocols: the ADXL345 always operates as a slave. In our evaluation boards the ADXL345 and the ATmega328P (master) communicate using the SPI protocol and they are connected through the 4-wire configuration as shown in the figure 3.7.

Table 3.1: ADXL345's pin description

Pin No.	Pin name	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage
2	GND	This pin must be connected to ground
3	RESERVED	Reserved. This pin must be connected to VS or left open
4	GND	This pin must be connected to ground
5	GND	This pin must be connected to ground
6	V_S	Supply Voltage
7	$\overline{CS}$	Chip Select m
8	INT1	Interrupt 1 Output
9	INT2	Interrupt 2 Output
10	NC	Not Internally Connected
11	RESERVED	Reserved. This pin must be connected to ground or left open
12	SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Alternate I^2C Address Select (I^2C)
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire)
14	SCL/SCLK	Serial Communications Clock. SCL is the clock for I^2C, and SCLK is the clock for SPI

3.3 The ITG-3200 gyroscope

The other sensor we employed to create our IMUs is the digital 3-axis gyroscope ITG-3200 by InvenSense; it is the world's first digital-output gyroscope realized in a single chip (all the three axis in a single chip) and marketed in a QFN (Quad Flat No leads) package whose sizes are only $4\,mm \times 4\,mm \times 0.9\,mm$ (thickness). We bought it already mounted in a breakout board (realized by SparkFun^{TM} Electronics) which provides spare pins for simplifying the connection to a system processor; figure 3.8 shows an enlarged photo of the board together with its size ($15\,mm \times 18\,mm$) and the name of each pin:

The core of ITG-3200 consists of three independent MEMS (Micro

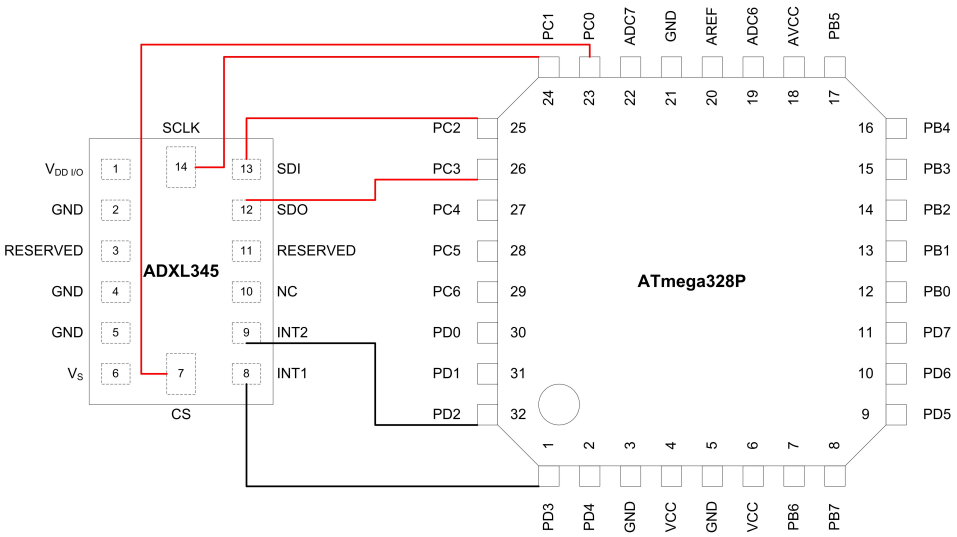

Figure 3.7: Connection between ADXL345 and ATmega328P (SPI 4-wire links in red).

Electro-Mechanical Systems) gyroscopes which, using the Coriolis Effect, detect the rotational rate about the X (roll), Y (pitch), and Z (yaw) axes. These three signals are amplified, demodulated and filtered in order to obtain three voltages proportional to the angular rates and then digitalized by three 16-bit ADCs (Analogs-to-Digital Converters) directly connected to each gyroscope. In addition to the gyroscopes, the ITG-3200 embeds a temperature sensor (measurement range from −30 °C to 85 °C) also connected to an ADC: all ADCs outputs are made available in the Sensor Data Registers (2 read-only 8-bit register for each output, 16-bit twos complement data) and are accessible through the I^2C Serial Interface (at up to 400 kHz)

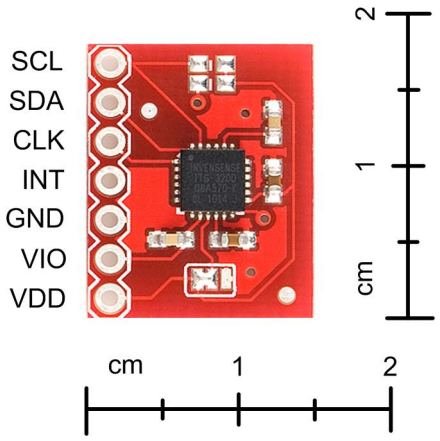

Figure 3.8: The ITG-3200 breakout board.

perhaps using the interrupt function for determining when new data is available. The registers involved in the data storing are GYRO_XOUT_H, GYRO_XOUT_L (respectively for the most significant and the less significant half of the data), GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H, GYRO_ZOUT_L for the angular rate on each axes and TEMP_OUT_H and TEMP_OUT_L for the temperature; the details of data coding will be given in the next chapter.

Figure 3.9 shows the complete functional block diagram of the ITG-3200. The optional clock provides the possibility of using different internal or external clock sources for the internal synchronous circuitry (ADCs, signal conditioning, registers); the internal sources for generating the clock are any of the X, Y, or Z gyros' oscillators with an accuracy of ±2% over

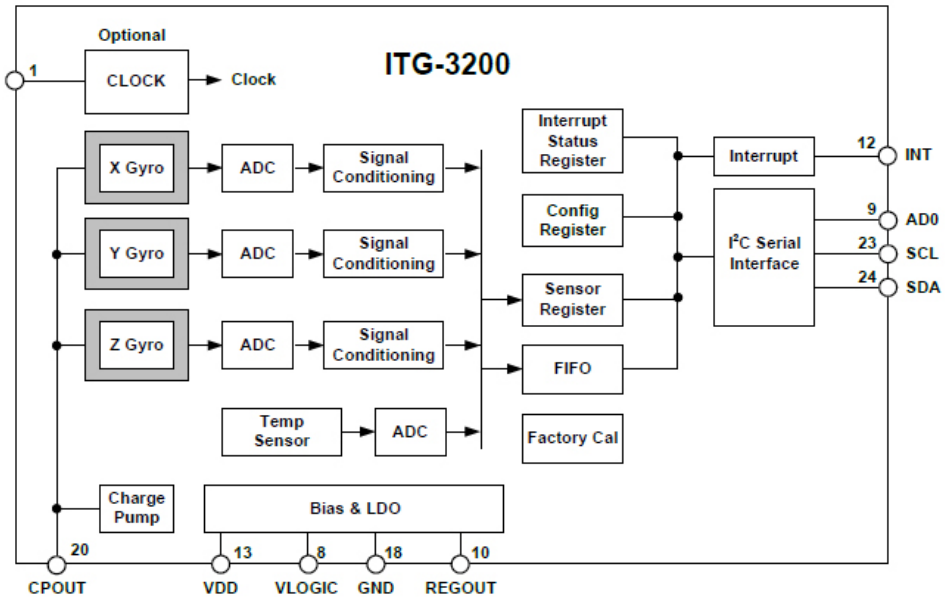

Figure 3.9: The functional block diagram of the ITG-3200 gyroscope.

full temperature range and a mechanical frequency of 33, 30 and 27 kHz respectively and a relaxation oscillator less accurate than the three gyroscopes. The external sources are clock inputs of 32.768 kHz or 19.2 MHz to synchronize with the system clock.

The operating voltage of ITG-3200 is between 2.1 V and 3.6 V with a current consumption of 6.5 mA (5 µA in sleep mode); the internal Charge Pump generates a high voltage (25 V) required by the gyros' oscillators. Even the I^2C interface has flexible voltage with a VLOGIC (digital IO supply voltage) range from 1.71 V to VDD (positive power supply voltage). All

these structural and electrical characteristics allows the ITG-3200 to reach a measurement range of ±2000°/s (with scale factor of 14.375 LSB/(°/s)) and a startup time of only 50 ms.

In figure 3.10 the pin configuration of ITG-3200 and the orientation of its axes of sensitivity and the polarity of rotation is shown; the description of each pin is given in table 3.2. As described before, the CLKIN pin provides the possibility of connecting external clock sources; the INT allows the detection of interrupt signals and the reading of new data as soon as they are available. The CPOUT pin is the connection for the charge pump capacitor; other three ceramic capacitors (whose specifications are given in the ITG-3200 datasheet [17]), should be connected to pins VLOGIC, REGOUT and VDD in order to obtain a typical operating circuit.

Pins 23 and 24 (SCL and SDA respectively) realize the serial I^2C communication with a system processor: like the ADXL345, also the ITG-3200 always operates as a slave. Apart the VDD pin (which powers the interface), another pin involved in this communication is the AD0 pin (No. 9) which contains the Least Significant Bit (LSB) of the slave address of the device; ITG-3200 devices typically has the 7-bit address b110100X, stored in the WHO_AM_I register (address 0x00). The possibility of specifying the last bit of the address allows the connection of two ITG-3200 devices to the same I^2C bus, resulting in an very useful feature for this kind of device and application.

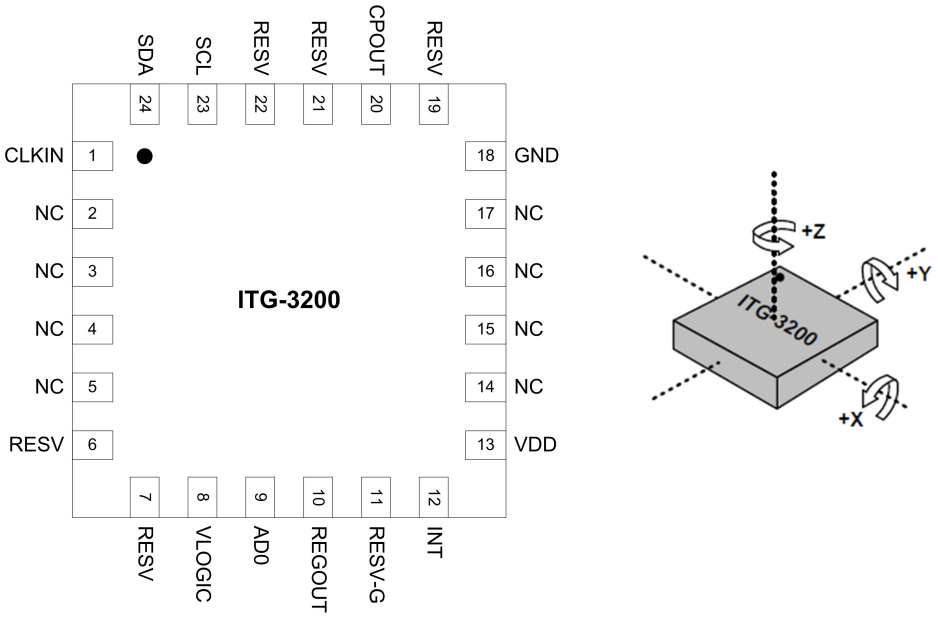

Figure 3.10: Pin configuration of ITG-3200 (top view) and axes orientation.

3.4 Connecting the ITG-3200 to the board

In order to obtain our IMUs, we needed to connect the ITG-3200 breakout board to the ADXL345 evaluation board, that is realizing the communication between ITG-3200 and ATmega328P and obtaining an integrated output with rotation data too. This step is not trivial and improper connection may result in irrecoverable hardware failures.

First of all, we had to identify the correct pins on each chip (ATmega328P and ITG-3200) and then evaluate the possibility of connection to

Table 3.2: ITG-3200's pin description

Pin No.	Pin name	Description
1	CLKIN	Optional external reference clock input. Connect to GND if unused
8	VLOGIC	Digital IO supply voltage. VLOGIC must be ≤ VDD at all times.
9	AD0	I^2C Slave Address LSB.
10	REGOUT	Regulator filter capacity connection.
12	INT	Interrupt digital output (totem pole or open-drain).
13	VDD	Power supply voltage.
18	GND	Power supply ground.
20	CPOUT	Charge pump capacitor connection.
23	SCL	I^2C serial clock.
24	SDA	I^2C serial data
6, 7, 19, 21, 22	RESV	Reserved. Do not connect.
11	RESV-G	Reserved. Connect to ground.
2, 3, 4, 5, 14, 15, 16, 17	NC	Non Internally connected. May be used for PCB trace routing.

the corresponding spare pin on the boards (if existing). We chose to maintain a single power supply source, so we accessed pins VCC and GND on the ADXL345 evaluation board to obtain these signals; we used the spare pins highlighted with the number 11 in figure 3.1. The typical operating voltage is 2.5 V for both sensors and also their voltage ranges are nearly the same, so two AA batteries are a perfect power source.

In the same area of the board there are also the spare pins from ATmega328P; in the section 3.2 we said that the ADXL345 supports both SPI and I^2C serial communication protocols; this could open the possibility of using only the I^2C protocol for communications between ATmega328P and ADXL345 and between ATmega328P and ITG-3200 (which does not

support SPI). Even if this could result in a simplification of the firmware, we preferred not choosing this way but we wanted to maintain two completely separated communication channels and buses; moreover, we could not take advantage of the possibility of the ITG-3200 to connect two devices in the same bus (because they must be both ITG-3200 devices) and the ADXL345 reaches the highest data rates only using SPI.

We can identify I^2C pins of ATmega328P in the figure 3.3: pins 27 and 28 (PC4 AND PC5) have respectively the alternate functions SDA (Serial DAta line) and SCL (Serial Clock Line), the two lines needed by this protocol for data transmission and clock purposes. Using these pins, we can already prompt the connection schema between ATmega328P and ITG-3200 as shown in figure 3.11.

PC4 and PC5 are spare pins from ATmega328P on the ADXL345 evaluation board, so they can be connected to SDA and SCL spare pins on the ITG-3200 breakout board; figure 3.12 displays a schematic representation of the connection.

Since we decided not to provide ITG-3200 with an external clock source, the CLK pin is connected to ground (GND); VIO is equal to VDD and both are connected to the positive power supply source (VCC). The interrupt of the ITG-3200 breakout (INT pin) is not connected: we preferred to use only the interrupt function of the ADXL345 and perform reading operation on ITG-3200 at the same time. In this way we had to manage only one interrupt, that is only one blocking operation in the firmware; the higher

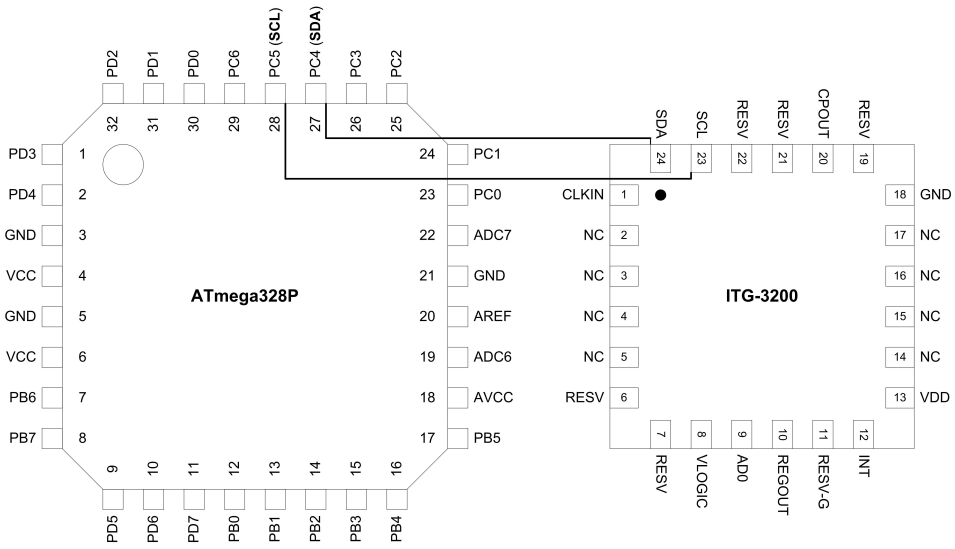

Figure 3.11: Connection between ATmega328P and ITG-3200.

sample rate of ITG-3200 (from 1 kHz to 8 kHz) gave us the certainty to always get new data at ADXL345's interrupts.

By soldering suitable connectors on spare pins and realizing the links illustrated before by copper wires, we obtained the IMU shown in figure 3.13; the ITG-3200 breakout board is placed in order to comply with the axes orientation of ADXL345 and make the center of the reference systems as close as possible.

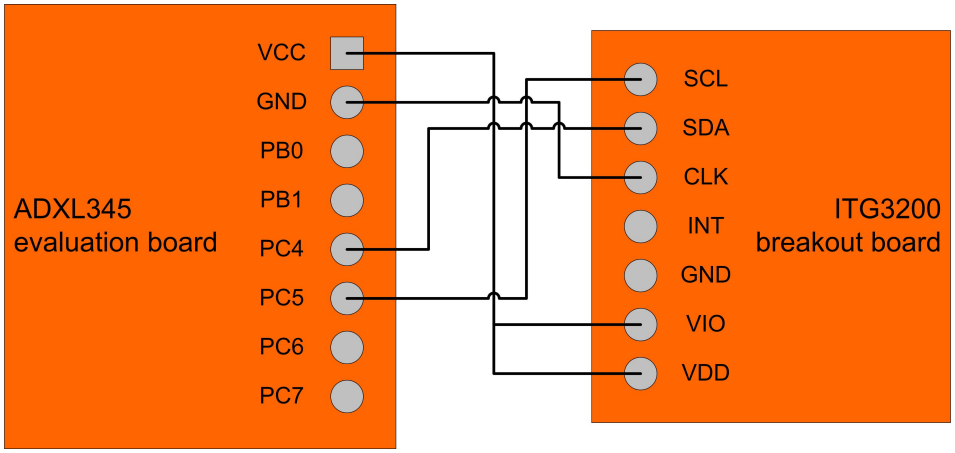

Figure 3.12: Connection between ADXL345 evaluation board and ITG-3200 breakout board.

3.5 Programming the IMU

Figure 3.13, apart from two AA batteries and a microSD card, shows a component of the ADXL345 evaluation board not present in figure 3.1: there is a 6-pin header on the left side of the miniUSB connector. We soldered this header for plugging an in-system programmer into the board.

In the section refATmega328P we said that the evaluation board comes with a firmware installed on the ATmega328P which performs basic I/O operations and also provides a command interface for modifying the value of accelerometer's registers. Obviously, once added the gyroscope, this firmware does not fulfill our needs anymore, but also for the use of the accelerometer is not the best choice because every change in the regis-

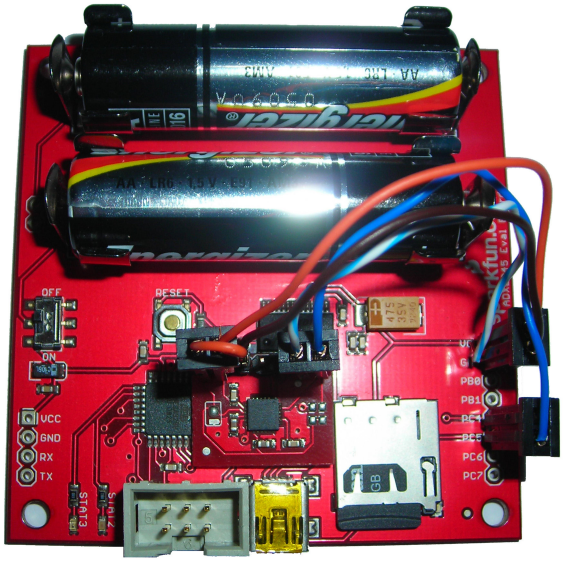

Figure 3.13: 6 degrees of freedom Inertial Measurement Unit (IMU) obtained by connecting the ADXL345 evaluation board and the ITG-3200 breakout board.

ters is lost after a power off or a reset operation, so permanent firmware modifications are necessary.

As an alternative to a programmer, we could use the Arduino bootloader installed on the microcontroller and load appropriate *sketches*, but with an external component this could result problematic and we chose to directly program the MCU and obtain a complete availability of the memory space and hardware resources. We bought the AVRISP mkII In-System Programmer by Atmel® Corporation, shown in figure 3.14.

Figure 3.14: The AVRISP mkII In-System Programmer by Atmel® Corporation.

The AVRISP communicates with the PC via USB and, combined with the AVR Studio software, it can program AVR 8-bit RISC microcontrollers (like ATmega328P) through their ISP interface. The latter is a Six-wire interface which connects the programmer (master) to the target board (slave); communications between these two components use the three SPI (Serial Peripheral Interface) wires Serial ClocK (SCK), Master In-Slave Out (MISO) and Master Out-Slave In (MOSI). At each pulse of clock provided by the master on the SCK line, one bit is transferred from the master to the slave through the MOSI line and one bit in the opposite direction through the MISO line. The RESET line is kept active (low) to enter and stay in Serial Programming mode and it is pulsed to perform a Chip Erase cycle. The other two wires of the interface are GND and VCC; GND must be connected to ground on the programmer and on the target board. VCC can be used in two different ways: the programmer can draw power from

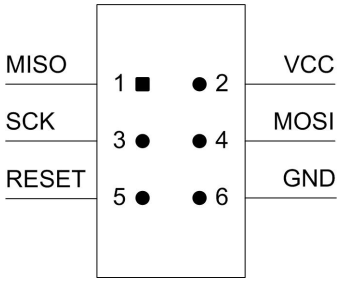

Figure 3.15: The ISP interface pinout (top view).

the target board running at any voltage from 2.7 V to 6.0 V or, alternatively, the target board can be supplied by the programmer for the duration of the programming cycle. This allows using only one power source.

The standard ISP connector is a 2×3 pin header contact, with pin spacing of 100 mils (1 mil is equal to one-thousandth of an inch); its pinout is shown in figure 3.15. In the ADXL345 the pin layout is exactly the same but, looking at the figure 3.1 or 3.13, the header results 90° counterclockwise rotated, so the MISO pin is the pin at the bottom left. In the cable of the AVRISP programmer, the MISO pin is indicated by a red wire.

Using AVR Studio, we wrote a suitable firmware for our IMUs in C language; this software is able to compile the C code in different assembly languages for many AVR platforms and then launch the burning process to the target processor through the AVRISP programmer.

We used the firmware provided with the evaluation board as a basis for the extension to the other sensor. First of all, we had to add ITG-

3200 definitions, containing registers names and addresses. After, since the ITG-3200 does not use SPI, we defined values and basic operations for I^2C protocol. An example of these instructions is shown in the following listing:

Listing 3.1: Some ITG-3200 and I^2C definitions

```
// ITG3200 Register Defines
#define WHO 0x00
...
#define TMP_H   0x1B
#define TMP_L   0x1C
#define Xrate_H    0x1D
...
#define Zrate_L    0x22
...
...
#define WRITE_sda() DDRC = DDRC | 0b00010000
...
void i2cSendByte(unsigned char data)
{
        delay_ms(1);
        printf("sending 0x%x\n", data);
        WRITE_sda();   // save data to the TWDR
        TWDR = data;   // begin send
        TWCR = (1<<TWINT)|(1<<TWEN);
}
```

Once the basic definitions were written, we had to properly develop reading and writing functions for the gyroscope; this step was not very simple, because it involved concurrent access to some microcontroller's registers and timing related problems. This resulted is many modifications also in reading and writing functions for the accelerometer.

When a new device is added, performing operations on it means modifying some ATmega328P's registers previously initialized for other devices, so every access is like a switch contest where pre-existing values must be saved and then restored after the operation. For example, registers DDRB, DDRC, DDRD and PORTC of ATmega328P have initialization values for the ADXL345 different from the values required by the ITG-3200 for performing a reading on gyroscopes output data through the I^2C interface. That means that before every reading on ITG-3200 these registers must be saved, initialized for the gyroscope and then restored with the old values. Listing 3.2 shows the core of the reading function for the ITG-3200: it is possible to see the reading of each half of the 16-bit data and the shifting of the higher half in a new variable. In the last part, after the printing of the original hexadecimal values, data are converted using the appropriate scale factor.

Listing 3.2: Reading operations on ITG-3200

```
...
//Angular rates on X axis
temp = ITG3200Read(Xrate_H);
```

```
xrot = temp << 8;
xrot |= ITG3200Read(Xrate_L);
...
//Temperature
temp = 0;
temp = ITG3200Read(TMP_H);
temperature = temp << 8;
temperature |= ITG3200Read(TMP_L);
...
//Printing and concatenation in a buffer
sprintf(ascii, "%04x", xrot);
strcat(buffer, ascii);
strcat(buffer, ",");
...
//Conversion
x = xrot/14.375;
...
tmp = (temperature + 13200)/280;
```

Readings on ADXL345 are performed in the same way; an example is given in the following listing, where it is possible to see the interrupt function, used only for the accelerometer. All these instructions are in a loop which polls the devices while an interruption from the user occurs. The first printing function prints the tick of the processor expressed in µs, useful to check time and frequency of the operations.

Listing 3.3: Reading operations on ADXL345

```
...
interrupt_source=adxl345_read(INT_SOURCE);
if((interrupt_source & DATA_READY)==DATA_READY)
{
        printf("us: %u\n", TCNT1);

        high_byte = adxl345_read(Xacc_H);
        low_byte = adxl345_read(Xacc_L);
        x_acc = (high_byte << 8) | low_byte;
    ...
    sprintf(ascii, "%04x", x_acc);
        strcat(buffer, ascii);
        strcat(buffer, ",");
    ...
```

Frequency and timing of reading operations are an important facet in the development of the firmware; readings on the two sensors, in theory, should be perfectly simultaneous and the period between two reading operations should be always the same, in order to minimize the error in the conversion techniques described in the next chapter. In practice, reading cannot be simultaneous on the two sensors because the processor is only one and must necessarily read from a sensor before and the period between two different readings can change due to external delays or hardware inaccuracies.

The first external delay is introduced by printing functions which print the output in a virtual terminal connected to the board via USB; the bit rate of the UART interface (default 57 600 bit/s) depends on the CPU's frequency and has a maximum of 691.2 kbit/s and 1.3824 Mbit/s respectively in normal and double rate mode at 11.0592 MHz, with an error rate of −7.8 % (at 8.0000 MHz, the default in the ADXL345 evaluation board, the maximum rates are 0.5 Mbit/s and 1.0 Mbit/s, with an error rate of 0.0 %). In light of these parameters, the best choice for a error free transmission is using the default CPU frequency; indeed the only possible higher selectable frequency (11.0592 MHz, see figure 3.4 and section 3.1.1 for details) allows error free transmission for a baud rate of only 230.4 kbit/s in double mode. The baud rate through the UART interface can constitute a bottle neck when printing statements occur together with locking operations like the reading from the accelerometer and the gyroscope: printing can require a time longer than the period between two readings and then alter the sample rate frequency. In order to avoid this problem CPU frequency and baud rate must be properly set or some printing statements should be removed from the code.

In the previous listings we saw the buffering of the output data from each sensor; this buffer resides in the ATmega328P's SRAM (Static Random Access Memory) which contains up to 2048 B. This hardware limitation requires downloading the buffer to the microSD card for avoiding memory data saturation: writing operations on microSD card can be too slow for

for complying with the sample rate frequency and introduce a problem similar to the one described before. In this case it is necessary to balance the size of the buffer between the time required by its downloading and the sample rate frequency.

Chapter 4

Sensors data processing

Once we have obtained a functioning IMU, we have to read and interpret data provided by sensors in order to obtain movements representative information. Contrary to what we might think, this is the most difficult step because both sensors and conversion procedures are affected by errors hard to eliminate completely.

4.1 Data coding

In the previous chapter we saw how the firmware of our boards reads surveys from accelerometer and gyroscope at different frequencies and saves them to a log file in the microSD card as hexadecimal strings or sends them to a virtual terminal in the same format. Each string represents

2-bytes data (six bytes for accelerations, six bytes for rotations, two bytes for temperature) plus an unsigned integer number (four bytes) for the CPU tick.

All these values are not ready to use at all; they need a transformation from their memory representation to readable acceleration and rotation values and then a processing and an interpretation to obtain the corresponding movement. Each sensor has its own internal coding: the specifications of this coding are essential for reading the data correctly.

As stated in section 3.2, the ADXL345 represents data using from 10 to 13 bits in two's complement: the number of bit used is called resolution and it defines different measurement ranges and different scale factors for each level; the default resolution level is 10 bits, with a measurement range of ±2 g a scale factor of 3.9 mg/LSB. Using the same resolution it is also possible to operate in the other ranges (±4 g, ±8 g and ±16 g) with the scale factors 7.8 mg/LSB, 15.6 mg/LSB and 31.2 mg/LSB respectively, resulting in a increasingly lower precision of the measurement. The best precision is achieved using the full resolution (13 bits) which allows reaching the maximum range with the finest scale factor; the corresponding coding is shown in table 4.1.

Also the output data of the ITG-3200 are 16-bit two's complement stored in 2 read-only 8 bit registers for each output. The table 4.2 shows the data coding for the angular rates; the coding for the temperature is similar but the scala factor is 280 LSB/°C and values are represented with an offset of

Table 4.1: AXDL345 full resolution data coding.

13 bit resolution, selectable ranges: ±4 g, ±8 g and ±16 g (scale factor 3.9 mg/LSB)

16 bit padding			**Sign**	**Absolute value**												**HEX**	**DEC**	**mg**	**g**
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF	4095	15970,5	15,9705
0	0	0	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF	2047	7983,3	7,9833
0	0	0	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03FF	1023	3989,7	3,9897
0	0	0	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	01FF	511	1992,9	1,9929
0	0	0	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	3,9	0,0039
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	-3,9	-0,0039
1	1	1	1	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	FE00	-512	-1996,8	-1,9968
1	1	1	1	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	FC00	-1024	-3993,6	-3,9936
1	1	1	1	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	F800	-2048	-7987,2	-7,9872
1	1	1	1	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000	-4096	-15974,4	-15,9744

13,200 (LSB).

4.2 From log file to usable data

The first step requires the transformation into decimal values (bearing in mind the representation in two's complement for negative values), the multiplication by a scale factor and, if necessary, the addition of an offset value. After this transformation we obtain acceleration values expressed in thousandths of g and rotation values expressed in °/s; the corresponding

Table 4.2: ITG-3200 data coding.

16 bit resolution, measurement range: ±2000°/s (scale factor 14.375 LSBs per °/s ≈ 0.069°/s per LSB)

Sign	Absolute value															HEX	DEC	°/s	rpm
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF	32767	2279,4435	379,9072
0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0001	1	0,0696	0,0116
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	0	0,0000	0,0000
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF	-1	-0,0696	-0,0116
1	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F000	-32768	-2279,5130	-379,91886

g value can be multiplied by 9,80665 in order to obtain an acceleration expressed in the standard unit m/s^2. Table 4.3 shows an example of "coarse" acceleration and rotation data for a single axis and their transformation into the correct unit of measurement. These values need now to be transformed into data which represent the movement. Before proceeding, it may be useful to briefly describe what is to be expected from the conversion process, also for detecting any malfunctions of the sensor.

The typical behavior of the acceleration values for a linear movement performed along a certain axis shows an initial positive change becoming negative during the pause (vice versa for first movement in the negative direction); velocity reaches a peak returning to zero when the sensor stops and the distance covered exhibits an increasing trend up to the point of arrival. In a perfect system, the acceleration's graphs would be a perfect sine wave and the velocity would be perfectly symmetric: in reality, this does not happen because the accelerometer is prone to ripples and noise

Table 4.3: From coarse data to acceleration and rotation values.

Coarse data		**Normalized values**		**Final values**		
Acc.	Rot.	Acc.	Rot.	g	m/s^2	°/s
000d	ffd5	13	-43	0.0507	0.4972	-2.9913
0014	ffcc	20	-52	0.0780	0.7649	-3.6174
ffeb	0003	-21	3	-0.0819	-0.8032	0.2087
fffd	000d	-3	13	-0.0117	-0.1147	0.9043
fff9	0018	-7	24	-0.0273	-0.2677	1.6696

which make the data fluctuate. These behaviors are shown figure 4.1 whose graphs depict acceleration, velocity change and consequent displacement; the example is referred to a linear movement of 5 cm along the X axes, sampled at 50 Hz for a total amount of 32 samples.

4.3 Conversion into movement data

The first graph of figure 4.1 can be obtained directly from the acceleration data (though actually some data filtering is required), whereas for the subsequent graphs conversion into movement data is required. The easiest way is to perform a double integral of acceleration data so as to obtain the velocity values from the first and the distance covered from the second, as

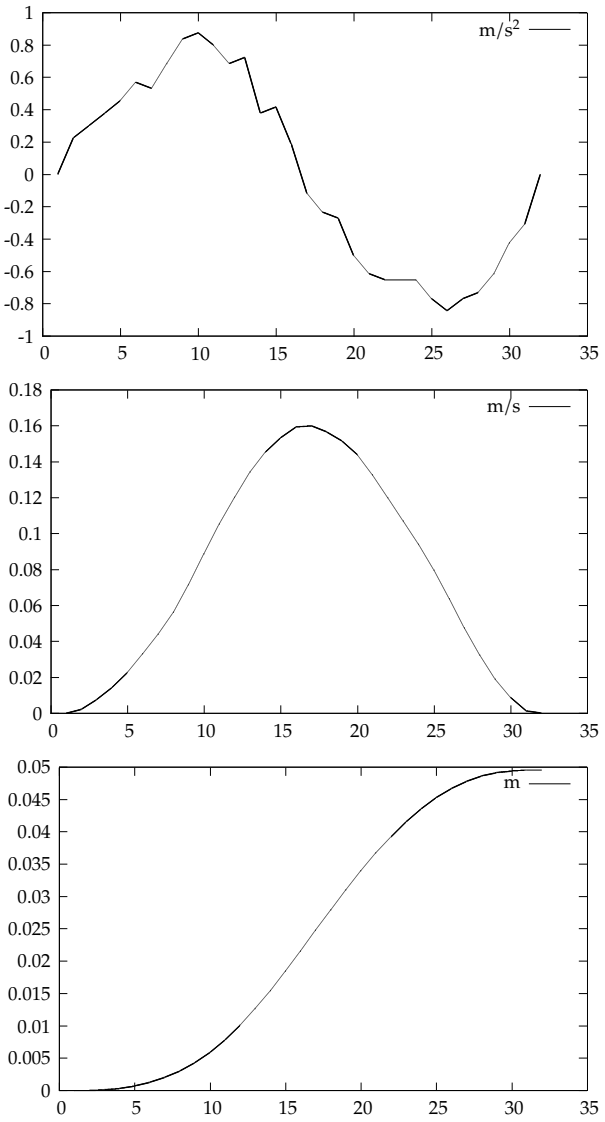

Figure 4.1: Typical graphs for acceleration, velocity and covered distance.

shown in the following formula:

$$v = \int (a)dt, \quad s = \int (v)dt \quad \rightarrow \quad s = \iint (a)dt \tag{4.1}$$

Since gyroscopes provides values that already represent a velocity (°/s), the procedure for rotation is the same but only one integration is sufficient to obtain the angle covered. Table 4.4 shows the integration procedure for a real acceleration: the data are a subset from the same linear movement (5 cm) depicted in the previous graphs; the changes described before can be seen.

4.4 Optimization

The basic conversion technique described before yields good results but it can be optimized for improving calculation precision: this kind of sensor is affected by typical problems such as inversion and error propagation.

In order to reduce the effects of error it is necessary, first of all, a calibration period in which the device stands in a certain position before starting any sensing; the data collected during this period of time are useful for calculating an average value which represents an offset to be subtracted from subsequent values. In other words, this procedure reduces floating values and calculates the actual zero measured by the standing device.

Table 4.4: Data integration: from acceleration to distance

Acceleration (m/s^2)	**Velocity** (m/s)	**Distance covered** (m)
0,2266	0,0023	0,0000
0,3031	0,0076	0,0001
0,3796	0,0144	0,0003
0,4561	0,0227	0,0007
...	...	...
0,4179	0,1534	0,0184
0,1884	0,1594	0,0216
-0,1176	0,1601	0,0248
-0,2323	0,1566	0,0279
...	...	...
-0,7295	0,0325	0,0486
-0,6148	0,0191	0,0492
-0,4236	0,0087	0,0494
-0,3088	0,0014	0,0495

A useful contribution to the optimization described above can be obtained with a fine setting of the threshold of the sensors, so that they can be less sensitive to vibrations, unimportant changes or other kinds of noise. The result can be reenforced by applying a low-pass filter to the data when, observing the calibration part of the log, there are floating values again.

An high level of optimization can be obtained using the Kalman filter; it is an efficient recursive filter which, starting from a set of measurements affected by random noise, estimates the values of the true measurements. The Kalman filter is applied to many real time systems (like GPS or objects

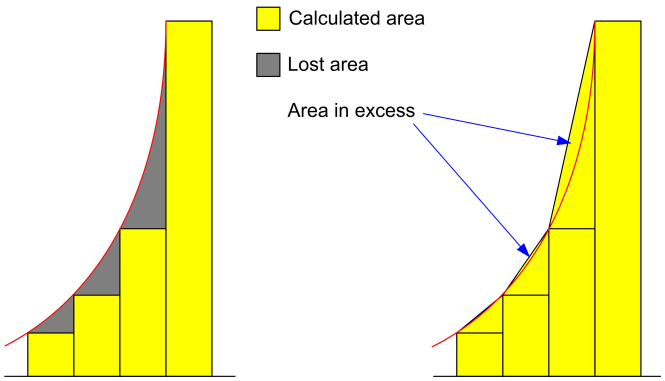

Figure 4.2: Errors in simple integration process and in first order interpolation.

tracking) and can be also applied to acceleration and rotation data; we implemented different versions of the Kalman filter both for processing a log file and for the real time processing of the data coming from our IMUs. The adjustment produced by the filter, in some cases, allows us to reach excellent results.

Even after fine optimizations, an important source of error is inherent the data integration described in the previous section. Since acceleration and rotation data are discrete, the integration procedure calculate areas smaller than the real areas: the lower the sampling frequency is, the bigger this sample loss is. The best way to solve (not completely) this problem is using a sample rate as high as possible and interpolating the real signal (the real acceleration or rotation curve); figure 4.2 shows the initial error and a first order interpolation using one triangle.

Table 4.5: Some results for linear movements and rotations

Real distance (m)	**Real angle** (°)	**Sampling frequency** (Hz)	**Calculated distance** (m)	**Calculated rotation** (°)
0,05	15	50	0.0495	15.6427
0,10	30	25	0.0966	32.1843
0,10	30	50	0.1092	31.2419
0,10	45	100	0.1087	45.6392
0,20	45	50	0.1971	41.0331
0,40	90	50	0.3921	92.7348
0,50	90	25	0.5196	93.8813
0,50	180	50	0.4976	184.3126
1,50	360	50	1.5128	366.4193

In light of the above considerations it is clear that the crucial issue is the precision achieved in measuring and representing the real movement. We tested several computation techniques and considered numerous device-dependent parameters, obtaining every time more or less close-to-reality results. Table 4.5 shows some significant results obtained for linear movements and rotations over a single plane: the importance of the sample rate can be seen.

In order to verify the correctness of the computation for linear movements, we simply measured the displacement of the movement performed. For complex movements, where error accumulation is greater, verification was performed by constraining bodies with bars on which angles of rotation and displacements were measured.

Chapter 5

Conclusion

For each issue discussed in the previous chapters, we will give some final considerations and development ideas.

5.1 Communication schema and security

At the beginning of our studies we proposed the tree structured model described in the first chapter for environmental survey applications. It was only an introduction to sensor networks (in this doctoral thesis too) before moving to the specific problem of wearable sensor networks.

The aim of the model is an efficient and safe communication schema in applications where nodes and data protection is required. The level subdivision introduced by the tree structure enables critical network areas

to be identified and minimizes the paths to the sink node but introduces geometrical constraints; this problem can be solved by introducing intra-level communications or multi-sink configurations and designating some groups of nodes as landmarks, in order to simplify pure recursive operations on the tree. Another important facet is tree maintenance when the number of nodes is very large or critical situations occur; to this end it may be necessary to develop a control protocol to quickly solve tree interruptions.

In chapter 3 we saw how the hardware resources of sensor nodes are limited: in the security issues proposed together with the model we chose to rely on hash algorithms and shared keys for message authentication and integrity complying with sensor node capabilities. We preferred to avoid the use of encryption algorithms which introduces a considerable data overhead. However, the use of cryptography (even if symmetric) could be useful for ensuring greater protection of some communications, for simplifying key redistribution and for ensuring data privacy when necessary.

5.2 Movements representation and comparison

In the second chapter we moved the focus to the core of our research and introduced wearable sensor networks; starting from movement data obtained by sensor nodes applied to the human body we proposed analysis

methods useful to represent and compare the movements.

Our method differs from others generally used in the approach adopted for recognizing, interpreting and evaluating a movement: we tried to reconstruct the actual trajectory of the whole movement instead of analyzing the output of sensors as waveforms in which a pattern is recognized or only final poses of the movement are identified. In this sense our approach is more accurate because evaluates the entire execution and not simply the relative positions of the limbs at the end.

We performed many tests and simulations of our analysis techniques obtaining optimum results for the use in a system for movements assessment and analysis. Compared to other kinds of approach, the sensor based analysis has the possibility of always performing objective and error-free evaluations: as explained in section 2.4, visual interaction alone can be restricted by the quality of the images, by the difficulties in evaluating them and capturing some body positions, that is by the lack of an objective measurement of movements. However visual interaction can reach a more realistic representation of the subject by reconstructing its surface: our focus was analysis and processing of data, which are themselves challenging problems, and proposed only a basic graphical representation of the subject.

5.3 IMUs

When we started our studies there were only digital tri-axis accelerometers commercially available but not gyroscopes in a single chip; we bought the ADXL345 evaluation boards and studied the characteristics of these devices and the main problems, such as inversion and the impossibility to detect the yaw angle (we experimented with a solution using two aligned accelerometers at a given distance).

After a few months the ITG-3200 was released, we bought the break-out boards and performed the integration with the evaluation boards as described in chapter 3. The IMUs we obtained accomplish the operations and the employments prompted but a full integration in a single board or in a single chip would be the best solution; nowadays it is possible to buy the IMU Digital Combo Board-6 Degrees of Freedom ITG3200/ADXL345 by SparkFun ^{TM} Electronics (figure 5.1) that integrates an ADXL345 accelerometer (left side of the photo) and an ITG-3200 gyroscopes in a tiny board (14 mm $\times$ 15.5 mm).

We anticipated the solution shown in the picture with our testing boards. Of course an integrated solution solves many problems related to protocols, timing and addressing (the IMU Digital Combo Board uses only the I^2C protocol) and allows the assembling of smaller IMUs. Again, from Sparkfun or other manufactures 9 degrees of freedom IMUs which embed magnetometers too are available.

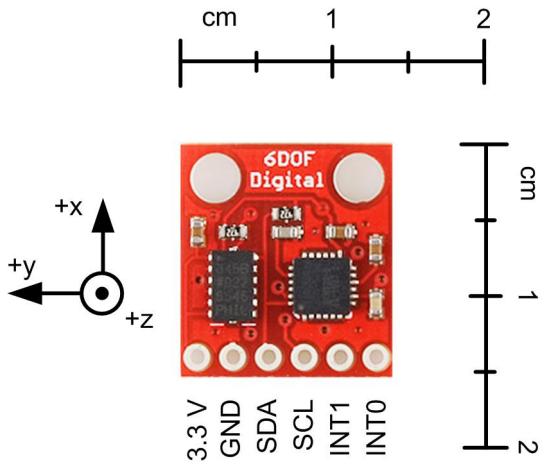

Figure 5.1: IMU Digital Combo Board by SparkFun.

An important improvement that could be done on our devices is the integration of a wireless module in order to eliminate cables for real-time analysis and introduce the possibility of having node-to-node communications.

5.4 Movements measurement and applications

In chapter 4 we proposed the details of movements measurement; a good result in this step is a fundamental prerequisite for applying movement data to analysis techniques described in chapter 2.

Using our IMUs we performed numerous tests and experiments to reconstruct the whole movement and to represent it graphically, to obtain an

immediate visual feedback and directly compare it to a reference movement represented in the same way. The full movement reconstruction is the crucial issue and represented the most difficult problem in our activities; obtaining a fine result is not easy at all and requires the control of many parameters. As presented before, we obtained a sufficiently good degree of precision for the use described here. However we observed that precision is greater for basic movements, i.e. movements performed along a single plane for limited duration.

The accuracy of results is an index for the applicability of hardware and technique to different fields. We prompted the use of our solutions in movements assessment systems for sports or healthy applications where the main purpose is to verify the correct execution of the movement as a whole without any major displacements, whereas minor differences in the positions are acceptable.

In the sports sector it is possible to use our techniques to evaluate athletic skills, for correction and preparation in training and for distance learning systems. Since devices are designed to store the data, these activities can be performed in real time with the (distance) control of a trainer/teacher or "off-line", after having downloaded the log file from the devices. This does not rule out the possibility for the athlete to view the results, using his own computer, obtaining immediate feedback on the execution of the exercises and improve on it.

The same techniques can be applied to a medical scenario where a

patient needs an objective evaluation on his improvements during, for example, a rehabilitation activity; in this way the patient is allowed self-exercising, doing away with the need for attending a physiotherapy/hospital center but every activity is stored in the device and can be analyzed by a specialist subsequently or with daily distance checks. This facilitates both patient and specialist activities and also provides a more reliable assessment and tracking methods of the rehabilitation protocol. Moreover, the important data sets obtained from the devices are useful for creating statistics and visualizing the results according to different parameters, so as to compare the activity performed with a standard protocol or with the patient's expected level of improvement.

The applications and the future developments proposed before have two common needs: improving computation precision for more complex movements and making the most of the hardware employed. That means eliminating, to the greatest possible extent, the errors and reaching the best hardware configuration. We are working on these aspects and cooperating with movement science researchers to validate our proposals in the fields previously mentioned.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[2] "ADXL345 datasheet," Analog Devices, Inc., p. 40, November 2010. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf

[3] "ATmega328P datasheet," Atmel® Corporation, p. 562, April 2010. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

[4] J. Barnes and R. Jafari, "Locomotion monitoring using body sensor networks," in *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. Athens, Greece: ACM, 2008, pp. 1–4.

[5] N. Bidargaddi, A. Sarela, L. Klingbeil, and M. Karunanithi, "Detecting walking activity in cardiac rehabilitation by using accelerometer," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, Melbourne, Australia, dec. 2007, pp. 555 –560.

[6] A. Bonfiglio, N. Carbonaro, C. Chuzel, D. Curone, G. Dudnik, F. Germagnoli, D. Hatherall, J. M. Koller, T. Lanier, G. Loriga, J. Luprano, G. Magenes, R. Paradiso, A. Tognetti, G. Voirin, and R. Waite, "Managing catastrophic events by wearable mobile systems," in *MobileResponse'07: Proceedings of the 1st international conference on Mobile information technology for emergency response*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 95–105.

[7] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 569–577, 2003.

[8] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 346–358, 2007.

[9] N. Crampton, K. Fox, H. Johnston, and A. Whitehead, "Dance, dance evolution: Accelerometer sensor networks as input to video games," in *HAVE 2007: IEEE International Workshop on Haptic Audio Visual*

Environments and their Applications. Ottawa, Canada: IEEE Computer Society, Oct. 12–14, 2007, pp. 1–6.

[10] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. Washington, DC, USA: ACM, 2002, pp. 41–47.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. Seattle, Washington, United States: ACM, 1999, pp. 263–270.

[12] G. Fenu and G. Steri, "Safe, fault tolerant and capture-resilient environmental parameters survey using WSNs," in *Security and Privacy in Mobile Information and Communication Systems*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan, J. Xiaohua, A. Zomaya, G. Coulson, A. U. Schmidt, and S. Lian, Eds. Springer Berlin Heidelberg, 2009, vol. 17, pp. 180–189, 10.1007/978-3-642-04434-2_16.

[13] ——, "Two methods for body parameter analysis using body sensor networks," in *Ultra Modern Telecommunications Workshops, 2009. ICUMT '09. International Conference on*, Saint Petersburg, Russia, Oct. 12–14, 2009, pp. 1–5.

[14] ——, "IMU based post-traumatic rehabilitation assessment," in *Applied Sciences in Biomedical and Communication Technologies, 2010. ISABEL 2010. 3nd International Symposium on*, Rome, Italy, Nov. 7–10, 2010, to be published.

[15] P. Gutmann, "Data remanence in semiconductor devices," in *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*. Washington, D.C.: USENIX Association, 2001, pp. 4–4.

[16] A. Hornung, S. Sar-Dessai, and L. Kobbelt, "Self-calibrating optical motion tracking for articulated bodies," in *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, Bonn, Germany, mar. 2005, pp. 75 –82.

[17] "ITG3200 datasheet," InvenSense, Inc., p. 39, March 2010. [Online]. Available: http://invensense.com/mems/gyro/documents/PS-ITG-3200-00-01.4.pdf

[18] B. Knorr, R. Hughes, D. Sherrill, J. Stein, M. Akay, and P. Bonato, "Quantitative measures of functional upper limb movement in persons after stroke," in *Conference Proceedings. 2nd International IEEE*

EMBS Conference on Neural Engineering, 2005. Arlington, Virginia, USA: IEEE, 2005, pp. 252–255.

[19] J. Macedo, C. Vangenot, W. Othman, N. Pelekis, E. Frentzos, B. Kuijpers, I. Ntoutsi, S. Spaccapietra, and Y. Theodoridis, "Trajectory data models," in *Mobility, Data Mining and Privacy*, F. Giannotti and D. Pedreschi, Eds. Springer Berlin Heidelberg, 2008, pp. 123–150, 10.1007/978-3-540-75177-9_6.

[20] N. P. Mahalik, *Sensor Networks and Configuration*. Springer, 2007.

[21] I. Manunza, A. Sulis, and A. Bonfiglio, "Organic semiconductor field effect transistors for unconventional applications: Flexible sensors and wearable devices," in *BSN '06: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 208–211.

[22] C. E. Perkins, *Ad hoc networking*. Addison-Wesley Professional, Dec. 2000.

[23] G. N. Pradhan and B. Prabhakaran, "Storage, retrieval, and communication of body sensor network data," in *MM '08: Proceeding of the 16th ACM international conference on Multimedia*. Vancouver, British Columbia, Canada: ACM, 2008, pp. 1161–1162.

[24] R. Ramachandran, L. Ramanna, H. Ghasemzadeh, G. Pradhan, R. Jafari, and B. Prabhakaran, "Body sensor networks to evaluate standing balance: interpreting muscular activities based on inertial sensors," in *HealthNet '08: Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*. Breckenridge, Colorado: ACM, 2008, pp. 1–6.

[25] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann, "Local and global skeleton fitting techniques for optical motion capture," in *CAPTECH '98: Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*. London, UK: Springer-Verlag, 1998, pp. 26–40.

[26] S. Skorobogatov, "Data remanence in flash memory devices," in *In CHES Workshop*, Edinburgh, UK, 2005, pp. 339–353.

[27] L. Song and D. Hatzinakos, "A cross-layer architecture of wireless sensor networks for target tracking," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 145–158, 2007.

[28] H. Tan, S. Jha, D. Ostry, J. Zic, and V. Sivaraman, "Secure multi-hop network programming with multiple one-way key chains," in *WiSec '08: Proceedings of the first ACM conference on Wireless network security*. Alexandria, VA, USA: ACM, 2008, pp. 183–193.

[29] Y.-C. Tseng, C.-H. Wu, F.-J. Wu, C.-F. Huang, C.-T. King, C.-Y. Lin, J.-P. Sheu, C.-Y. Chen, C.-Y. Lo, C.-W. Yang, and C.-W. Deng, "A wireless human motion capturing system for home rehabilitation," in *MDM '09: Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 359–360.

[30] A. Vehkaoja, M. Zakrzewski, J. Lekkala, S. Iyengar, R. Bajcsy, S. Glaser, S. Sastry, and R. Jafari, "A resource optimized physical movement monitoring scheme for environmental and on-body sensor networks," in *HealthNet '07: Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*. San Juan, Puerto Rico: ACM, 2007, pp. 64–66.