Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, INFORMATICA E DELLE TELECOMUNICAZIONI

Ciclo XXVIII

Settore Concorsuale di afferenza: 09/E3 - ELETTRONICA

Settore Scientifico disciplinare: ING-INF/01 – ELETTRONICA

POWER OPTIMIZATION FOR SENSOR HUBS IN BIOMEDICAL APPLICATIONS

Presentata da: **Filippo Casamassima**

Coordinatore Dottorato **Relatore**

Prof. Vanelli Coralli **Prof. Luca Benini**

Esame finale anno 2016

Dept. of Electrica, Electronic and Information Engineering (DEI)
University of Bologna

XXVIII Cycle

Power optimization for sensor hubs in Biomedical applications

Filippo Casamassima

PhD Coordinator
Prof. Alessandro Vanelli Coralli

PhD Supervisor
Prof. Luca Benini

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy

Abstract

The design and development of wearable inertial sensor systems for health monitoring has garnered a huge attention in the scientific community and the industry during the last years. Such platforms have a typical architecture and common building blocks to enable data collection, data processing and feedback restitution. In this thesis we analyze power optimization techniques that can be applied to such systems. When reducing power consumption in a wearable system, different trade-off have to be inevitably faced. We thus propose software techniques that span from well known duty cycling, frequency scaling, data compression to new paradigm such as radio triggering, heterogeneous multi-core and context aware power management.

Acknowledgements

I would like to express gratitude to:

- My supervisor Luca Benini
- My second supervisor Elisabetta Farella
- Bojan Milosevic, Simone Benatti, Alberto Ferrari and Prof. Lorenzo Chiari
- My family and friends

Dedication

To my family, my wife and my little monkey.

Contents

Chapter 1

Introduction

Wearable devices gained significant interest during the last few years. And they have been able to help people to better manage their health and their healthcare costs. Wearables, together with Body Sensor Network, which are an academic research field, are now widely adopted and accepted in population under various forms,: fitness devices, fashion gadgets etc.[KS15].

The main components of wearables and BANs are the so called ***"Sensor Hubs"*** that are low-power motion platforms that combines a low power micrcocontroller or DSP and high-accuracy sensors. The tight coupling between sensing unit and processing unit, enables the combination, or fusion of data coming from different sensors, to compute something more than could be determined by any one sensor alone. An example is computing the orientation of a device in three-dimensional space. That data might then be used to alter the perspective presented by a 3D GUI or game.

The body area network field is an interdisciplinary area that can allow inexpensive and continuous health monitoring with real-time updates of medical records through the Internet. A number of intelligent physiological sensors can be integrated into a wearable wireless body area network, which can be used for computer-assisted rehabilitation or early detection of medical conditions.

There are a number of challenges that must be faced to fully implement BAN including high

costs, package size and weight limitations, power efficiency and battery lifetime, memory storage, connectivity, ease of use, reliability, application level accuracy, security, and privacy issues. Since wearable sensor networks are battery-operated and may have critical and life-saving purposes, power efficiency is considered the most challenging design consideration in their real life deployment. In medical applications, wearable units are mainly used for remote and continuous patient monitoring, and therefore, their power consumption needs to be minimized to guarantee their long term operation and infrequent battery charge or replacement [GASS15].

In many cases studies do not address the battery lifetime issue directly, for scientific papers the length of related pilot studies is relatively short and they are often carried out in controlled environments. Tasks performed constantly such as sensing, processing (e.g., classification), and wireless transmissions incur significant energy expenditure in wearable nodes. As such, battery energy, if not properly managed, can be completely used up within hours and recharging is often impractical when the nodes are in-use.

In BANs, it is possible to distinguish among two operating modes that differently determine the mains source of power consumption:

- raw data is simply streamed to the gateway, the largest energy consumer is the radio subsystem (e.g., wearable ECG monitors), with the processing unit only required for formatting the data according to the utilized communications protocol.
- on-node processing (e.g., movement monitoring and wearable EEG monitors), the processing subsystem is the most energy consuming subsystem. In such systems, a signal with a lower bit rate will be transmitted to the gateway after processing.

The latter method necessitates further optimization of the computing units power consumption in order to prolong the lifetime of the entire system. This group of BANs often employ embedded signal processing and machine learning blocks that use sensor data (e.g., acceleration of body segments) to extract relevant information (e.g., types of movements) about their subjects.

Signal processing and machine learning methods are defined by the application and vary in complexity. Current techniques for wearable systems especially those concerning activity recog-

nition aim at using a reduced feature set to characterize the monitored signals in a real-time fashion while meeting wearable systems memory and processing constraints.

Statistical, time/frequency domain, and heuristic features comprise the reduced feature set at the end of the feature selection process. Feature extraction is often time consuming and can deplete the battery if an exhaustive feature set is extracted from the signal. Although classification accuracy is the ultimate measure of relative performance, for wearable platforms, there should be a mechanism to gauge the amount of useful information extracted for a given energy budget. Hence, real-life deployment of wearable platforms necessitates the incorporation of energy components into performance measures.

The traditional feature selection algorithms focus on specific criterion that finds redundancy and relevance in a given feature set. This approach is generally acceptable in conventional algorithms such as image processing and text mining, which run on highly powerful computers. These techniques, however, do not take into consideration computing complexity of individual features. That is, they allocate equal weights to features of varying complexity.

In wearable systems this approach is not effective, as these systems have limited processing power and need to operate in real-time. For instance, while computing peak-to-peak amplitude of a signal segment demands a linear mathematical function, calculation of Root-Mean-Square (RMS) power is quadratic, and a Fast Fourier Transform (FFT) requires ($nlogn$) operations to complete.

As the signal segment grows in length, complexity of these functions that need to run in real-time can result in unfeasible signal processing and machine learning algorithms. None of the feature selection techniques studied in the past takes into consideration the computing complexity of the selected features, an important measure in designing wearable monitoring systems. Furthermore, none of the existing power-aware schemes in embedded system design has dealt with feature selection algorithms and how the energy saving mechanisms can cleverly prevent some features from being accounted for in classification mechanisms.

1.1 Motivation and Objectives

As aforementioned, one potential application is in the form of Sensor Hub for measuring physiological/context parameters. The aim of Sensor Hubs based networks is to provide continuous monitoring of patients under their natural physiological states so that transient but life threatening abnormalities can be detected and predicted. In order to provide continuous monitoring, micropowered, miniaturised, and low cost wireless biosensors are required. To enable the development of wireless biosensors and BSN, a flexible and low power wireless development platform is required. Early BSN platforms are developed mainly based on modifying the Wireless Sensor Networks (WSN) platforms, and the development WSN has greatly facilitated the development of BSN platforms.

Nevertheless, as BANs are resource-constrained in terms of power, memory, communication rate and computational capability, many challenges still remain and must be tackled by researches, both in Academia and industry, among them we can highlight:

- Interoperability: WBAN systems would have to ensure seamless data transfer across standards such as Bluetooth, ZigBee etc. to promote information exchange, plug and play device interaction.

- Sensor nodes: The sensors used in WBAN would have to be low on complexity, small in form factor, light in weight, power efficient, easy to use and reconfigurable.

- Cost: Today's consumers expect low cost health monitoring solutions which provide high functionality. BAN implementations need to be cost optimized to be appealing.

- Constant monitoring: Users may require different levels of monitoring, for example those at risk of cardiac ischemia may want their WBANs to function constantly, while others at risk of falls may only need WBANs to monitor them while they are walking or moving. The level of monitoring influences the amount of energy required and the life cycle of the BAN before the energy source is depleted.

- Constrained deployment: The WBAN needs to be wearable, lightweight and non intrusive. It should not alter or encumber the user's daily activities. The technology should be transparent to the user i.e., it should perform its monitoring tasks without the user realizing it.

- Consistent performance: The performance of the WBAN should be consistent. Sensor measurements should be accurate and calibrated.

Currently the level of information provided and energy resources capable of powering the sensors are limiting. While the technology is still in its primitive stage it is being widely researched and once adopted, is expected to be a breakthrough invention in healthcare, leading to technologies like telemedicine and mobile Health becoming available not only in research environemnts.

1.2 Contributions and organization of the manuscript

In this work we present the development and evolution of a Body Area Network for Parkinsons's Disease rehabilitation. Simple sensor nodes, with basic data streaming functionality, evolved to the state of Sensor Hubs. This goal has been achieved trough different steps, and improvements, both hardware, and software. The work starts with the description of the Cupid Project for which sensor nodes have been developed with the aim to collect patient's walking characteristic. The chapter 2 shows how we were able to develop and implement algorithms on the sensor nodes to assist PD and more in general elder people in gait rehabilitation.
The work continues with chapter 3, where we describe an analysis performed on the sensor nodes to find the best power management policy, each low power mode is indeed characterized by a wakeup time and power consumption. Moreover components behaves and interact differently according to the power state, for this reason the power state of the system (which is different from low power behavior of the single components) must be taken in account. In Chapter 3 we also describe another firmware improvement on the sensor nodes, that is essential in a BAN: the possibility to have a common time reference for data collected by different devices, we adopt a novel synchronization method based on Bluetooth piconet clock, that does

not require additional hardware or access to Bluetooth stack.

Chapter 4 Is an extension of the Cupid Project, we were able to use a market available device, to perform upper limbs and handwriting rehabilitation. In this chapter the hardware modification to the consumers device and software strategies for handwriting assessment and feedback restitution are described.

In Chapter 5 we propose the implementation of a power aware Real Time OS,this has been achieved through the development of a middle-ware for an existing RTOS. The Energy Broker Middleware is able to detect the available battery energy and configure the system, according to it, so to consume more energy and deliver better accuracy only when such energy is available The work continues with Chapter 6 where we introduce a novel concept: the Context Aware Power Management, that fully qualifies a standard sensor node into a more evolved sensor hub. The sensor hub is able to detect user activity, collecting inertial sensor data, and manage the various device components to apply the most convenient power management policy, dynamically. We analyze different activity detection algorithms to find the one that has lesser impact on battery consumption. Different improvement have been made to further reduce the energy needed to detect the context.

On Chapter 7 we further extend the CAPM concept to a network of Sensor Hubs, adding a piece of hardware, called Wake Up Radio (WUR). The is a radio receiver whose current consumption in receive mode is 5 order of magnitude lower than other radios used in BANs (e.g. Bluetooth). The network is thus constituted by an intelligent sensor hub that detect the user's activity and can wake up the peripheral nodes, only when necessary, allowing a great improvement on battery life of peripheral nodes, and minimal impact on the central node.

In the last Chapter 8 of this work we bring another hardware improvement to the Sensor Hub power consumption: using a dual core architecture, we are able not only to achieve less computational time, but to reduce energy consumption, thanks to the heterogeneity of the dual core platform. The presence of two different cores, allows to use each core for a different task, if proper task assignment is performed power savings can be significant. The fact that the dual core platform shares a number of resources, further reduces energy demand if the two cores are able to perform their tasks in parallel.

Chapter 2

Body area network for PD rehabilitation

In this chapter a novel approach for Parkinson's Disease (PD) Patients is described. The approach consists in the use of wearable sensors, that together constitute a Body Area Network (BAN) for personalized at home support in rehabilitation.

2.1 The Cupid Project - Overview

A major focus for the rehabilitation of several severe conditions is the restoration of walking capabilities, as this ability is directly linked with a recovery of independence [APH13]. Rehabilitation for gait difficulties is needed in different situations, such as to recovery the patient's capabilities after orthopedic surgery [CWS08], during post-stroke therapy [LMM+04], or to prevent gait impairments arising from progressive degeneration of the central nervous system in Parkinson's disease (PD) patients [YSGBH12]. In all this cases gait difficulties cause a strong impact on the patient's quality of life and the patient is prescribed exercise therapy to train and improve the gait performance.

The advantage of training exercises is that they can be autonomously performed by the patient at home, but with this approach the clinician can not evaluate the quality of the performed

exercises. Poor exercise technique can often result in poor outcomes for these patients and delay their return to full function. During training sessions at the clinic, patients are motivated by the clinician, who can assess the correct execution of the desired exercise and give a timely feedback to the patient [NRMS09]. Even in this case, a measurement and an objective evaluation of the exercise are difficult and require complex and expensive instrumentation (e.g multi-camera motion tracking systems or force platforms). Moreover the existing measurement solutions have a very limited range, confined to a short path in the laboratory [LGR13].

Recent research has shown that external cues in the form of visual [VHZ+12], audio [PMFC11] or haptic [RCM+13] feedback can help in movement planning and execution. The wearable system we propose in this work can evaluate and assist the user's gait performance during training exercises in a unconstrained daily life scenario. We expect that an approach based on accurate and real-time gait analysis, linked with a closed-loop feedback for knowledge-of-performance, may overcome the current limitations and increase treatment efficacy. Moreover, the possibility for the patient to use such a system in an unobtrusive way during daily life exercises, will increase the potential of the therapy.

The contribution of the CuPid Project, described in this chapter can be summarized as follows. We present a system for gait rehabilitation for use in a daily life setting, developed in the context of the EU-funded project *CuPiD*[1]. The proposed system employs a fully portable and wearable architecture formed by a wireless body sensor network, based on inertial measurement units (IMUs), connected with a smartphone used as a portable processing platform. The therapeutic hypothesis is centered on the concept of motor learning and is based on intensive repetitions of a motor task.

The system performs a real-time computation of gait spatio-temporal parameters, which are compared against the patient's reference walk. The results and indications on how to improve the training are returned to the subject by means of vocal messages, with the aim of helping the patient to perform the correct movement pattern, stimulating the motor learning process. The messages are returned to the user by the automatic and stand-alone system and they are inspired by the type of vocal instructions given in traditional rehabilitation contexts by clinical

[1]Closed loop system for personalized and at home rehabilitation of Parkinson's disease

Gait Features	Content of the Instruction to the User
Cadence (steps/min)	increase/decrease gait speed
Step	length keep steps longer/shorter
Gait	speed increase/decrease gait speed
Gait	asymmetry increase right/left step length
Trunk	flexion keep upright posture
Clearance	rise right/left leg

Table 2.1: Gait features and the corresponding instruction to be fed back to the user in real time and in a closed loop by the wearable sensors and system.

operators.

2.1.1 System Concept and Design

The system concept and design principles were built into a patient-specific framework, with the aim of dealing with specific scientific and technological challenges, including:

- Automatic and real-time computation of gait features from an easy-to-use system set-up;
- The definition of rules for real-time feedback restitution;
- Identification and optimization of the type of feedback (reward/negative; continuous/discrete).

The systems use was planned to be very easy, possibly exploitable in a domestic setting and based on a very simple sensors set-up. The information returned to the user during training aims at reflecting the nature of instructions normally provided to the patient by the therapist during traditional rehabilitation trials, potentially with higher precision and reliability than human instructions. Following clinicians specific experience, a literature review and feasibility criteria [FFMK11], the gait features listed in Table 2.1 were defined as key characteristics, especially for PD, and as good candidates as variables to be trained.

In summary, the system and its training possibilities are designed to allow the functional goals of improving the rhythm and amplitude of the gait, increasing vigorousness, decreasing the asymmetry of movements and preventing shuffling and forward inclination. Spatio-temporal

gait parameters, such as cadence, gait speed, etc., provide the information needed to monitor the performance of a subjects walk and an accurate measure of the overall efficacy of the locomotor function. In addition, beside the computation of typical gait features related to the lower limbs, the system was intended to monitor any abnormal forward inclination of the trunk. This is a strong disabling symptom in some PD patients with important effects on gait efficacy [TDMM12].

In the scenario of use, the subject is trained to adopt and maintain a correct trunk posture and correct gait patterns, during short periods of the day, via real-time detection and subsequent feedback restitution of his/her trunk kinematics and gait performance. In this way, patients can, at least partially, re-learn the correct gait pattern and trunk posture or develop strategies to overcome their disability [RBH+10]. To be able to compute the aforementioned features,

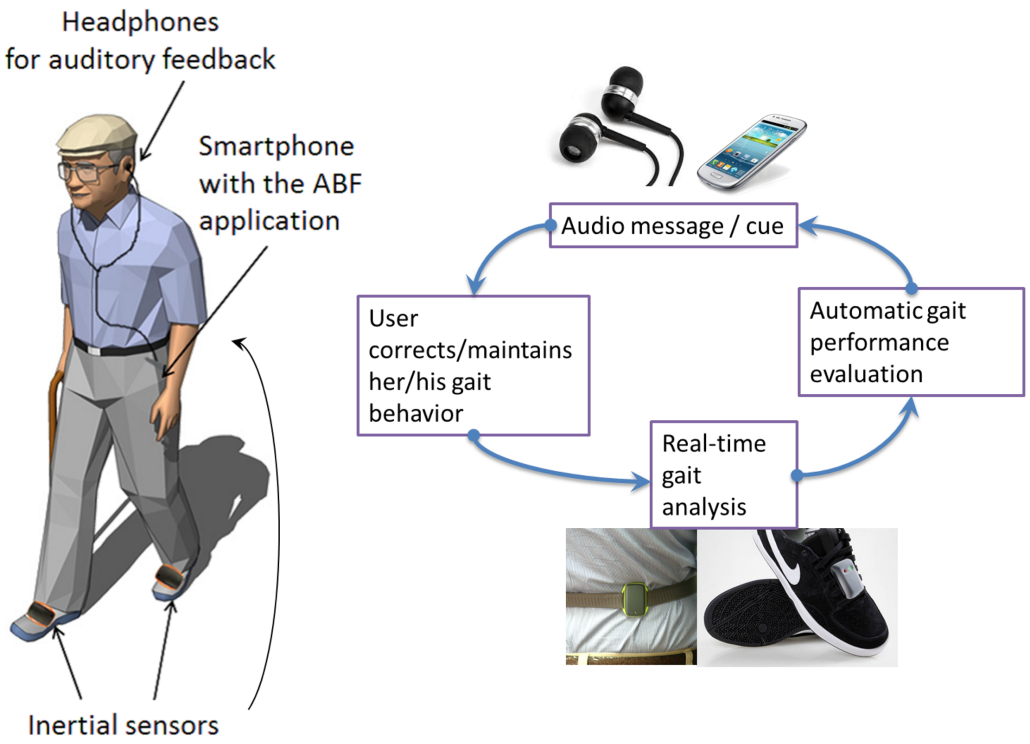


Figure 2.1: Scheme of the scenario for the use and the components of the system

minimizing the number of sensors, but maximizing the reliability, wearability and flexibility of the system, a set-up based on three sensors and one processing unit (a smartphone) has been selected. Two sensors are mounted on the shoes and a third one is mounted on the trunk (Figure 2.1). The feedback is provided using vocal messages delivered via headphones.

The therapeutic plan may follow a progressive adaptation and modification of the combinations of variables monitored by the system. Specifically, as soon as a specific gait feature has been trained to a certain extent, the biofeedback system will be tuned to focus on a different feature.

The training scenario at the basis of this design includes unconstrained gait (with possible stops, turning, road-crossing, etc.) to be performed in combination with the biofeedback system. The long-term goal is to train people to maintain their own optimal gait for the entire duration of the trial (possibly lasting around 30 min). The system presented, and specifically developed, in this study is essentially composed of three elements:

1. The inertial sensor nodes;
2. An Android mobile phone;
3. The intelligence on-board able to compute gait features and to propose real-time feedback.

Hardware (and related firmware) components will be presented in the next section, while the application and the related algorithms will be introduced in the following one.

2.1.2 System Architecture

In the present study, a smart sensor platform based on IMUs and specifically designed in the context of the EU-funded project, CuPiD, was customized and used for the first time. The sensing unit, EXLs1 (by EXEL, Bologna, Italy) (Figure 2.2) is an open and versatile alternative to available commercial solutions.

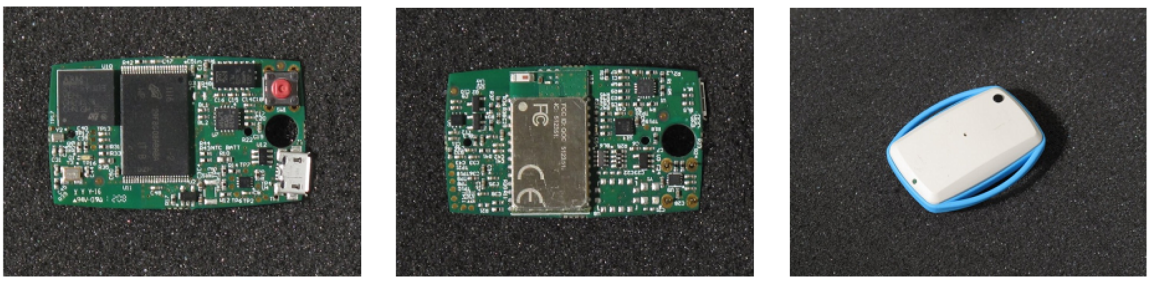


Figure 2.2: The EXLs1 sensing unit.

Each node includes an STM32F103 microcontroller, based on the ARM Cortex-M3 core running up to 72 MHz. A combination of Micro-Electromechanical Systems (MEMS) sensors produced by STMicroelectronics (accelerometer, gyroscope and magnetometer) is embedded in the sensor nodes, while communication with external devices is performed with a standard Bluetooth module (STM SPBT2532C2.AT). The hardware configuration is completed by a 1 Gb NAND FLASH memory for local data storage and a Li-ion battery. The sensor nodes have a standard micro-USB port, which is used to recharge the battery. This can be easily performed with a standard smartphone charger, compatible with all Android phones, or with a docking station provided by the manufacturer.

The firmware, specifically implemented for the present study, was designed to optimize the hardware components and for allowing the sensing units to work in different modes: data streaming via Bluetooth; data logging in the internal FLASH memory; with an on-board processing if necessary.

2.1.3 Sensing Nodes Calibration and On-Board Orientation Estimate

For the correct use of sensor data, a rapid and reliable calibration procedure was defined. Even if the different sensors were factory calibrated, an additional calibration step was required to compensate for possible misalignment during their assemblage into the final device. Calibration methods for inertial and magnetic sensors are well documented [MNF+12]. For the calibration procedure, six static poses and rotations of the device around its three main axes are considered, and the data is processed to obtain a calibration matrix and an offset vector for each sensor.

To evaluate the performance of the sensor node, we compared it with a commercial high-end solution, such as the MTw sensor platform (by Xsens, Enschede, Nedherlands). For this purpose, we collected raw data from the two platforms in a static condition for 10 s, which was used to compute the sensors noise in terms of the standard deviation around the mean values. The results are in Table 2.2, and they show a very similar performance for the two platforms,

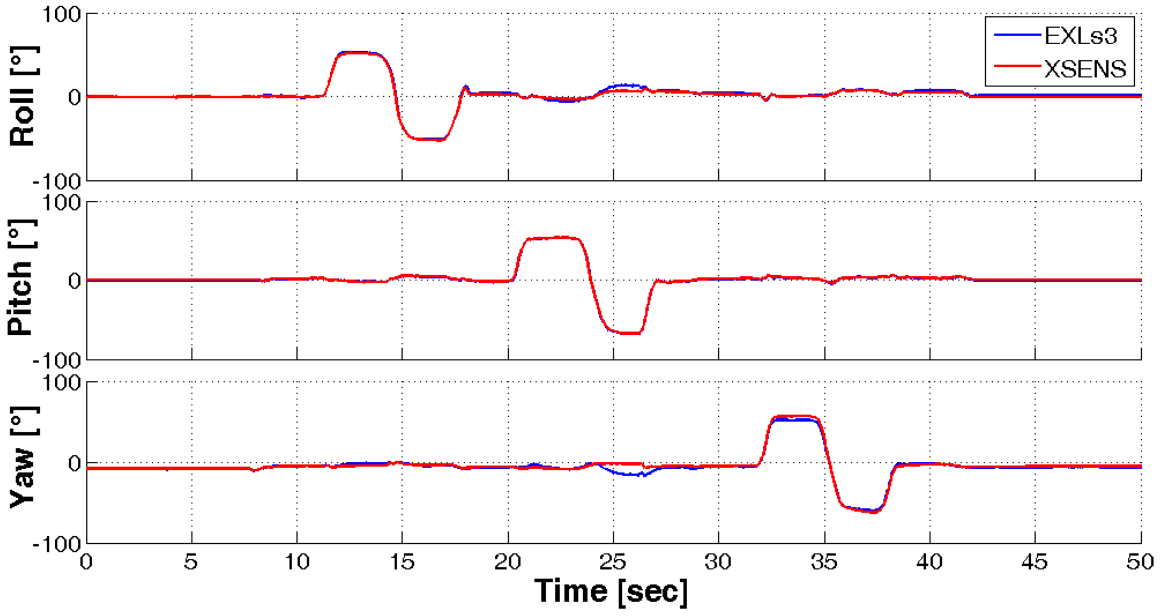
	EXLs1			MTw		
	X	Y	Z	X	Y	Z
Accelerometer	0.0034	0.0019	0.0204	0.0156	0.0126	0.0325
Gyroscope	0.0103	0.0034	0.0047	0.0049	0.0055	0.0054
Magnetometer	0.0079	0.0077	0.0220	0.0014	0.0013	0.0023

Table 2.2: Static noise estimation for the different sensors of the EXEL and Xsens sensing nodes.

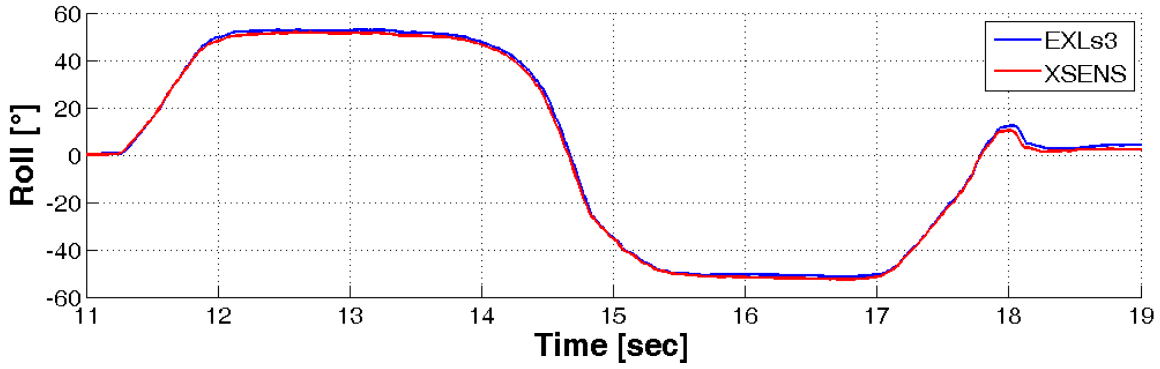
reporting the standard deviations of each axis for the three sensors.

IMU sensor data can be used to estimate the orientation of the device, providing the implicit orientation of the body segment on which they are mounted. Being one of the most common types of information to be extracted from an inertial body sensor network, the orientation estimation is calculated directly on-board the devices. To avoid the creation of a communication and computation bottleneck on the host device, each employed node can compute its own orientation by exploiting the embedded microcontroller, and achieve better energy efficiency avoiding to send all the sampled data to the host device. State-of-the-art algorithms for the estimation of orientation from IMU sensor data were optimized for the EXLs1 sensor node, including the Kalman filter [Kal60], its extended variation and recently developed complementary filters [MHP08]. These different algorithms were implemented and are available on the EXLs1 sensor platform, where they were directly compared. All the algorithms were optimized to run in real time on the resource-constrained embedded MCU, and their performance was compared in terms of accuracy, computational cost and energy efficiency. The Kalman Filter approach resulted in being the most flexible one and was therefore the preferred choice for our application. The precision achieved on the MCU was tested against the MATLAB implementation, which uses 64-bit double precision values and was computed on a PC. For this purpose, both the sensor data and the computed quaternions were sent from the device to the PC, to allow for the elaboration of the same stream of data on the two platforms. The difference between the orientation computed on the two platforms is negligible, having an RMSE lower than 0.004 degrees for roll, pitch and yaw angles. A further sensor validation check was performed in terms of orientation estimation, comparing the two sensing platforms. In this case, two nodes were attached to each other and rotated in air, while logging the computed orientation from the

systems. The result of 50 s of computed data is shown in Figure 2.3. The difference between the two estimations is minimal, even if the two nodes were manually aligned, with an RMSE of 1.818, 0.594 and 2.941 degrees for the three estimated angles.



(a)



(b)

Figure 2.3: (a) Comparison of orientation estimation as computed by Xsens MTw (red) and EXLs1 (blue) sensor nodes using the Kalman filter. (b) Zoom of the roll plot.

2.2 Algorithms and Real-Time Feedback for Gait Rehabilitation

Overall, the functional goal of the system entails the improvement of the gaits rhythm and amplitude, the reduction of asymmetry and the correction of unstable posture assumption by means of recurrent audio feedback provided in real time during unconstrained walking. Feedback restitution is performed by a logic flow of states and conditions able to tutor the patient in maintaining the gait pattern and performance at a specific target set by a clinician. Furthermore, the frequency and total amount of messages provided to the patient are adaptively controlled, to avoid saturation effects. Besides, the degree of difficulty of the task is automatically tuned to be challenging, but not too demanding. Finally, vocal messages, encoded by a text-to-speech application, have been formulated from clinicians to stimulate those motor adjustments usually prompted during rehabilitation sessions in clinical environments. In order to implement these functions, an Android mobile phone (equipped with a Bluetooth connection capable to run Android 2.3) has been adopted to manage the network, send commands and receive data from the sensor nodes. The intelligence on-board the system encompasses two main functions:

- soft real-time automatic identification of gait parameters;
- soft real-time feedback restitution to the user.

The estimation of gait spatio-temporal parameters concerns, first of all, the segmentation of walking into gait cycles, which entails the detection of the events, initial contact (IC) and foot-off (FO). A new online automatic algorithm was developed to detect in real time the IC and FO by processing the angular velocity along the medio-lateral axis of the IMUs, which are rigidly attached to the patients shoes. Once these events are detected, it is possible to determine the gait spatio-temporal parameters, such as cadence, gait asymmetry or step length by means of ad hoc signal filtering techniques (see Sections 4.1 and 4.2).

Based on clinical practice and feasibility criteria, the following gait parameters were identified as the most clinically meaningful, namely: cadence, step length, trunk posture, gait speed, gait asymmetry and clearance. The real-time computation of these parameters was implemented into the application.

2.2.1 Soft Real-Time Gait Events Detection

For the automatic detection of gait events using inertial sensors, several approaches have been proposed in [PD+92, SBV+13]. They differ in terms of sensor locations and the processing of accelerations and/or angular velocities to estimate IC and FO events.

In this context, it was decided to keep the number of IMUs as limited as possible in order to: (1) ergonomically improve the ease of use, decreasing the required time to attach sensors on body segments and reducing the hindrance to natural walking; and (2) technically reduce the power and memory requirements of the central processing unit.

Among all the different methods proposed in the literature for detecting IC and FO, the one from Lee et al. [LP11] has been used as the starting point to implement a novel method conceived specifically to deal with our application scenario. This new algorithm is able to determine IC and FO by processing the angular velocity signal arising along the medio-lateral axis of the foot. In particular, during each single gait cycle, the foot rotates alternatively clockwise and anticlockwise around the ankle joint, hence allowing the possibility of segmenting the gait cycle in its different temporal phases. The implemented algorithm first identifies all positive peaks (feet in anticlockwise rotations by looking at a patient walking from his/her right side) associated with mid-swing events [PD+92], then, within each couple of mid-swing peaks, finds out the first negative peak (feet clockwise rotations), being the IC, and the second (in time) negative peak, being the FO (see Figure 2.4).

The identification of the positive and negative peaks is based on the MATLAB built-in function, findpeaks(). Gait speed and pattern and the presence of flat or uneven terrain produce peaks differing in terms of amplitude and sharpness. In order not to lose sensitivity and specificity

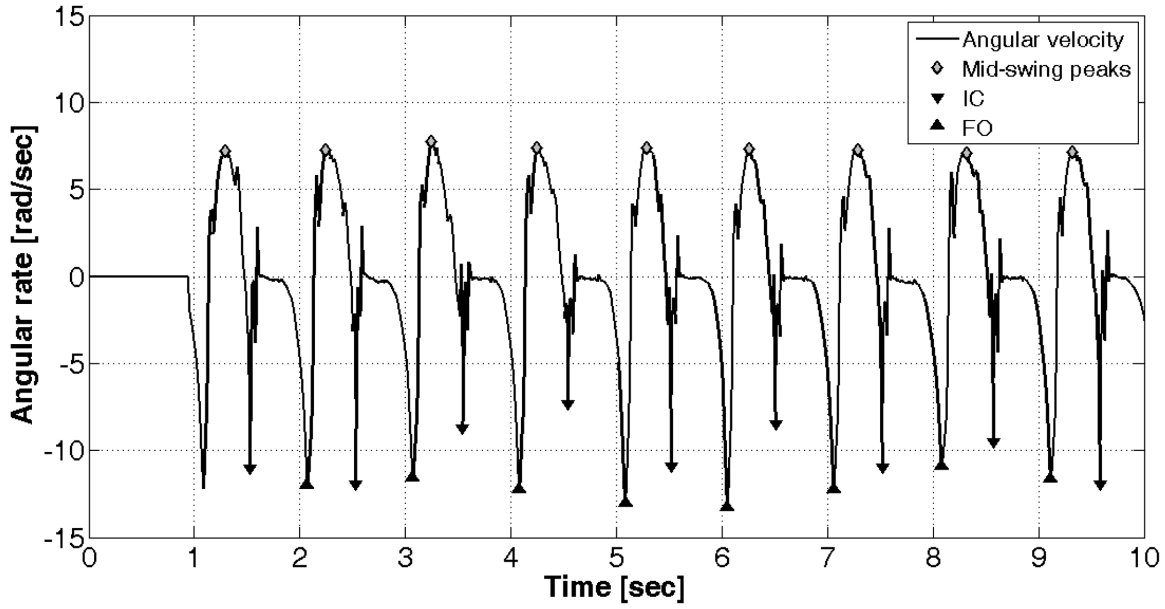


Figure 2.4: Angular velocity and the detected initial contact (IC) and foot-off (FO) events.

in IC and FO identification, the internal parameters of the findpeaks function (minimum peak height, minimum peak separation in time, minimum height difference and number of peaks) are automatically tuned from the algorithm on the basis of a dataset of healthy subjects and patients walking at different speeds (ranging from 0.4 m/s to 1.8 m/s), with different patterns outdoor and indoor (see Section 4.4 for the accuracy of the method). From the knowledge of the IC and FO instants, it is then possible to easily define cadence, duration of stride, stance, swing phases and all other gait temporal parameters.

2.2.2 Soft Real-Time Step Length Estimation

Pedestrian dead-(for deduced) reckoning allows one to estimate the feet positions in real time from the use of two sensors attached to the feet. Dead-reckoning is the process of calculating ones current position by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time and the course. Theoretically, considering the accelerometer as a linear motion sensor and the gyroscope as a rotation sensor, by double integration of the acceleration of an IMU (e.g., attached to the shoe), it is possible to track the position in time of a subjects walking in any environment. However, in practice, this is an unfeasible operation, since the orientation-dependent gravitational component is not

straightforwardly separable from the inertial acceleration, and even minor drift internal to the gyroscope or the accelerometer (being single-/double-integrated) causes errors that increase cubically.

Pedestrian dead-reckoning allows continuous computing of the position, orientation and velocity of a subjects motion by implementing additional filtering based on human walking constrains. When a person walks, his/her feet are periodically in a stance-stationary phase in which the entire foot is in contact with the floor and the body shifts forward by rigidly pivoting the lower limb on the ankle joint (the second rocker phase of the gait cycle). During this stance stationary phase, it is possible to draw an advantage from the condition of having the foot at zero velocity, hence filtering errors accumulating in the integration of acceleration and correcting the position estimated up to the second rocker phase. In other words, zero-velocity updates (ZUPTs) act as pseudo-measurements, allowing for the reduction and limitation of position errors. The pedestrian dead-reckoning algorithm implemented in this study was developed starting from the method proposed by Nilsson et al. [ISH12].

A range of detectors aimed at identifying when the IMU is stationary, so that the ZUPT can be applied, have been proposed in [SNH10]. These detectors are all generalized likelihood ratio tests; they differ in the assumptions of prior knowledge about signals dynamics, under the hypothesis that the IMU is stationary. As shown in [SNH10], the methodological differences among different ZUPT interval detectors may lead to double the errors in position estimation. The angular rate energy (ARE) detector [SNH10] is computationally one of the lightest and provides the highest position accuracies; and, it is one of the most robust to changes in gait speed.

Based on these characteristics, ARE was selected as the ZUPT interval detector to be used in the system. In addition to the statistical analysis of the angular rate signal, an innovative module exploiting IC and FO time and amplitude knowledge was introduced in order to improve the specificity of the identification of stationary states. In particular, within each interval lasting from an FO to following IC, it was imposed to not have ZUPT occurrence. Furthermore, the standard deviation of gyroscope noise used to control the ARE trust in the gyroscope data,

as well as the threshold and the window size used to determine the ZUPT occurrence are automatically tuned by the algorithm on the basis of the IC and FO amplitudes.

As a real-time application, gait events, ZUPT interval identification and signal filtering are executed in a continuous way as soon as new measurements are available, allowing the tracking of movements in real time (see Section 2.2.4 for the method accuracy).

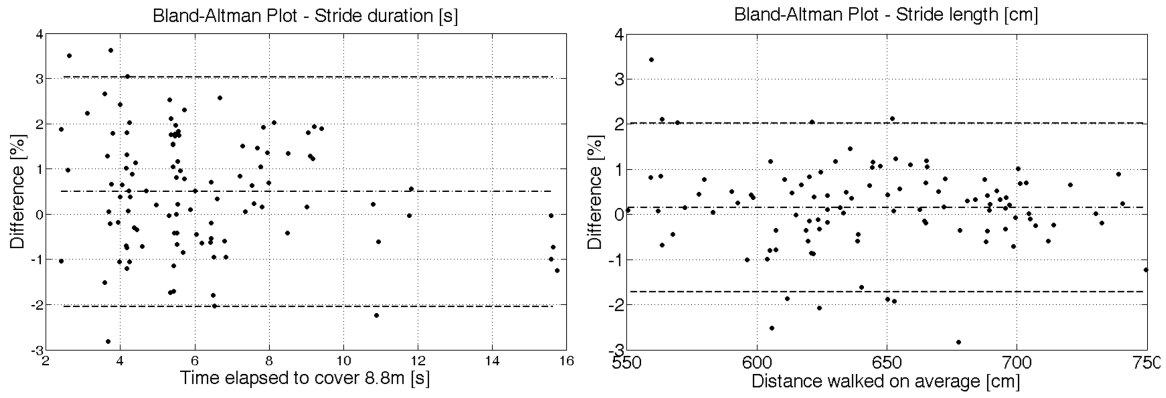
2.2.3 Trunk Flexion and Gait Asymmetry Estimation

Gait asymmetry was obtained by applying the equation provided in [YPP+07]. Trunk flexion was simply obtained as the inverse of the sinus of the anteroposterior component of the acceleration signal, provided from the sensor attached to the trunk. Similarly to other gait parameters, trunk inclination is monitored during the walk, and a vocal request of restoring an upright posture is provided every time the patient bends forward (see Section 2.2.4).

2.2.4 Step Length Estimation Accuracy in Clinical Settings

In order to test the step length, five male PD patients walked five times at comfortable, twice at increased and twice at decreased gait speed over an 8.80-meter instrumented GAITRite walkway (GR, CIR Systems Inc, PA) set at 100 Hz. The patients were aged 5,087 years with Hoehn and Yahr stages 13. Hoehn and Yahr is a clinical rating scale for disease severity in Parkinson, ranging from zero (no signs of the disease) to five (bedridden). The patients were tested on medication, taken in their usual dosage. The IMUs were set at 200 Hz and fastened on top of the participants shoes; the same two IMUs with the same Kalman filter settings were used for all patients. IMU data collection was performed via Bluetooth by means of the Android application. For each of the nine trials, the total duration and length of the strides was obtained both from IMUs and GR. Figure 6 shows, in Bland-Altman plots, the differences in the percentage between step duration and length estimated with IMUs and GR.

The results obtained show a good agreement between the INSand GR method, with a bias close to zero and an SD within 3% for stride duration and 2% for stride length. The performance



(a) (b)

Figure 2.5: BlandAltman plots for percentage differences for (a) the step duration and (b) the step length estimated with inertial measurement units (IMUs) and GAITRite (GR).

obtained among the five subjects and the two sides (that is, two IMUs) is consistent, revealing a good inter-subject and inter-hardware reliability. A single side of one patient produced accelerations above the full scale of the IMUs (6 g), revealing the requirement of setting the acceleration full scale higher than 6 g.

In conclusion, ZUPT-aided INS by means of foot-mounted IMUs is robust and accurate enough to estimate step length over intervals of a mid-range distance on patients with PD.

subsectionReal-Time Audio Feedback Application In order to monitor and guide the patient performance during the gait, the system computes in real time the key gait parameters and relates them with respect to a reference, providing a vocal request for changing the gait pattern every time the match is not satisfying or a vocal reinforcement every time it is.

The system architecture is schematized in Figure 2.6. This architecture has been implemented into an Android standalone application, which processes data streamed from the sensor nodes via Bluetooth. It was tested on different dual core smartphones, always achieving real-time performance and running with no delay. The application was evaluated on an LG Nexus 5 phone, where the time needed to perform the algorithm was measured. The operations executed at each new sample and at each new step detected represent the main computational load of the algorithm, and they are executed in 0.11 ms and 0.23 ms, respectively. This computational load guarantees the real-time execution of the application with no delays.

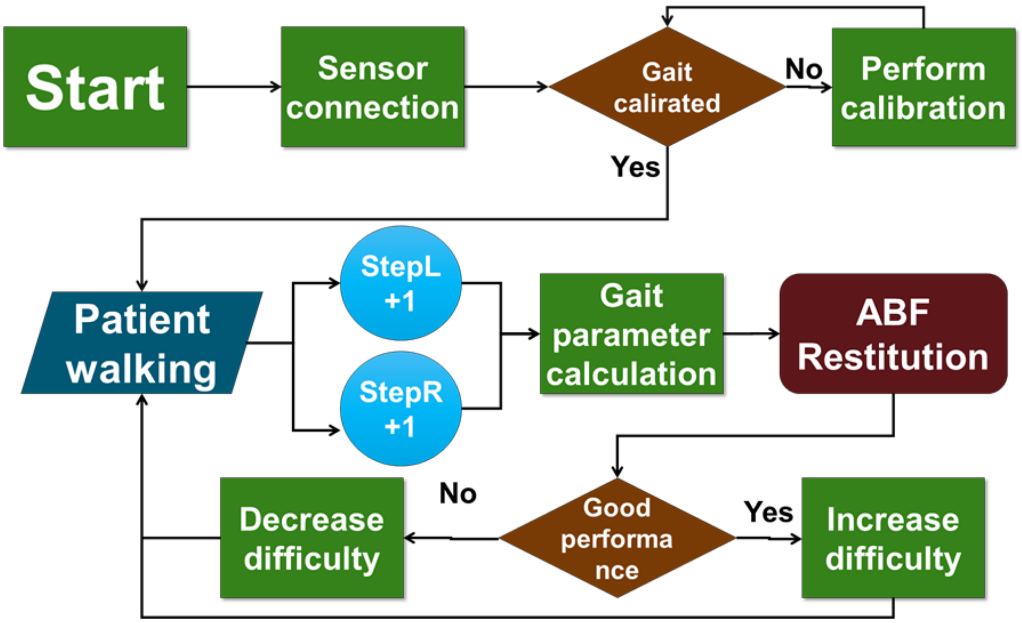


Figure 2.6: Block diagram of the audio feedback application..

Considering the huge variability among the gait patterns of PD patients, the reference values cannot be the same for each patient, but should be tailored to specific subjects needs, taking into account the actual patients status and clinical expectations. For this reason, the system includes a function able to compute a subject-specific calibration. During this calibration phase, the patient is asked to walk about 20 to 40 m under the supervision of the clinician. In this session, the clinician assesses the patients gait pattern and gives instructions on how to improve aspects of the gait (e.g., take large steps). When the clinician is satisfied with the improvements made by the patient, the trial data is stored into the system and establishes the target (zone) for following gait trainings. It is the clinicians knowledge and patients locomotor evaluation that determine what gait pattern and overall performance the patient should provide during this calibration trial. In this way, the system stores a customized target that the patient should commit to maintaining along the following daily practice.

Conceptually, the vocal requests of changing the pattern are fed back to the patient every time the (mean between right and left sides) percentage difference of the current value with respect to the reference is above an upper tolerance or below a lower one. Upper and lower tolerances represent unitless parameters, theoretically variable from 0% up to 100%, that is, respectively, complete or absent adherence to the reference. The resulting confidence interval

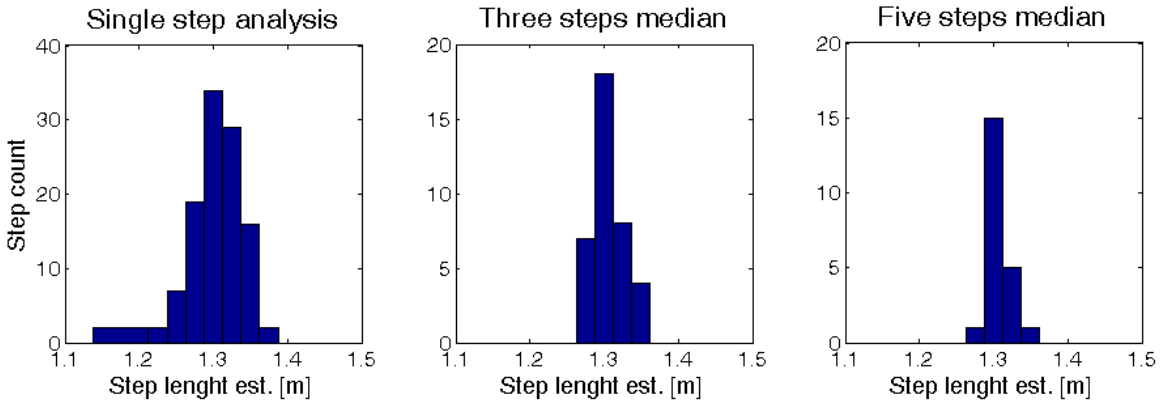


Figure 2.7: Average estimated length of steps for 1, 3 and 5 step windows.

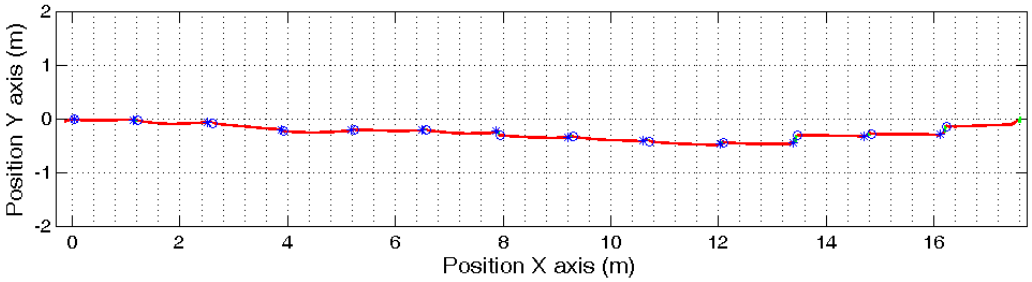
allows the system customization on the basis of patient motor abilities. Once the patient is able to perform within this interval, a vocal reinforcement is played.

Noteworthy is the system being able to automatically increase the difficulty of the task to follow a patients performance. In particular, every time the patient is able to remain within the confidence interval for a certain amount of steps (set by the clinician), the upper and lower tolerances are progressively decreased to a certain extent (set by the clinician), providing a more challenging exercise. Moreover, the system is able to adjust the feedback messages in order to be compliant with patient preferences. In particular, in order to avoid an annoying and/or distracting effect, two solutions were adopted. First, each vocal message provided to patients is randomly selected from three different mp3 files, each containing a different expression of the same content. Second, as long as the patient is able to provide a constant performance, either remaining within the confidence interval or out of it, such that the content of the message is the same, the amount of vocal feedbacks returned to the patient is progressively reduced.

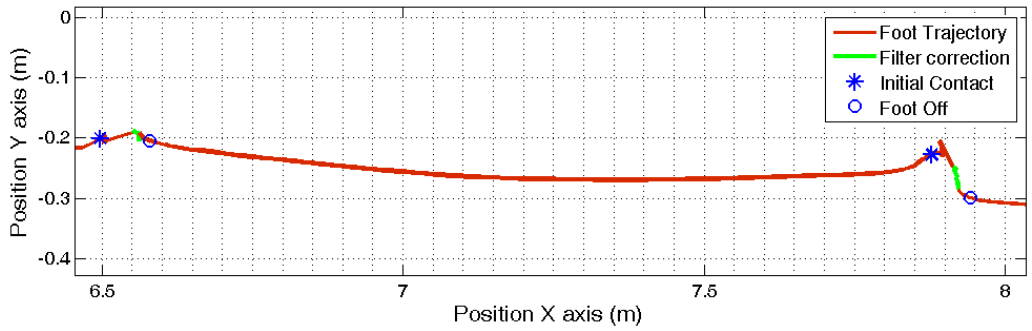
2.2.5 Proposed usage scenario

The typical scenario of use of the system entails the patient walking freely outdoors, in a park for example, over single or multiple periods of 30 minutes each. During a training session, the system measures the selected gait parameters and compares them with respect to reference values over windows of some steps (e.g. every 5 steps).

The system has the additional capability to detect if the patient is walking or not, to avoid the computation and the restitution of feedback while the patient is standing still. This feature was introduced to simplify the usage of the system which can be used during unconstrained walks. It will detect if the patients stops and resume the exercise when the patients resumes walking.



(a)



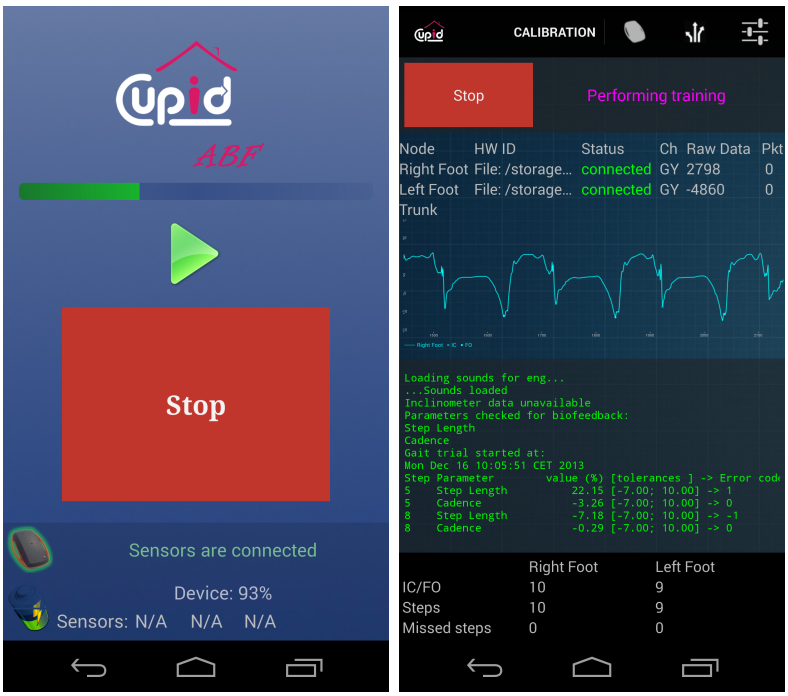
(b)

Figure 2.8: Example of an estimated foot trajectory: (a) during a training walk and (b) detail of a single step.

2.3 Experimental results

The proposed ABF system has been tested on healthy subjects in order to validate its performance and estimate the future usability on PD patients. All the rehabilitation protocol has been tested: from the sensor's mounting on the shoes to usability, parameter accuracy estimation and offline data collection.

The system is intended to be used by the patient and the clinician. The application interfaces dedicated to each of them is shown in Fig. 2.9. The patient (Fig.2.9(a)) has a streamlined



(a) (b)

Figure 2.9: Screenshots of the proposed application: (a) streamlined patient view and (b) clinician (expert) view.

interface which allows to start/stop the training, receive the audio feedback and review the training results. The clinician, on the other side, has access to hidden functionalities which allow to finely tune the system parameters and to review a real time plot of the estimated gait parameters and the reconstructed foot trajectory (Fig.2.9(b)).

The system is designed to be autonomously used by the patient after a first assessment in the clinic. During this first use, a clinician shows to the patient how to operate the system and acquires a calibration walk, which will be used as a reference during subsequent trainings.

2.3.1 Calibration walk

The reference walk acquired by the clinician is used to extract parameters that will be compared with the ones computed in real-time. This walk is performed by the patient under the supervision of a clinician. The reference walk is of great importance to assess user performance during the rehabilitation session. From it the reference parameters are extracted, i.e. standard deviation, average of mid-swing phase angular velocity, the number of steps for each foot. The

accuracy of computed mid-swing phase angular rate is determined by sensors accuracy and is in the order of 0.03 dps.

2.3.2 Training session

The patient will later use autonomously the system during training sessions, which consist of unconstrained walks. The algorithm used to compute temporal gait features (step duration and cadence) relies on the detection of IC and FO events to identify. To prove its reliability it has been tested by a user for a total of 1000 steps. The user was counting the executed steps and compared the step count with the algorithm output, resulting in zero error for step detection and segmentation.

The evaluation of the results of the spatial features computed by the system (e.g. step length) has been performed on guided walks. Markers on the floor were used to aid the user to perform a guided walk and to keep a fixed step length through a continuous walk. The results from the step length estimation algorithms have been compared with the distance of the markers on the floor. Markers were equally spaced by a distance of 130cm, and the analysis was performed on both feet, but in two separate sessions: a 110 step session for the right foot and a 100 step session for the left foot were recorded. From the results shown in Fig. 2.7 we can observe how the average step length is always centered on the actual step length. Since the biofeedback is not returned on a single step, but the median value of 3, 5 or more steps is considered, we presented also those results which show an increase in the precision of the estimated average step length. It can be noticed that standard deviation error decreases as more steps are taken into account and it is respectively 0.0358m, 0.0216m and 0.0156 m for the one, three and five steps.

Standard deviations are: Step length estimation gave us an average value of 131cm, this is a systematic error, but in our approach we compare a reference walk with the output of the algorithm, a systematic offset present both in trial and calibration does not affect accuracy when evaluating patient's gait performance. The estimation of the step length is computed from the reconstructed foot trajectory, obtained using the algorithm described in Section 2.2.4.

This approach takes advantage of the different stages of a gait cycle: it double integrates the accelerometer value to estimate the foot position and uses a Kalman filter approach to correct this estimation while the foot is not moving. An example of the reconstructed foot trajectory is shown in Fig. 2.8(a) for a 13 steps walk and in Fig. 2.8(b) for a detailed reconstruction of a single step. The red line represents the estimated foot trajectory, while the green line shows the filter correction during the stationary phase. The IC events are marked with a star, while a circle marks the FO events. The correction impact on the estimation can be viewed as a metric to evaluate the quality of the reconstruction of the trajectory via the accelerometer integration. Analysis of the collected data shows that the correction has a little impact: it has an impact of less than 4% on the estimation of the trajectory of each step. Once the 3D trajectory of a step is estimated a straightforward computation of the step spatial features is performed.

2.4 Conclusion

In this chapter we presented an intelligent system to tutor gait in patients with PD. Noteworthy, the system can be applied in a generic case when patients need to train walking capabilities. The system is composed of two wearable inertial sensors to be worn on shoes and a smartphone acting as a portable processing device.

The application running on the smartphone collects sensor data from the two nodes and computes the patient's foot trajectory and all gait spatio-temporal parameters during a prolonged walking trial. The computed parameters are compared with reference ones set during a calibration trial. According to this comparison, a feedback is returned to guide and tutor the user to improve his/her performance. The algorithm is capable of automatically adjusting task difficulty to meet the patient's challenge and level of fatigue.

Chapter 3

Firmware improvements

In this chapter we are going to present research on two topic to improve lifetime and reliability of the Cupid System described in chapter 2. Namely the definition of power management policies ti improve battery life and a novel synchronization algorithm based on Bluetooth Piconet internal clock. Results o this chapter can also be applied to different WBANS based on similar components (inertial sensors and Bluetooth transceiver)

3.1 Power management

Once the general hardware architecture has been analyzed (section 2.1.2, we highlight general software architecture. We define a state-machine, depicted in Figure 3.1, which is typical of sensor node operation.

- Low power mode: all components are configured to save power with respect to normal operations
- Wakeup: during this phase components are restarted or reinitialized if necessary
- Read sensors: at this stage values are read from sensing elements
- Process data: if necessary data might be compressed or processed

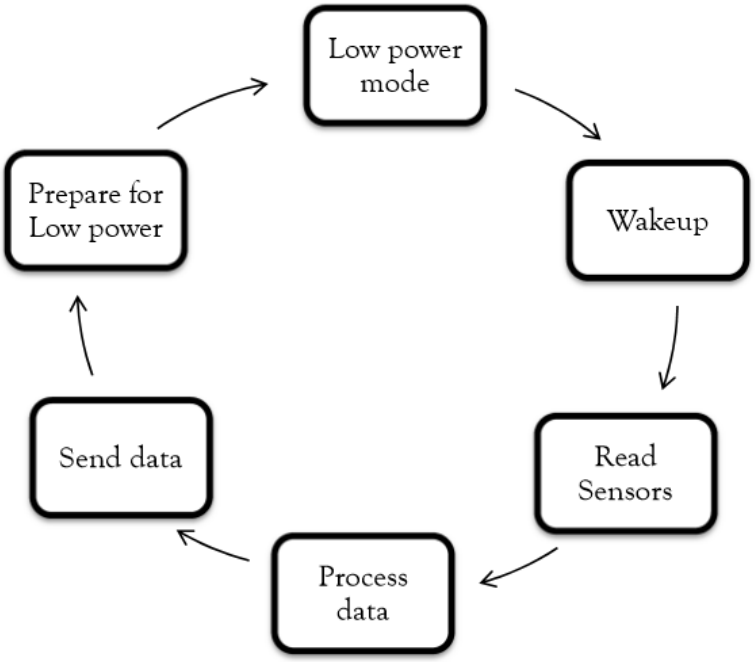


Figure 3.1: State machine for typical WBAN node

- Send data: if request by application data can be sent, or alternatively stored on the node
- Prepare for sleep: this state is necessary to configure the source of the next wakeup event

Usually modern MCUs have different low power modes, which differ from each other for two aspects: power consumption and wakeup time. A key requirement for a practical low-power strategy is to avoid compromising sampling frequency because of power state transitions.

3.1.1 Usage scenario-aware Policies

It must be pointed out that a WBAN can be constituted by heterogeneous nodes, and each of them can be used in different scenarios. In most of these scenarios, one key parameter that affects power consumption is the sensors sampling frequency; indeed sensor nodes can operate in:

1. Data streaming: in this configuration data read from sensor are directly streamed to other nodes or to a master device. For every sensor sampling, one data is transmitted, and the only free parameter remains the sampling frequency.

2. Data Storage: if the node has an onboard memory, data can be logged and sent later, for an offline analysis. Transmission is not required in this case, and main parameter affecting power consumption is sampling frequency.

3. Real-time feedback: in this scenario the node or the network is able to process data and return a sensory feedback to the patient (e.g. using actuators). Also power consumption spent for data processing is mainly affected by sampling frequency

Keeping in mind which are the possible usage scenarios for nodes, we can now define power management policies. We would like to point out that our purpose is not to define "generic" low power policies, but to focus WBAN sensor node application. From typical sensors bandwidth, we can identify the following scenarios:

- Low sampling frequency (below 50Hz): in this scenario low power modes should be widely used to spare energy

- High sampling frequency (above 50Hz): in this scenario low power modes should be used carefully and sparingly

- Event driven sampling frequency: typically this is a scenario in which sensors are always on and send interrupts to other components.

A parameter that can be used to choose the proper low power policy is the energy spent per cycle, defined as the energy used by the node between two consecutive sampling operations. Energy used per cycle can be calculated with the simple formula:

$$E_{cycle} = \sum_{i=1}^{n} P(i) \times T(i) \tag{3.1}$$

where $P(i)$ is the power consumption in a defined state of the node, $T(i)$ is the time spent in that state and n the number of states. Looking at Figure 3.1 it is easy to understand that using different policies to preserve energy will affect only *Wakeup* and *Low Power* states. It is

possible therefore to expand the formula considering the following contributions:

$$E_{cycle} = E_{run} + P_{lp} \times T_{lp} + P_{wake} \times T_{wake} \tag{3.2}$$

Where E_{run} represents the energy spent for operations that must be performed using any of the low power policies. And can be expressed as: $\sum_{i=1}^{n-2} P(i)_{run} \times T(i)_{run}$; this energy is constant and does not depend on the sampling frequency or power saving policy, at the same time, once policy has been chosen, $P_{wake} \times T_{wake}$ is constant. Graphical representation of (3.2) can be seen in Figure 3.2, the graph may help to choose the correct policy, as the sampling frequency grows, it is always more convenient to use less aggressive sleep states since this will give an energy advantage.

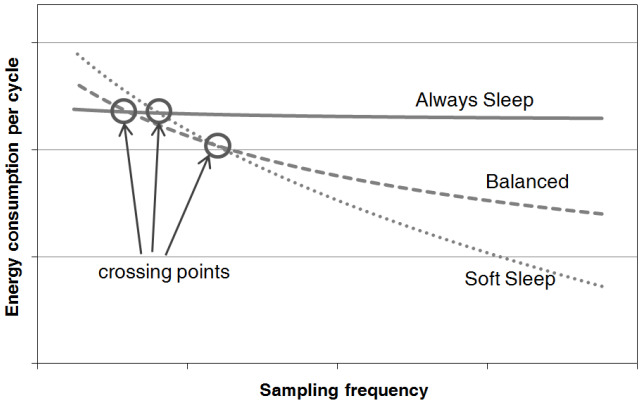


Figure 3.2: Different power saving policy comparison

Three different policies are shown in Figure 3.2, which we will refer to as *Always-Sleep, Balanced* and *Soft-Sleep* and the energy consumed by each policy for different sampling frequencies. If the system is in a situation where sampling frequency is low, it will be convenient to use always-sleep policy, where aggressive low power techniques are used. Always-sleep policy requires that all elements (radio, memory, MCU, sensors) enter in the deepest low power mode. This will guarantee very low power consumption during the sleep phase, but in contrast requires consistent power during wakeup phase. Balanced policy uses low power states, but trying to keep essential components (sensors and MCU) responsive, this will reduce significantly wakeup time and also wakeup energy; in contrast, the balanced policy increases power consumption during low power state, that is why this policy is efficient when the node is instructed for

medium sampling rate. Soft-sleep is the configuration in which the MCU is most responsive, in this scenario, the wakeup energy is very low, and higher power consumption can be measured during low power state; this configuration is best used with high sampling rates.

3.2 Power management for device based on COTS

An implementation of the described policies has been made on an actual node base on Components Off the Shelves (COTS), used to monitor gait. The sensor node on which experiments and optimization has been performed (see Figure 2.2) is equipped with an STM32F103 MCU based on cortex M3 capable to operate up to 72MHz, with a combination of inertial MEMS sensors by ST Microelectronics (gyroscope, accelerometer, magnetometer), which can take advantage of sleep states to reduce power consumption. The device is provided with a Bluetooth module with embedded stack that implements standard Bluetooth low power modes (sniff, park and hold) and a NAND flash memory with 8Gb of capacity.

The node has been designed to be flexible and capable to operate in different usage scenarios, each of them can be power optimized with the correct low power policy adoption. The adopted policies for different applications are:

- *Balanced*: Data streaming (data from sensor nodes will be streamed via Bluetooth to a master node)
- *Soft-sleep (radio Off)*: Data Logging (data from sensors will be stored in the on board NAND Flash memory)
- *Soft-sleep (radio On)*: Data Streaming + Data Logging
- *Always-sleep*: Event driven approach (only few data are sampled after a specific event occurred)

In all cases the node will be part of a Bluetooth piconet.

3.2.1 MCU

The policy integration process has been mainly performed on the MCU, optimizing power consumption for each operation, we'll now discuss how we optimized power consumption, problems encountered and which result the application of the policy provided.

Clock Frequency

MCU clock frequency has been decreased as much as possible to reduce power consumption. We've noticed several differences from theoretical expectations. For example according to STM32 manual, I^2C peripheral clock must be at least 2MHz for standard mode and 4MHz in Fast mode; these value refers to a configuration in which only one sensor is connected to I^2C bus. When multiple sensors share the same lines, bus capacitance C increases, when coupled with pull-up resistor R, this results in an higher RC time constant. This becomes significant with increasing bus clock frequencies, as less time is available for the line to rise. I^2C peripheral is controlled by MCU trough polling; If polling frequency is not sufficiently high, some I^2C events might be lost, compromising communication. The minimum operating frequencies for peripheral and core clock are shown on Table 3.1.

Table 3.1: Minimum operating frequency

MCU Clock	Periph. Clock	I2C Clock	UART baud
12 MHz	6 MHz	100 KHz	115,2 Kbps
24 MHz	6 MHz	200 KHz	230,4 Kbps

Low power modes

Taking into account the three low power modes of STM32, we have mapped our three policies on each of them, here after described:

- Sleep mode: only the core clock is halted, peripheral clock is still distributed and peripherals are active. Wakeup time from this state is just a few microseconds.

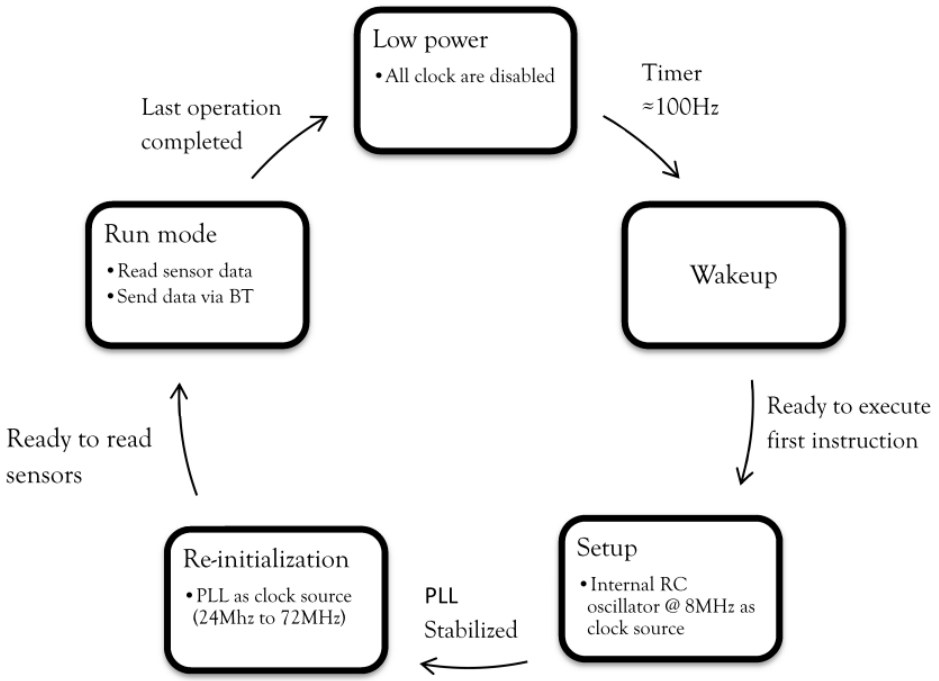


Figure 3.3: State machine for low power modes

- Stop mode: core and peripheral clock is disabled and SRAM content is preserved. Supply current in this mode is 24μA and wakeup time varies from few microseconds if the internal oscillator is used as clock source, to hundreds of microseconds if the PLL needs to be reinitialized.

- Standby mode: core and peripherals are switched off and SRAM content is lost, backup registers content is preserved and all pins are in high impedance. This allows the lowest power consumption with higher wakeup time.

A comparison among the three available sleep modes has been performed to assign each low power state to one policy. The state machine in Figure 3.3 represents MCU states and condition to exit. The states are here explained:

- Low power mode: the MCU is in Sleep, Stop or Standby mode according to the configuration analyzed

- Wakeup: in this state power regulator is enabled if were disabled and clock is distributed to MCU components

- Setup: during Stop and Standby, PLLs are disabled and need to be reinitialized to be selected as clock source

- Re-initialization: this state is necessary only when the MCU goes in Standby mode, and it is necessary to reinitialize MEMS and restart the program (since RAM content is lost)

- Run mode: in this state the MCU reads data from sensors and sends them via Bluetooth

In Table 3.2 we present results in terms of current consumption together with wakeup timings for MCU only, referred to a sampling frequency of 100Hz. Current consumption measurements have been performed using current probes on VCC line, using a different board, since the pins that powers the MCU, on sensor node, are not accessible.

3.2.2 Policy adoption

Using information presented in Table 3.2 and the formula in (3.2) it is possible to evaluate policy adoption to our real case; more in detail we have mapped the low power policies according to the following scheme:

- Sleep mode: mainly used in soft-sleep policy

- Stop mode: mainly used in balanced policy

- Standby mode: used only with always-sleep policy

Measurement of energy consumption for each policy at different operating frequencies is shown in Figure 3.4, which should be compared to the theoretical results of Figure 3.2. We have already discussed MCU configuration, and in the following we will present radio and sensor configuration to explain deviations from theoretical expectations.

Table 3.2: Low power modes timings

	Standby Mode:	Stop Mode:	Sleep Mode	Unit
Current consumption in Standby	2	24	5800	μA
Current consumption during Wakeup	5	5	5	mA
Current consumption during Setup	10	10	10	mA
Current consumption during Initialization	19,4	19,4	19,4	mA
Current consumption in Run mode	19,4	19,4	19,4	mA
Time spent in low power mode	4620	6714	6874	μs
Wakeup Time	50	6	6	μs
Setup Time	160	160	0	μs
Sensor and Peripheral Initialization	2050	0	0	μs
Time spent to read and send data (Run mode)	3120	3120	3120	μs

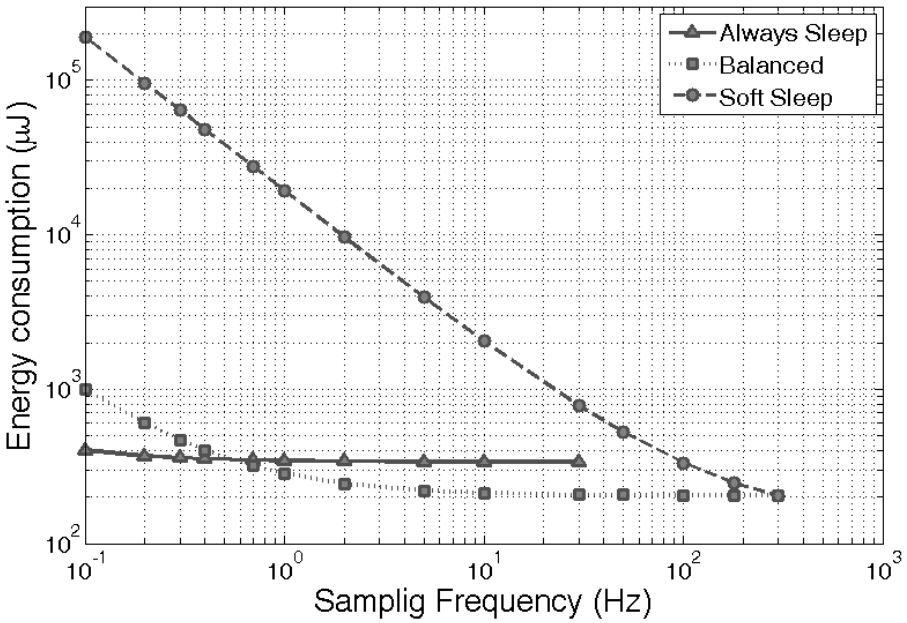


Figure 3.4: MCU Power consumption per cycle

3.2.3 Bluetooth Power Management

The radio section in a sensor node contributes significantly to power consumption, therefore an effective low power strategy must deal with it. The Bluetooth module uses 3mA when in idle state and this value can decrease to 0.4mA in deep sleep mode. The latter state makes the module not visible and therefore not able to establish a new connection. When the module is connected to another Bluetooth device, power consumption is 22.4mA, even if no data is received or sent, and around 30 mA when transmitting or receiving. Bluetooth stack already includes some low power modes: Park, Hold, and Sniff. The latter has been implemented,

allows less power consumption than Hold and better response time than Park [KRK03]. In Sniff mode, if there are no data to send, the slave node sleeps for a defined period and then wakes up to receive data from the master. It is possible to configure the Bluetooth module so to accord the Sniff period to the data transmission frequency, this configuration seems to be the most efficient, since the radio module will wake up only when new data are ready to be transmitted. Communication among Bluetooth modules only occurs during assigned timeslots; each timeslot has duration of 625μs. A parameter of the sniff mode defines for how many slots the Bluetooth should not communicate with the master device; this parameter must be an even number greater than zero. As result theoretical minimum sniff interval is 1.25 ms. Sleep time has been set to 100ms, in this configuration power consumption is reduced to 3.5mA when in Idle, and command are received with a maximum delay of 100ms; but data can be sent with the desired frequency. This will justify our choice to use the radio in sniff mode for the *soft-sleep* and *balanced* policy; deep sleep mode has been used in *always-sleep* policy. Measurements have been performed on a separated Bluetooth module and considered average current consumption in a period of 10 seconds.

Table 3.3: board components power consumption

MEMS (active)	8.1 mA
MEMS (low power)	1.1 mA
LEDs (duty cycled)	0.6 mA
Board	1.7 mA

3.2.4 MEMS & LEDs

For the selected application (body movements monitoring) sensor's output data rate (ODR) is set to a value near 200Hz. Higher ODR does not give useful information and increases the noise. Inertial sensors are powered by a dedicated voltage regulator that can be enabled or disabled by MCU. After wakeup, sensors needs to be reinitialized, setting the desired registers via I^2C. In addition, there is a wakeup time, which is $\frac{5}{ODR}$, that is 31ms using the ODR more suitable for the application. If sensors are alternately switched on and off to save power, sampling frequency must be reduced to values lower that 30 Hz. Combined current consumption of MEMS sensors is shown in Table 3.3. LEDs could also be a significant source of power consumption, the

board is equipped with a green and a red LED. They are in fact duty-cycled to reduce current drawn. To measure power consumption of the board it is possible to disable the MEMS's power regulator, remove Bluetooth module and put the MCU in stop mode.

3.3 Discussion on power reduction achievements

Having described MCU, radio and sensors configuration, it is now possible to justify differences between expected energy consumption of Figure 3.4 with respect to Figure 3.2. First, as already anticipated, it was not possible to decrease MCU clock frequency as much as expected. Minimum operating frequencies are shown in Table 3.1. Lower values do not allow proper communication with sensors or radio module. The clock frequency has been fixed at 24MHz, even if the microcontroller can work down to 12MHz. Working at 12MHz allows to reduce power consumption of 50% w.r.t. working at 24MHz, at the cost of requiring twice the time to perform the same instruction set. Nevertheless two considerations guided our choice: in the first place, lower clock frequency does not affect power consumption of all components, therefore power reduction is not strictly proportional to clock frequency; moreover, as the time necessary to read and transmit data increases, less time is left for processing or idle state, allowing to work only at lower sampling rates. As result, the slope of the curve representing *Soft-sleep* policy is higher w.r.t. Figure 3.2. The reason is that Fig. 3.2 ideally expect to have I^2C and USART working at the minimum clock (that in our case would be 4MHz), while we are forced to use a clock of 24MHz with higher power consumption in run mode.

The second consideration, when we compare results of Figure 3.4 with prediction of Figure 3.2, concerns the maximum achievable sampling frequency. In our system we had to take into account the time necessary to read and send data (shown in last row of Table 3.2). This limits the maximum sampling frequency to 300Hz. As a consequence, the *soft-sleep* policy seems to be advantageous only in a very tiny range of frequencies around 300Hz. It is also evident that the *always-sleep* policy has not been evaluated for all possible frequencies. The reason is similar to the previous case: it is due to the impossibility to reach frequencies above 30Hz. The

limitation in this case lies in the board hardware: when the MCU is in the lowest power mode, it disables all GPIO therefore turning off also the sensors, with the consequence that they must be reconfigured, each time requiring 31ms to setup. This also dramatically increases break even time [BBDM00a]: the energy spent to enter and exit sleep mode for the *always-sleep* policy is so high that its use is worth only for very low sampling rates.

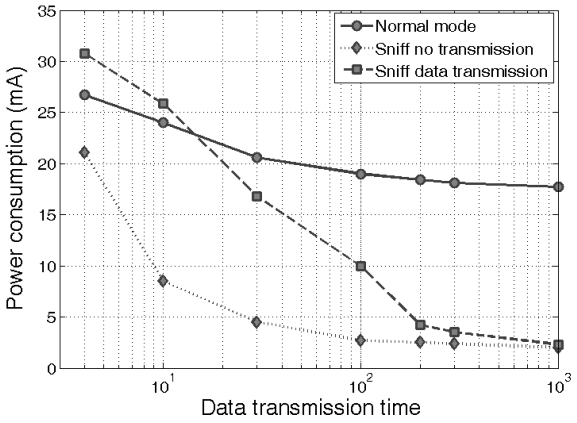


Figure 3.5: Bluetooth power consumption

Power consumption of Bluetooth module showed deviation from expectations as well. According to our hypothesis, the use of sniff mode should significantly decrease energy consumption. Such hypothesis is only verified when Sniff interval is high compared to timeslot duration. In figure 3.5 we showed current drawn by our Bluetooth module in different configurations: (i) when the sniff mode is disabled and data are transmitted with a period equal to the sampling frequency; (ii) when the sniff mode is enabled, we have set the sleep period of the sniff mode equal to the sampling rate; (iii) Sniff mode is enabled with different sleep periods, but no data are transmitted. Measurements have been performed ranging from sniff interval of 3.75ms (minimum achievable value of 6 time slots) to 1sec. It is evident that Sniff mode is not favorable for sniff intervals below 15 ms. This is an unexpected result, but we can explain it as follows: in normal mode, Bluetooth modules are requested to communicate with the master device every 20ms; in Sniff mode communication occurs after a period defined by the Sniff interval. If the Sniff interval is lower than 20ms, the low power mode actually adds an overhead to the communication, which is not present in normal mode.

As final result, we want to show the energy consumption of the whole system, including sensors,

radio module and MCU (Figure 3.6). We choose sampling frequencies that are common for our specific application: gait monitoring. It is evident that the use of an appropriate policy can greatly reduce energetic costs. Energy consumption can be reduced from 7.5% @ 300Hz to 30% @ 30Hz, up to 3 times when sampling @ 3 Hz.

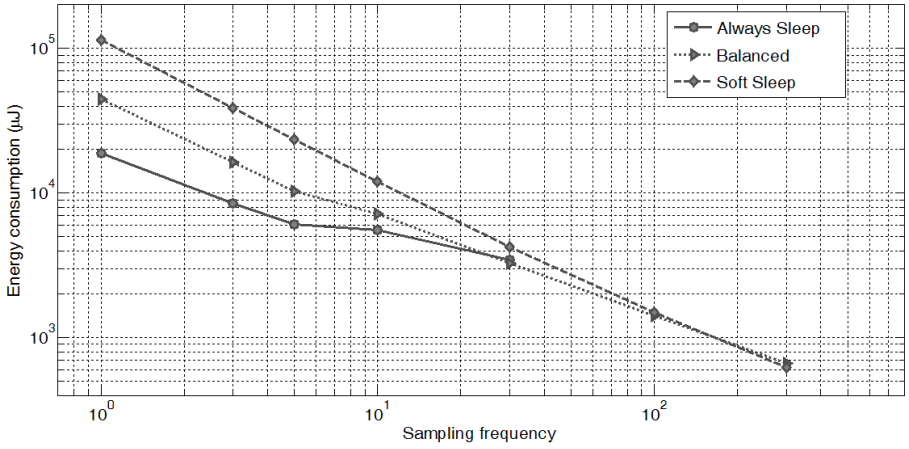


Figure 3.6: Overall system energy consumption

3.4 Synchronization on Bluetooth WBAN

Bluetooth is widely used in Wireless Body Area Networks, since it guarantees both high data rate (up to 2Mb/s) and inter-operability with a number of devices. However, while early WBAN applications required in many cases just a single node [CLCC09], more and more the number of nodes is increasing to monitor diverse physiological parameters (e.g. motion, temperature, pressure, galvanic skin response, etc.) or to track the same parameter but at different body location [TTT+09]. In such cases and particularly when sensor fusion algorithms need to be used, a common time reference among all WBAN nodes is necessary.

The level of accuracy required for WBAN synchronization depends on the application and on the parameters to be monitored (e.g. for temperature monitoring also 1 sec error might be acceptable, while for ECG or inertial measurements, errors above 1 ms can not be tolerated [PHH+11]). Data synchronization might be also performed through wired sensors, but this is an uncomfortable solution. This chapter shows how we addressed the problem of synchronization

in a Bluetooth based WBAN where sensors are placed on human body for gait analysis. Target users are Parkinson's Disease (PD) patients, for which parameters of interest are gait velocity, cadence, legs rotation while walking, legs and arms movement symmetry, etc. In this context, it is reasonable to have more than one node to improve accuracy, but not so many to limit usability. The novel method proposed in this paper enables:

1. Achieve 1ms accuracy in a Bluetooth network.
2. Perform synchronization with no communication overhead and simple modifications to sensor node's firmware.
3. Achieve higher performance when Bluetooth low power modes are used.

In the following, we explore two approaches: the first uses the information on the data reception time from the master and the latter is based on the information provided by the Bluetooth Piconet Clock. For each of them we will evaluate possible improvements and application scenarios.

3.4.1 WBAN synchronization: application requirements

In the context of WBAN we are developing a system for gait monitoring and analysis, thus our requirement is to use inertial sensors to identify benefits of a rehabilitation therapy or drug on motor behaviour. Synchronization among nodes is crucial to identify small improvements and specific parameters such as gait symmetry. The requirements of our application, concerning radio protocol can be summarized as follow:

- High data rate (a minimum bandwidth of 500kb/s is needed, but 2Mb/s are desirable).
- Easy interoperability with other devices such as computers, smart-phones or tablets.
- Network size from 3 to 7 nodes plus a master device.
- Synchronization error must be lower than 1ms.

- Good noise immunity and transmission range compatible with WBAN applications.

Sensor nodes should maintain synchronization during data streaming, and during data logging, when few or no data is transmitted through the radio. Those requirements lead to the choice of Bluetooth technology.
Our WBAN is based on identical nodes (shown in Fig. 2.2) each of them equipped with inertial sensors, a micro-controller, NAND FLASH memory, Bluetooth radio and a battery. Data from sensors are sampled at a frequency that can vary from 50Hz to 400Hz, system can operate with a minimum of 3, up to 7 sensor nodes. Other radio protocols such as ANT or Zigbee do not have sufficient data throughput (16 bit x 9 DOF x 400Hz x 7 sensors).

Unfortunately, Bluetooth does not provide a high-level mechanism to synchronize data with accuracy needed for our application. Simple synchronization methods in a Bluetooth network can achieve accuracy in the order of tens of microseconds [RGL04], which is unacceptable for gait monitoring, since some gait parameters like heel strike have a duration of few milliseconds [LVW+10] [CL10]. When it is necessary to combine information of multiple sensors to reconstruct gait parameters, synchronization method should provide time resolution similar or better than the sampling frequency [ISH12].

3.4.2 Time Of Arrival

Bluetooth channels use a Frequency-Hop/Time-Division-Duplex (FH/TDD) scheme. Transmission/reception takes place in timeslots that are 625 microseconds in duration. The master-to-slave transmission starts in even numbered slots, while the slave-to-master transmission starts in odd-numbered slots. Masters and slaves are allowed to send 1, 3, or 5 slot packets, which are transmitted in consecutive slots. Information can only be exchanged between a master and a slave. A slave is allowed to start transmission in a given slot, if the master has addressed it in the preceding slot. The master addresses a slave by sending a data packet or, if it has no data to send, a 1-slot POLL packet. The slave must respond by sending a data packet or, if it has nothing to send, a 1-slot NULL packet. In many WBAN applications, when continuous

monitoring is needed, data are sent periodically with the same time interval. This is the case we want to analyse to synchronize data on the receiver side.
In a Bluetooth network, when sensor nodes periodically send data, if the receiver node (usually also master node) is aware of the transmission rate that each sensor uses to send data, it is theoretically possible to synchronize data coming from different nodes. This can be done thanks to the (FH/TDD) scheme (i.e. transmission occurs only in periodic time slots). Unfortunately, this kind of setup would also allow limited flexibility, since adding or removing a node, or modifying transmission rate will affect time slot assignment. For this reason, almost all Bluetooth devices have a buffer capable to store incoming data from sensors and transmit them when one or more time slots are assigned by the piconet's Master.

To measure synchronization in this condition the setup is straightforward and simple: a single timer runs on the receiver node and when data are received, the sensor's value is stored together with timer value. This synchronization method is easy to implement and does not add any overhead to data communication, but requires continuous and constant data streaming from sensors. Different setups have been tested: using multiple nodes and using different devices as network master, best results are obtained when an Android mobile phone is used as master. This configuration allows to stream data from 6 sensors sampling at 400Hz.

Results of this synchronization method are shown in Fig. 3.7 and are compatible with a similar approach described in [WOL11]. From the graph, it is possible to notice that synchronization error increases with data rate, but it also depends on the number of nodes connected. Moreover, this approach is strongly dependent on the master, on the implementation of the Bluetooth stack and on the operating system and workload; in fact, tests with a supposedly more powerful computer achieves worse results than a phone. Therefore the approach is not robust and scales poorly. Unfortunately, those results did not meet the requirements of our application; the delay is strongly dependent on the number of connected nodes, on the master device and on data rate. Moreover, in many cases synchronization error is above 10ms, unsuitable to produce an accurate gait analysis.

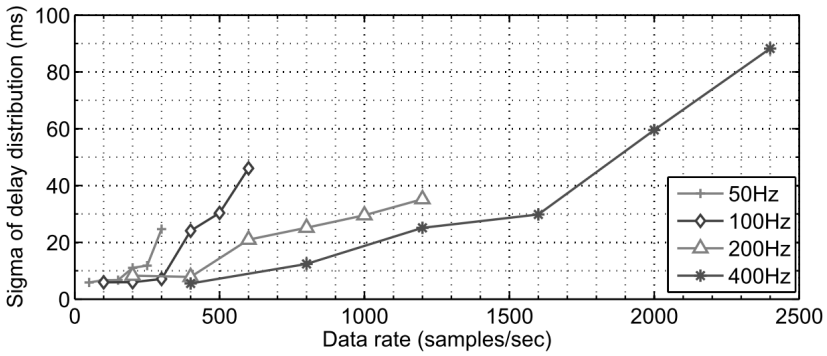


Figure 3.7: Synchronization measurement results, data rate is dependent from sampling frequency and number of nodes connected

3.4.3 Bluetooth piconet clock

The Bluetooth clock is a 28-bit counter with 0.3125 ms resolution and a mandatory maximal drift of ±20 ppm. This results in an overrun every $2^{28} \times 0.3125$ ms ≈1 day.

Each Bluetooth device has a unique 6-byte base-band address (BD ADDR) similar to the medium access control address of Ethernet devices. The hopping sequence is a pseudo-random sequence of communication frequencies calculated from the BD ADDR of the piconet master device. Because of the frequency hopping, a special procedure called inquiry is required to discover other devices (i.e. their address and hopping sequence). During an inquiry, a device uses a special inquiry hopping sequence and doubles its hopping rate to rendez-vous with other devices. As a result of an inquiry, the BD ADDR and the difference between the local Bluetooth clock and the clock of the remote devices are acquired. Based on this information, a node can calculate the hopping sequence of discovered nodes and is thus able to connect to these devices.

The clock offset between two connected devices is specified as the difference between the clock of the slave node (CLKslave) and the clock of the master node (CLKmaster). Each slave node stores in a local register the offset, this value can be used to reconstruct the CLKmaster, so that all nodes refer to a common clock. It is thus possible to use the reconstructed CLKmaster value to synchronize all nodes. Measuring the offset between the system clock and a Bluetooth clock is non-trivial as reading the Bluetooth clock requires sending a command message to the Bluetooth modem over the serial interface between the main processor and the Bluetooth modem and receiving a reply message (a so-called event) over the serial interface that contains

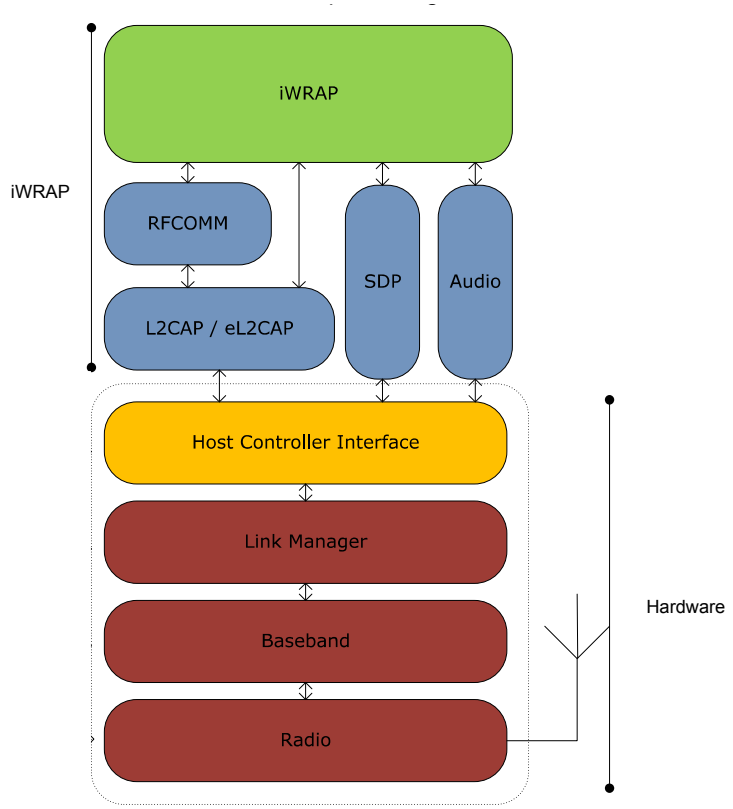


Figure 3.8: Bluetooth stack implemented in WT12 modules

the requested clock value.

Reading piconet clock: Our purpose is to use a standard Bluetooth adaptor and measure the value of the common clock present in the piconet. Before explaining our clock reading procedure, we will briefly introduce Bluetooth stack elements since this helps to understand our results, our methodology and the source of error when reading piconet clock. The Bluetooth protocol stack is split in two parts: a "controller stack" containing the time-critical radio interface, and a "host stack" dealing with high-level data. Controller stack is composed by:

1. Bluetooth radio: it is the transceiver that transmits and receives modulated electrical signals from peer Bluetooth devices. It is not part of the stack since it is a physical chip but directly communicates with the baseband layer.

2. Baseband: it is the physical layer of the Bluetooth, which manages physical channels and links apart from other services like error correction, data whitening, hop selection and Bluetooth security.

3. Link Manager: it essentially handles link set-up, security and control. It also manages devices in different modes (Park, Hold, Sniff and active).

While the host stack contains:

1. L2CAP: it is the Logical Link Control and Adaptation Layer protocol. L2CAP permits higher-level protocols and applications to transmit and receive L2CAP data packets.

2. RFCOMM: it is the protocol that emulates the serial cable line settings and status of an RS-232 serial port and is used for providing serial data transfer. RFCOMM connects to the lower layers of the Bluetooth protocol stack through the L2CAP layer.

Communication between host stack and controller stack is performed through the Host Control Interface, a standardized interface with common commands that allow communication also through different hardware devices [Blu12]. The information that we need (the piconet clock, or shift between Master clock and Slave Clock) is updated in the Baseband layer, which is accessible though HCI. The implementation of the host stack is never trivial; there are essentially three options: one is to buy the license for a proprietary Bluetooth stack; a second one is to develop or port an existing stack to the desired platform; a third one is to use modules that already implement a stack with different profiles. For the latter case, it is not always possible to send direct HCI commands to the host stack, because HCI communication is managed by other software/hardware components that are not accessible. This is a very common situation, to reduce developing time and costs, sometimes a solution that works out of the box is chosen; many vendors implements specific command to read piconet clock. To read piconet clock value it would be desirable to have direct access to baseband layer (Figure 3.8) in order to obtain minimum communication overhead and delay. As said this is not possible, since lower accessible level is HCI. The complexity of Bluetooth stack (and often the presence of two separate chips) clarifies the reason why in many cases reading piconet clock is inaccurate.

3.4.4 Piconet clock synchronization method

Our synchronization method has been implemented on Bluegiga WT12 Bluetooth module that uses a proprietary Bluetooth stack called iWrap shown in Figure 3.8. WT12 modules have the flexibility to be controlled using iWrap firmware or alternatively bypass it and use HCI commands. We choose the former alternative since our micro controller host cannot run application firmware and Bluetooth stack. Our configuration had 7 slaves sensor nodes and one master device. The master after issuing and establishing a connection using Serial Port Profile (SPP) to slave nodes, was able to send commands to manage sensor's operations and eventually to receive data from them. In addition if Bluetooth implements a Health Device Profile (HDP), SPP has been chosen since for data streaming the two profiles are comparable in terms of packet loss [NDCB10].

3.4.5 Bluetooth clock

As first approach, we choose to read piconet clock at regular intervals using the iWarp command *"CLOCK {link_id}"* where link_ id specifies the connection of which clock is read, since Bluetooth module can be part of different Bluetooth networks, thus having different values for piconet clock. In our case all sensor nodes are slaves, thus the command is *"CLOCK **0**"*. Unfortunately, commands cannot be sent to Bluetooth module once the connection has been established: all data sent by the sensor node MCU to the WT12 module are retransmitted to the master node. If the connection drops or is intentionally closed, it is possible to issue commands to WT12. The alternative is to switch the module from "data mode" (default mode when connection is established) to "command mode" (default mode when no connection is established); this can be done sending a special character, but the whole operation requires at least 2 seconds or configuring the node to switch mode raising a pin.

The operations to be performed to read the Bluetooth clock are shown in Fig. 3.9. Once the Bluetooth clock is read, the value is stored together with data on the NAND flash memory on the node, or a special packet is sent to the master node. Accuracy of our synchronization

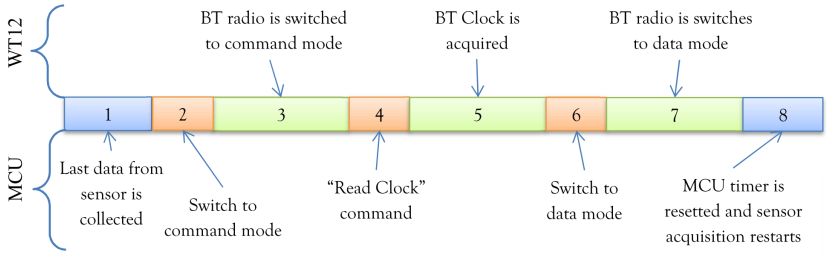


Figure 3.9: Sequence of operation performed during to read piconet clock

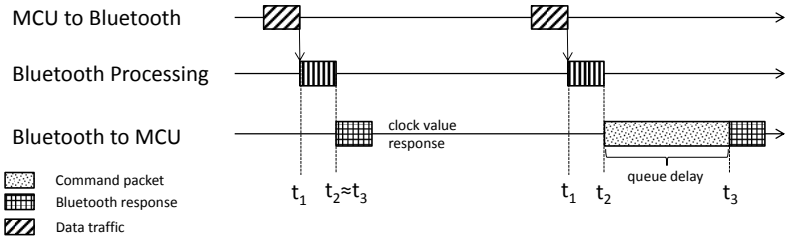


Figure 3.10: Description of how the *"read clock"* command is processed by a Bluetooth module [RR07]

method was measured using the following setup: a pin from each of the sensor node MCU was connected to the same cable; firmware on MCU was configured to detect interrupt on that pin with the highest priority and read piconet clock when such interrupt is detected. Piconet clock value is then compared among all nodes.

We had unsatisfactory results with the described configuration: synchronization error had a mean value of 15ms and was often above 30ms.

3.4.6 Bluetooth clock correction

Since all Bluetooth nodes in a piconet must be synchronized and the clock offset with respect to the master node is stored in the Baseband layers, we suspected that such delay was due to Bluetooth stack operations performed by the host controller on WT12 modules. We indeed noticed that the response time to the read clock command was not constant. Our hypothesis is that the command is received and processed in a constant time, but the response time can vary according to how busy the host controller is. Figure 3.10 represents what happens when a command is issued to Bluetooth node. It is possible that when the command is issued (t_1), the

host controller is busy for other operations [RR07], and the response does not occur soon after processing (t_2), but instead may be delayed and sent after completing other operations (t_3). It has to be noted that the Bluetooth module does not transmit any data while in command mode, but it can accept incoming connections and is visible to other devices. Furthermore, incoming data are buffered, thus we can infer the host controller is busy in random times. In this condition, it is possible to remedy and mitigate correcting the received piconet clock. It is indeed possible to start a timer on the sensor node MCU right after the last byte of the command is sent to the WT12 module and stop the timer when the first byte of the response is received. Therefore, all the modules will start the timer with equal delay (sending the command requires the same amount of time on every node). The piconet clock value can be then corrected using the value of the timer. The above approach result in a systematic error, which equals the time between the transmission of the command and the actual readout of the piconet clock. This systematic error can be easily cancelled since we use Bluetooth clock as time reference and the time difference between two measures elides the error.

3.4.7 Improving piconet clock correction

Our analysis has been moved to the piconet clock acquisitions with the higher synchronization error, trying to find a feature that may allow to filter them out. Our efforts have been repaid when we noticed that the acquisitions that required more time are also the ones with higher error. A simple threshold filter has then been implemented on the MCU cutting out piconet clock values that requires 5% more than the average reading time. In this case, a new acquisition can be made if the reading time is in the acceptance range.

3.4.8 Low power performance

Since WBAN nodes relies on batteries, power management is crucial to extend sensor node's operating time. Bluetooth protocol provides some low power mode (Park, Sniff, Hold). Among them Sniff mode is the more appealing since combines module's responsiveness and low power

consumption. A Bluetooth module in the Sniff mode stays synchronised in the piconet. It listens to the piconet at regular intervals (T_{Sniff}) for a short instant. This enables it to re-synchronise with the piconet and to be able to make use of this Sniff window to send or receive data. The consumption is as low as the T_{Sniff} is large (compared to the Sniff window). If T_{Sniff} is in the region of a second and the duration of Sniff is in the region of several ms, the consumption will be about 1 to 5% of the maximum transmission consumption (average consumption in our module ranges from 3mA to 5mA approximately). In our synchronization method, we hypothesised that piconet clock readings were delayed because the host stack was busy. Using Sniff mode we should increase accuracy; experimental results confirms this hypothesis.

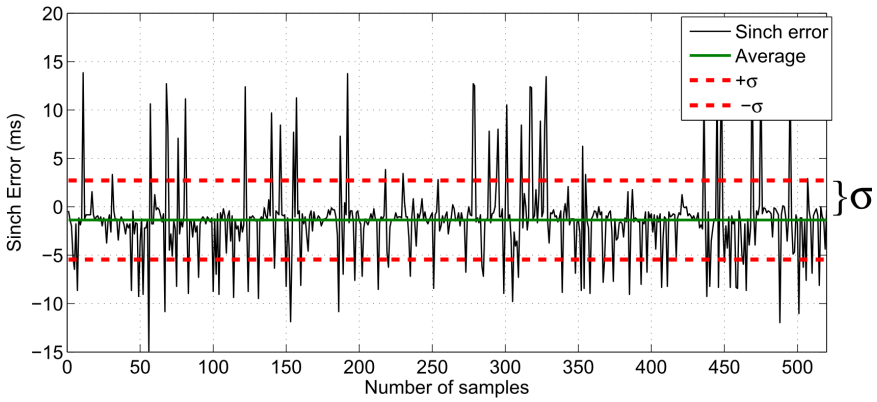


Figure 3.11: Synchronization error after correcting the Bluetooth clock value

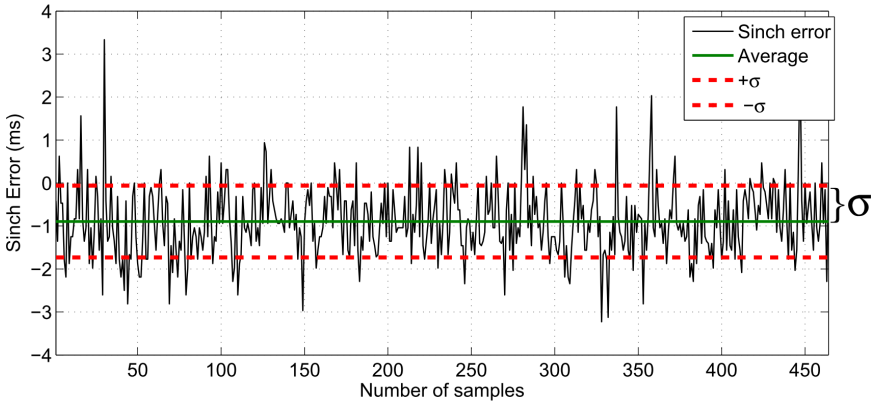


Figure 3.12: Synchronization error excluding piconet clock values when reading time is above 125ms

Table 3.4: Results when reading piconet clock in Sniff mode

T_{Sniff}	Ck read time	% of discarded	Synch accuracy
0 (no Sniff)	122.1 ms	12.1%	1.039 ms
50 ms	94.3 ms	9.8%	0.618 ms
100 ms	93.6 ms	9.0%	0.370 ms
1000 ms	93.3 ms	7.5%	0.313 ms

3.5 Synchronization accuracy assessment

We present here the results for synchronization accuracy using method described in section 3.4.6, 3.4.7 and section 3.4.8. Measurements have been performed using the following approach: an electric pulse was sent at the same time on all sensor nodes. Piconet clock value is then read and compared among all nodes. Measurements have been performed in both cases during data streaming and when no communication occurs; this two cases did not show any difference in algorithm performance. Results of the synchronization method described in 3.4.6 are shown in Figure 3.11, where the delay measured when reconstructing synchronization from two sample nodes is plotted. In this case, error can be quantified in 5ms. We improved these results, using the approach described in 3.4.7, as it can be seen in Figure 3.12, synchronization accuracy improved; it has indeed been measured 1,039ms as synchronization error. We have tested the WT12 modules in different power saving configurations. We noticed that Sniff mode not only reduces power consumption, but it also improves piconet clock reading time and accuracy. Using the approach described in 3.4.8 we measured synchronization error in Sniff mode. Results are shown in Table 3.4. It is evident that when the WT12 module is in Sniff mode we have better results in terms of discarded readings and synchronization accuracy. This is due to the fact that the controller is less busy in Bluetooth stack operations when Sniff mode is enabled and can be more responsive to commands sent by MCU. Synchronization in Sniff mode achieves up 3 times better accuracy than synchronization in normal operation mode.

3.6 Conclusion

In this chapter, we have analyzed how power consumption of a WBAN node can be greatly affected by sampling frequency, and how is possible synchronize Bluetooth based sensor nodes with minimal firmware modifications. We have proposed a policy based approach that matches many typical usage scenarios and can be used to select low power mode according to the node's sampling frequency. The use of policies has been implemented in a COTS based sensor node. As result, we have noticed that the model holds with good approximation. Nevertheless, the policy that keeps the node in the shallowest sleep state is not profitable, due to platform constraints, making the balanced policy the most usable for a wide range of sampling frequencies.

Furthermore, we have seen that state transition costs are still non-negligible and that peripheral management is often the limiting factor in the selection of the lowest power state.

For the Bluetooth synchronization methods, both implementations have been built on top of Bluetooth Stack, and can be easily integrated in most of available Bluetooth modules. We would like to remember that the configuration used by the sensor's MCU does not implement the Bluetooth stack; in fact, we used a Bluetooth module that integrates host and controller. This choice reduces dramatically developing time. As result, it is possible to synchronize data using the time of arrival, but accuracy is strongly affected by number of connected nodes and data rate. If Bluetooth piconet is used, and we apply an exclusion algorithm to remove delayed readings, error decreases to 1ms. In many WBAN applications power efficiency is crucial for success; we have also proved that when in low power (Sniff) mode synchronization accuracy can be further improved reaching a minimum error of 0.313ms. Our synchronization method does not require HCI access and adds no communication overhead if data are stored on the sensor node. If synchronization is performed on the master node, overhead consists in only 4 byte to be sent every few minutes.

Chapter 4

Smart Pen for Fine Movement Assessment in PD

In this Chapter we present another system developed for PD rehabilitation, the CuPiD Project was focused on Gait rehabilitation, this system has its main focus on upper limbs, trough assistive handwriting.

4.1 Overview

Handwriting analysis through technology-enhanced devices brought the opportunity to accurately capture and analyze writing features and opened the way to a whole new range of applications to augment the handwriting experience. One of the fields where this trend is being leveraged is the rehabilitation and assessment of illness severity.

An effective solution targeting this application field must satisfy demanding requirements in terms of spatial-resolution, precision of the input device and an appropriate working area. Furthermore, for "human-in-the loop" training and rehabilitation applications, requiring feature analysis for direct feedback provisioning, the system must meet strict real-time constraints. We present a system designed with all the above mentioned requirements. We present an in-depth quantitative characterization of working area, spatial precision and real-time performance. De-

sign constraints were derived from the application domain for fine motor skill rehabilitation and handwriting dysfunctions; a common scenario for Parkinson's Disease patients.

Parkinson's Disease (PD) and Parkinsonism in general are neurological disorders that can impact a person's fine movements abilities [Jan08]. Clinical evidence has shown that a typical PD symptom affecting the gait (Freezing of Gait or FoG [N$^+$11]) can also affect the distal upper-limb fine motor. Attempts have been made to tackle these problems both in the disease severity assessment [C$^+$10] and in the rehabilitation therapy fronts. Handwriting is a preferred target domain to find interesting clinical approaches to study and mitigate the disease progression involving the distal upper-limb fine motor skills, as shown by [P$^+$08, T$^+$97].

The setups used in existing solutions, often involve a computing workstation where the patient interacts with an input device like a tablet, a digitizer or some custom device connected to a PC. Those approaches need the clinician to be always present during the exercise sessions to help the patient deal with the interaction. Such systems can be uncomfortable to use, in particular at home, for people not used to technology, even though the patients are highly motivated to avoid the decline or loss of handwriting ability. For example, the use of a tablet and a resistive touch screen as input device can result to be unnatural for a PD patient and can hamper transfer to daily life writing. She/He has to be careful not to lay the hand on the device to avoid glitches on input data, also the perceived effect of sliding the tablet pen on the surface and seeing the computed stroke displayed on the screen is not as natural as a regular pen writing ink on paper.

We have designed a solution to meet the needs of ageing PD patients who are not used to "virtual handwriting" on glassy surfaces. In addition, our handwriting device is fully untethered and does not require a connection to a PC. Furthermore, it has a natural and streamlined user interface which does not require the presence of a clinician during the exercise sessions to guide the patient. The system uses a commercially available digital pen connected to an embedded platform that analyzes the handwriting and gives real-time natural audio feedback to the patient guiding and hinting her/him during the exercise execution. At the same time, the device logs all the data to be then analyzed by the clinician that downloads the exercise sessions data logs into a PC application.

In this scenario the patient can perform the exercises at home, in a comfortable and familiar environment and without the stress induced by a supervisor. Therapists can, in addition, compare the physical handwritten exercise against the data recorded by the device. The system, in fact, enables to extract quantitative information on the writing performance.

4.2 System architecture

To pursue our purpose to implement a handwriting tool providing a familiar and natural feeling, we decided to augment a commercial digital pen. Digital pens are input devices able to capture and convert analog writing information created using pen and paper into digital data. Typically this data are used by other applications such as digital notebook, or artistic drawing applications. They can be based on different technologies such as inertial sensing, cameras, infrared or ultrasounds emitters and receivers. Unfortunately, the majority of those digital pens are not open to developers and when an SDK is provided, it is limited to few functionalities. Therefore it is not possible at the moment to develop our algorithm directly on the commercial device. Thus, we identify a couple of them, compliants with application requirements (see Section V), with which it was possible and sufficiently easy to capture spatial and temporal coordinates and developed a prototype add-on board where real-time performance feature extraction and audio-feedback was implemented. We also designed a set of pre-printed sheets reflecting the exercises typically given by therapists. We also use pre-printed elements for calibration and exercise selection at the beginning of a training session.

To better understand the architecture of the system, the description of a typical exercise session is given. The patient conducts the exercises writing with the digital pen on a pre-printed paper corresponding to a specific writing exercise assigned by the therapist and, depending on the level of difficulty, she/he will be supported by visual cues such as pre-printed rows. The Digital Pen provides the handwriting digitized data corresponding to the tip position that are processed by the main board to extract from raw data writing performance features (e.g. speed or size). Depending on the specific exercise, the main board renders an auditive feedback such

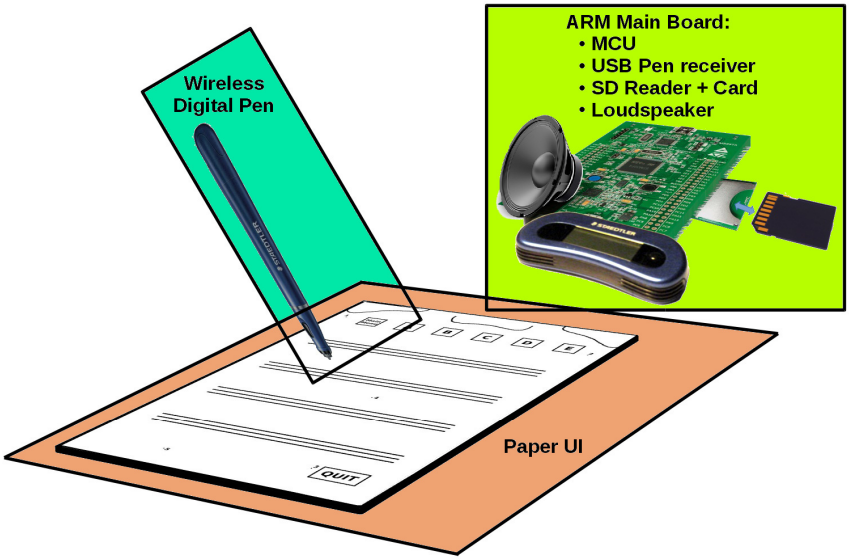


Figure 4.1: Hardware Setup.

as a pre-recorded audio message (e.g. "you're slowing down your speed", "please augment the size", "you're doing fine", etc.). At the same time raw data are logged in a SD memory card for off-line further analysis. Therefore, the system considers the user in the loop. In fact, the writing performance extracted by the user interaction with the system generates a feedback that stimulates the user to correct her/his writing, which is again the input to feature extraction from which the feedback is generated accordingly. The user, writing with a normal pen on paper at home, without medical supervision, is relaxed and performs the rehabilitation exercises without stress, giving the best context for performance assessment to the clinical analysis.

The prototype device designed is composed of three main hardware/physical components (Fig.

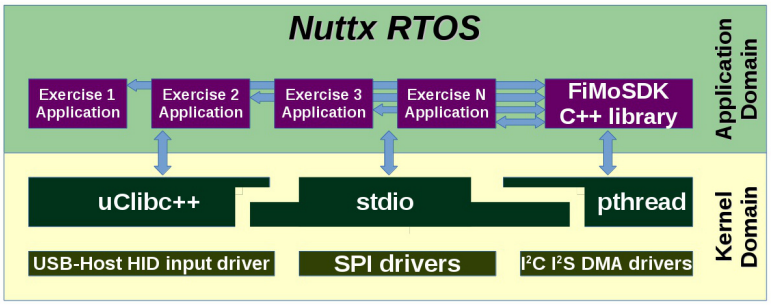


Figure 4.2: Software Architecture.

4.1):

1. the Paper UI;
2. the digital pen;
3. the main board.

The software architecture is composed of the following parts (Fig. 4.2):

1. the NuttX RTOS;
2. the FiMoSDK library;
3. the exercises applications.

The digital Pen receiver is connected to the main board using the USB interface and the USB 2.0 FS[1] controller using the HID[2] class protocol extension.
The Speaker is directly connected and driven by the output pins of the audio codec chip [Cir10], the chip uses two interfaces with the MCU: an I2C interface for control and an I2S interface for audio samples transfers. Finally the SD Card reader is connected to the MCU using an SPI interface.

4.2.1 Digital Pens

In our work, we focused on two different commercially available digital pens; the Staedtler DigitalPen 990 [Sta], and the Wacom Inkling [WAC]. Both devices work using infrared and ultrasound technology. The pen is paired with a receiver (merged in the main board to form a single box) fixed on the paper. The pen simultaneously transmits infrared encoded beacons and ultrasound bursts. The receiver triangulates the pen tip position counting the time interval between the IR reception and the ultrasound reception by two different ultrasounds receivers. Both devices have been tested and characterised to assess their fitness for the application (see Section V.A).

[1]FS: Full-Speed; a signaling rate of 12Mbit/s introduced by USB 1.1.
[2]HID: Human Interface Device; a standard USB device class extension.

4.2.2 Paper UI

The digital pen selected enables writing on regular (blank) paper without pre-printed patterns. However, to implement the training protocol designed by therapists for at home rehabilitation, we needed to design a Paper User Interface to enable the selection of daily exercises and to implement the specific assignments. At the same time, to capture the writing exercise correctly, the paper enables an initial calibration. The Paper UI therefore provides a framework where the user can interact with the training program by means of input commands and audio guidance. Fig. 4.3 shows the layout of an example page where the various elements are highlighted:

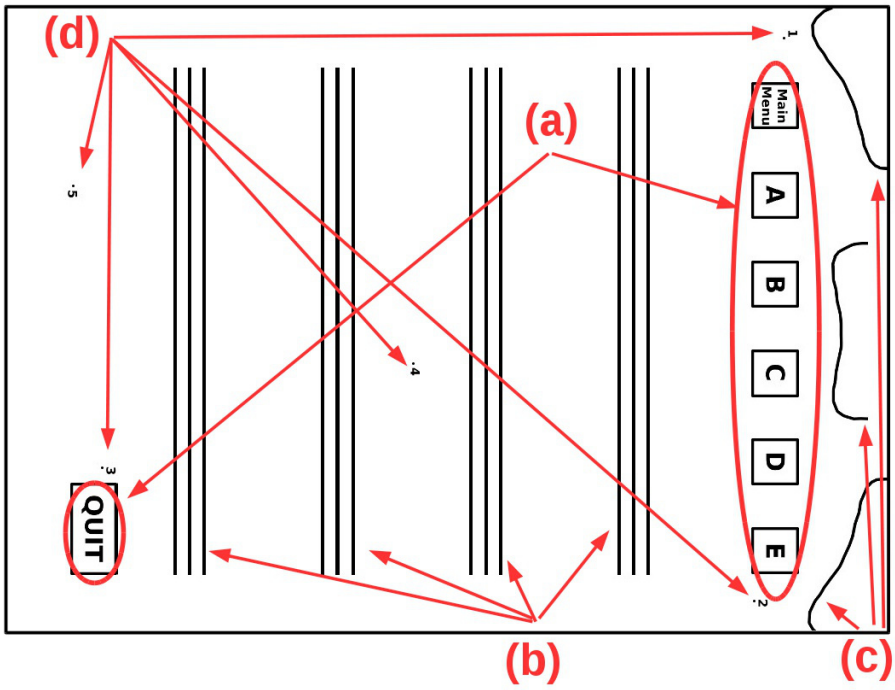


Figure 4.3: Paper User Interface elements layout. a) Menu Buttons; b) Exercise rows; c) Pen Receiver suggested positions; d) Calibration points.

a) Menu Buttons: The user, following the audio voice indications, writes a cross inside a box to select a specific command or to activate an exercise.

b) Exercise Rows: These are examples of visual cues customized on the specific exercise. Depending on the configuration parameters the rows can be present or not, partially present and thay can be set to different heights and the paper must be printed accordingly.

c) Pen Receiver suggested positions: The receiver can be placed anywhere around the paper, the top positions are only suggestions.

d) Calibration Points: The prototype system works with 5 calibration points. Their coordinates in the "default" reference system are stored in the application configuration file. They can be changed and/or increased in the printed layout and in the configuration file.

4.2.3 Main board

To prepare a preliminary prototype of the main board, we exploited the availability of the STM32F4-Discovery, based on a Cortex-M4 MCU with 1024KB of Flash memory, 192KB of RAM and running at 168MHz. The board includes many useful peripherals, such as USB-OTG FS, USART and SPI interface; it also includes an audio CODEC chip.
The main board therefore extended the Discovery board with an SD card reader using the SPI and a loudspeaker to the audio jack. We used the USB interface and micro connector to connect the digital pen receiver, and the audio CODEC for the audio feedback rendering. The board can be connected to a host PC via USART for console interfacing, for debug and development purposes. All this hardware components form a single entity to be packaged in a box.

4.2.4 NuttX RTOS

Among the Operating Systems targeting the selected board, the NuttX [Nut14] RTOS has been chosen; it has mature support for the board, is extremely modular and mimic the POSIX standard offering a functional shell and a set of utility to manage the system. It supports C++ language with most of the OOP language features and offer uClibc++, a standard C++ library implementation. NuttX has a binary loader to run applications from a file stored in any local or remote media. It also has POSIX `pthread` facilities to implement a multithread application, like our FiMoSDK library. NuttX is supported by an active developers community and uses a BSD[3] license to distribute the code.

[3]BSD: Berkeley Software Distribution

We integrated the USB-HID driver with the missing logic to support the digital pens. For the audio codec device, we integrated the corresponding STM libraries in the NuttX project. The work needed to patch NuttX in order to support the USB pens, has been the implementation of the input driver the RTOS exposes to the applications.

4.2.5 Fine Movements SDK

The FiMoSDK C++ library is a framework we designed to be used to implement handwriting exercises. `HWInferrer` and `AudioFB` are the entry-point classes for the application code; `HWInferrer` infers handwriting features using all the rest of the framework's classes. `AudioFB` manages the output of audio messages, and can be thought of and used as a `printf()` function that plays the audio file instead of writing messages on the console.
A `RawData` instance is used to receive the samples from the pen and to package them into `Sample` instances for further uses by `HWInferrer` and for logging purposes.
The `Stroke` class is instantiated to store contiguous sets of `Samples` having the `Tip` boolean attribute set. Each `Stroke` is analysed by the `HWInferrer` to compute its features like length, bounding box, speed and direction.
The `rect` class produces support objects to manage bounding boxes and allows mutual boolean comparisons like intersection or inclusion. `calibrationDesc_t`, `calibrationPoint_t` and `transformationParams_t` are supporting structures used to manage the calibration as explained in the following section.
The `HWInferrer`, `RawData` and `AudioFB` classes are designed to execute their main task using a separate thread of code execution. All the objects that may fail, throws `FiMoExceptions` if they do reach an error condition.

4.3 Algorithms

In the following we describe the calibration and the logic governing the audio feedback, both algorithms implemented in the FiMoSDK.

4.3.1 Calibration

For the digital pens to work, it is important to place the pen receiver on a fixed position w.r.t. the writing paper. Once started the writing session, the receiver must be integral with the paper to avoid distorted data acquisitions.
Since the system must be used by the patient without supervision at home, it is important to make it robust w.r.t. the receiver placement.
The problem to be tackled each time the system is started, is to find the right parameters for the reference roto-translation with respect to a "default" reference taken once with the receiver placed in a well known position. The solution is to use n calibration points' coordinates (5 points in our case), compare them with the same points taken in the "default" reference system and infer the parameters of the following linear roto-translation transformation:

$$\bar{P_r} = \bar{\bar{A}} \cdot \bar{P_a} + \bar{b} \quad \equiv \quad \begin{bmatrix} P_{r_x} \\ P_{r_y} \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} \\ a_{yx} & a_{yy} \end{bmatrix} \cdot \begin{bmatrix} P_{a_x} \\ P_{a_y} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

where: $\bar{P_r}$ is one calibration point from the "default" reference system; $\bar{\bar{A}}$ is the rotation matrix of the linear transformation; $\bar{P_a}$ is the corresponding calibration point in the "actual" reference system; $\bar{b}$ is the translation component of the linear transformation. In this linear system, $\bar{P_r}$ and $\bar{P_a}$ are known; $\bar{P_r}$ coordinates were taken once and are stored among the configuration data of the system. $\bar{P_a}$ are retrieved at run-time during the calibration sequence. The unknown data are $\bar{\bar{A}}$ and $\bar{b}$. Theoretically 3 calibration points would solve the equations; but the noise and the human inaccuracy in tapping on the exact calibration point suggest to include some redundancy. The solution is to create an over-determined linear system using as many points as possible and find the optimal solution that minimize the error from the exact one; the minimization in the least square sense is proved to be an effective technique for linear systems affected with normal noise. The core of the calibration is based on the Least Square fitting algorithm implemented in the GSL project [Gou09]. As a tradeoff between accuracy and usability, we reduced the set point to 5. The system passed as input to the library is $\bar{y} = \bar{\bar{X}} \cdot \bar{c}$; where: $\bar{y}$ is the vector of the observations ($\bar{P_{a_n}}$ in this case); $\bar{\bar{X}}$ is the matrix of the predictors ($\bar{P_{r_n}}$ in this case); $\bar{c}$ is the

vector with the unknown parameters to be fitted ($\bar{\bar{A}}$ and $\bar{b}$ in this case).
The numerical method also computes the coefficient of determination R^2 that indicates how well the regression fits the model, ranging from 0 (no meaning) to 1 (perfect fit).
The algorithm implemented in the library uses the R^2 coefficient to filter out potential outliers (for example if the user tap on a wrong point). The code tries to use all 5 calibration points in the LS fitting, but if the R^2 coefficient is under a selected threshold (part of the application configuration data), then 5 iterations follows where the LS method is used with 4 calibration points; each time excluding a single point. If the results show 4 solutions with a good R^2 and only one result with a bad (under-threshold) R^2, then the code can discard the single calibration point that was wrongly acquired and validate the calibration parameters using the remaining 4 valid points.

4.3.2 Audio Feedback

An important aspect is the audio feedback generation management. During the exercise session, the user periodically receives audio messages indicating her/his performance; whether she/he is doing good or is writing out of the exercise specifications. The approach used in the prototype demos, uses a set of configuration parameters to fine-tune the output frequency and the threshold over which the messages should be given.
The demo application checks in real-time for stroke features like size and velocity. For each feature a numerical `TargetValue` is stored in the configuration, together with two thresholds; a `ThresholdGood` value and a `ThresholdBad` value. The thresholds are percentages of the distance between the actual value from the `TargetValue`. Using these three parameters, a `Score` value is computed that ranges from $[-1; +1]$ using the relation shown in Fig. 4.4.

In addition, the application switches between different states not to overload with too frequent feedbacks the user prolonging an inadequate performance, "relaxing" the `ThresholdBad`. The application is normally in a `Neutral` state; during the exercise, if the user goes out of `ThresholdBad`, the state gradually switches to a `Relaxed` state, through two intermediate states: `PreRelax1` and `PreRelax2`. The state changes if the distance overflows the relaxed

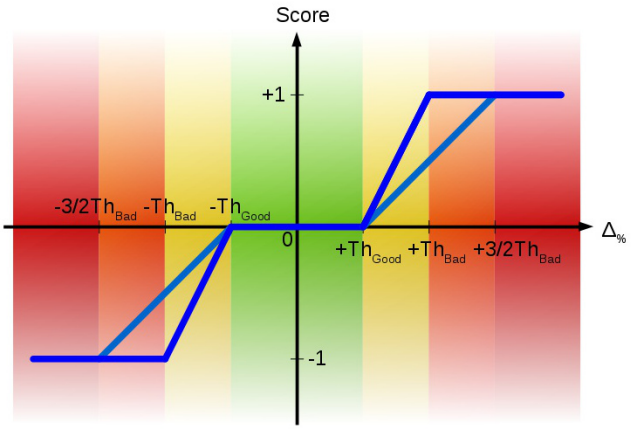


Figure 4.4: Score relative to the difference between measured value and TargetValue. Th_{Bad} and $3/2 * Th_{Bad}$ are used depending on the application state.

`ThresholdBad` value, as shown in Fig.4.5.

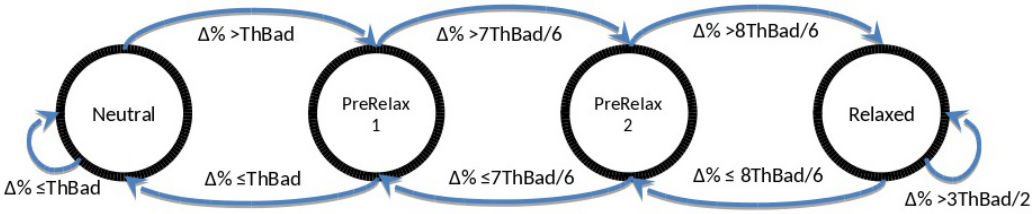


Figure 4.5: The state transitions happens with values from ThBad to 1.5·ThBad offering an adaptive behaviour to help and motivate the user.

4.4 Experimental Results

Here after we present the results obtained by the wireless digital pens characterization. Afterwards, a preliminary test on healthy users is described to assess system usability.

4.4.1 Digital Pen Characterisation

As previously explained, a system designed to offer real-time handwriting analysis and feedback, must necessarily meet specific requirements. At this regard, the choice of the input device, i.e. the digital pen, is crucial. It acts in fact as the sensing element of the system and therefore needs to be validated in terms of spatial resolution, working-area, sample frequency and accuracy.

To the purpose, we tested in particular two digital pens: the Staedler Digital Pen 990 and the WACOM Inking pen. Results are presented in the following.

Staedtler Digital Pen 990

The Staedtler DigitalPen is a commercially available USB-FS 1.1 input device. The device adheres to the official HID USB profile using a proprietary protocol extensions to produce the data. The physical device uses infrared and ultrasounds emissions to track the pen tip position. The time difference between light and sound propagation is used to track the pen tip distance and to triangulate its position using two embedded ultrasound receivers. The pen tip reacts to pressure with a switch that, when in Mouse mode, is used as the left mouse button. The pen is also equipped with an additional button located in the pen body. When connected to a host device via the USB cable, the pen can work in two different modes; "Mouse" mode and "Pen" mode.

When the device is in mouse mode, the host recognizes it as a standard HID pointing device. The working area of the physical pen is reduced to a rectangular area variable with the screen resolution of the PC.

When the device is in pen mode, the device changes the HID data descriptor used to send the samples. The working area is determined by the receiver range. When in Mouse mode, the

Description	Unit	Value	
		Mouse Mode	Pen Mode
X-Coordinate Range	pt	$\sim [1000; 9000]$	$\sim [-13000; 13000]$
Y-Coordinate Range	pt	$\sim [0; 6000]$	$\sim [0; 16000]$
X length on paper	mm	166	400
Y length on paper	mm	125	300

Table 4.1: The Staedtler Digital pen 990. Ranges in "Pen" mode and "Mouse" mode when the host device has a screen resolution of 1280 x 800 pixels.

working area is typically reduced to a rectangle, which size varies with the host device screen resolution. For example, testing the pen with a device having a screen resolution of 1280×800 pixels, the working area is $166 \times 125\ mm^2$. Coordinates (X, Y) are contained within the intervals shown in Tab. 4.1. Comparing the coordinate ranges with the length on paper, the spatial

resolution of the device is of $48pt/mm$ (or $0.021mm/pt$). The precision is reduced by the noise present in the data produced. The noise absolute amplitude is of 6 points in each direction (Fig. 4.6). Thus the resolution locates each sample within a $0.126 \times 0.126mm^2$ box. There is no

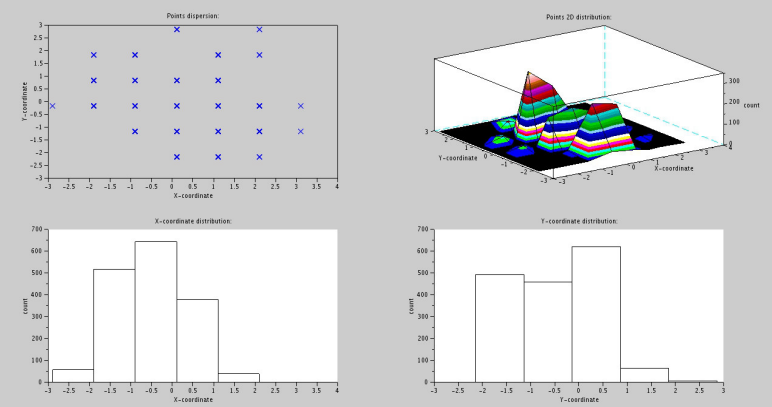


Figure 4.6: The Staedtler Digital pen 990 distibution of the coordinates of the pen tip held still caused by noise.

difference with the spatial accuracy when the device is in Pen mode or in Mouse mode. On the other hand, the timing of the samples produced differs between Pen mode and Mouse mode. When the device is in Mouse mode, it produces a new sample every $15ms$ ($\sigma = 3ms$). When in Pen mode, the samples are presented to the host device in pair every $30.9ms$ ($\sigma = 6.4ms$). The resulting single sample time performance in steady state is equivalent to the one in Mouse mode, but with reduced protocol overhead. The timing results are summarized in Tab. 4.2.

Device Mode	USB Transaction Timings		Single Sample steady state timings	
	Period (ms)	Frequency (Hz)	Period (ms)	Frequency (Hz)
Mouse (one sample)	15	66	15	66
Pen (two samples)	30.9	32	15.45	64.7

Table 4.2: The Staedtler Digital pen 990 samples timings in in “Pen” mode and “Mouse” mode

Wacom Inkling

The WACOM Inkling pen is a USB 2.0 FS device. It is a composite device embedding a MSC[4] memory device, and an HID input device. The interface can be switched and the two devices

[4]MSC: Mass Storage Class. A standard USB device class extension.

cannot be simultaneously active. The device, when connected to a USB Host and used as an HID device, reduces the working area to a box of $200 \times 150\ mm^2$. The pen HID device provides samples that include the pen tip position, the pen tilt and the tip pressure intensity. The working area for the Wacom pen (X and Y ranges) is detailed in Tab. reftab:WacomRange, achieving a maximum rectangle of $200 \times 150\ mm^2$. Comparing the coordinate ranges with the

Description	Unit	Value
X-coordinate range	pt	$[0; 1920]$
Y-coordinate range	pt	$[0; 1920]$
X length on paper	mm	200
Y length on paper	mm	150

Table 4.3: The Wacom Inkling ranges and length.

actual length on paper, the spatial resolution of the device is not proportional. The X resolution is of $9.6pt/mm$ (or $0.1042mm/pt$). The Y resolution is of $12.8pt/mm$ (or $0.0781mm/pt$). The coordinates received by the host are not affected by noise, this is probably because the receiver pre-filters the raw data before sending it over the USB interface. The samples are produced on average every $\sim 6.6ms(\sigma = 0.7ms)$. The USB transactions are distributed with intervals mostly of $6ms$ or $7ms$.

4.4.2 Comparison

After the characterisation, some considerations can be done regarding the performance of each device w.r.t application requirements.

Range: Both the devices have similar ranges, and both have a dual mode ("Mouse" mode and "Pen" mode); the big difference is that the Steadtler device has been designed to allow mode switch when connected to the USB Host, while this is not true for the Wacom device[5].

Timings: The Wacom device presents a faster timing profile for sample transmission that is $\sim 2.5\times$ the Steadlter device. Nevertheless both device resulted sufficiently fast to allow the handwriting analysis needed by the application.

Noise/Precision: Both devices presents a similar precision, due to the fact that both uses

[5]The mode switch command is not available for the Wacom pen, although is likely be to present like it is on the Staedtler pen.

the same underlying technology to track the pen tip. The main difference lies in the fact that the Wacom device pre-processes the raw data, smoothing the resulting stroke and filtering out the noise, while the Steadtler device transmits non pre-filtered raw samples.

X-Y aspect ratio: The Staedtler have a square aspect ratio that results in easier software management of the data (calibration, measurements, ecc.); the Wacom, having an anisothropic resolution, leads to a more complex handling of coordinates transformations.

Extended information: The Steadtler device, along with the coordinate position, transmits the binary status of the pen tip; pressed/released. The Wacom device transmits additional information; the pen tip is provided with a 1024-level pressure sensor, furthermore the pen tilt is transmitted in the packet. This data helps to compensate the effect of different handholds on coordinates calculation.

In this first system prototype we privileged the size of the working area and the isotropic aspect ratio, choosing the Steadtler pen w.r.t. the Wacom pen, even thought the latter offers interesting additional features. We therefore compensate the weaknesses of the Steadtler pen, e.g. filtering the noise on coordinates in the main board firmware.

4.4.3 Field Tests

A preliminary test, conducted on 5 healthy subjects, was aimed to assess the system usability. The main aspects under test were *a*) calibration; *b*) sound quality; *c*) physical setup.

Users were required to write repetitively the character l without interruptions within given reference rows at a given (parametric) speed, filling the entire row height. Test revealed a minor drift $(1 - 2mm)$ on coordinates for both pens due to different pen handhold and inclination between the calibration phase and the actual exercise execution. This drift can be mitigated by adding or subtracting the projection of the tilt to the received coordinates (available on Wacom pen only).

The sound quality of the digital samples is lossless PCM encoded audio sampled at 44000Hz with 32bit resolution; the codec outputs, the internally amplified signal and the final audio quality perceived by the users depends on the quality of the loudspeakers. During the development of

the prototype, different loudspeakers have been used with different results, ranging from signal intensity of loud/noisy for low quality speakers to loud/clear for good quality ones; in all cases the vocal messages content where always intelligible.
The physical setup was acknowledged as unobtrusive and usable and familiar as pen and paper.

4.5 Conclusion

In this chapter, we presented the work of design and implementation of a novel tool to be used in writing rehabilitation. We presented the characterization results and the feedback received by preliminary user tests.
The system targets in particular PD patients that experience difficulties with writing. Our aim is not only to provide a daily life instrument for training at home, but also to enable PD experts to investigate the impact of writing exercise to improve fine motor skills. Furthermore, the configurability of the system will also support the study of which feedback strategy is most user-friendly and effective in the long-term.

Chapter 5

Real time OS for Wearable sensors

5.1 Energy Broker Middleware - Overview

Development of embedded applications, targeting micro-controller based platform, experienced during the last years the same revolution that already took place in the 70s for ordinary computing platforms. The increasing amount of installed volatile and storage memories, together with the higher computational power of recent micro-controllers admit the adoption of software abstraction mechanisms, that 10 years ago were limited to very high-end class devices. We have seen the spreading of embedded operating systems, targeting different device classes, with different features and use cases. All of them are generally defined as Real Time Operative Systems (RTOS) and make use of a Real Time Microkernel (RTM); a microkernel is the near-minimum amount of software that can provide the mechanisms needed to implement an operating system. The definition of a RTOS does not necessarily imply that it provides mechanisms for hard-real time guarantees, but it is a general notion to identify systems that support multi-tasking, with optional task priorities. RTOS used for sensors nodes in Wireless Sensor Networks (WSNs) need also to deal with further constraints: limited computing capabilities, memory resources and energy availability. Many applications in facts run on low power 32bit or even 8 bit micro-controllers unable to integrate an OS like Linux. We have developed an Energy Broker Module (EBM) that can be used as standalone layer or easily integrated in many of the RTOS currently

used for WSNs. The proposed EBM is constituted by three different components: a core, for power management policy decision; hook functions to detect energy status of the system and libraries to interface the specific hardware. As result, we have a lightweight module capable to bring the concept of energy awareness and power management to many RTOS that do not implement any of these features.

5.2 Realted Works

Many complex embedded systems need to have an advanced power management strategy. This is often obtained using a complex OS like Linux that provides governors to scale CPU frequency and manage peripherals. Among different distributions of Linux, Real-time Linux (RTLinux) [Y+99] is widely used in embedded systems, and features Advanced Configuration and Power Interface (ACPI) specification. ACPI provides an open standard for device configuration and power management by the operating system. Unfortunately, Linux Kernel requires resources that may not be available in very low power MCUs; in fact typical MCUs adopted in WSN are e.g. the 8-bit Atmega family, 16-bit MSP430 or even the 32-bit Cortex M3 family, which all have limited resources that do not allow the use of RTLinux. (Table 5.1)

Unlike Linux, the majority of the widely used operating systems for wireless sensor nodes and embedded devices do not include any facility or support for power management; this aspect is generally left to the application. Even where this support is eventually provided, it generally relies upon management of the CPU sleep states, while support for peripheral management and integration is missing or tricky to obtain. In some cases, a specific RTOS has been developed to be used for WSN applications [ERR05]. Such approach guarantees good efficiency, but in many cases it would be desirable to use an already existing RTOS. Some works focus on defining the general structure of power management system for embedded RTOS [ARP06]. Such works lack of an implementation of the defined paradigms in an existing RTOS, making difficult to evaluate performance in a practical implementation. Our EBM can be integrated in an existing RTOS and has been tested in a real use case.

Among the most widely adopted RTOS for WSN and sensor nodes, TinyOS [LMP+05] is the most complete operating system in terms of power management support. It includes sets of primitives based upon Integrated Concurrency Control and Energy Management (ICEM) [KHL+07] derived concepts, and exposes a set of functions to implement parallelization of applications tasks and network stack operations. ICEM based concepts can be implemented also in EBM afterward the inclusion of the framework in operating systems that provide thread abstraction and make possible the definition of proper scheduling mechanisms. However TinyOs uses a programming language different from C, called NesC. The programmers has to get used to a new programming paradigm that includes concepts such as: components, modules, configurations, interfaces, etc. Our approach proposes an EBM that does not require learning a new programming language or the paradigms of a new operating system.

The Contiki operating system [DGV04] provides a set of primitives for power consumption estimation. This support is targeted only to a limited number of platforms, such as Tmote Sky [PSC05] and the embedded sensor board ESB platforms. This mechanism adopts a linear power consumption estimation function, and the application is required to supply data to the framework and act accordingly to the results. The compatibility with different hardware platforms is improving, although as we write, lot of common platforms such as STM32F2xx and STM32F3xx families are not supported.

MantisOS [BCD+05] has been developed at the University of Colorado; it is a specific operating system for WSNs that facilitates the programming of new applications with a completely different approach. Mantis makes use of a multi-threaded scheduler allowing that a short task, with strict time constrains, interleaves its execution with other long complex tasks. However, this ability of accommodating different tasks increases the RAM memory footprint and the energy consumed due to the task preemption [LPSS10]. FreeRTOS: This popular operating system does not provide any mechanism to support application power awareness, as the official documentation does not report any related argument. A special mechanism by FreeRTOS provides the user a function where per-case power management can be implemented. Table 5.1 depicts type of processes used by different RTM together with ROM and RAM memory occupancy.

Table 5.1: Operating systems comparison

OS	ROM Memory	RAM Memory	Type of Processes
Linux Kernel 2.6	4MB	1MB	Threads
TinyOS v2	3.4 KB	336 Bytes	Tasks, commands, events
Contiki	3.8 KB	230 Bytes	Protothreads
MantisOS	14 KB	500 Bytes	Threads
FreeRTOS	6 KB	236 Bytes	Tasks, co-routines

Our EBM has been developed as an independent component and then integrated in FreeRTOS, since this RTOS lacks of an advanced power management strategy.

5.3 RTOS module architecture

The proposed EBM is constituted by three independent components: the core, which include the power manager; the hooks that are functions used to read data from energy sensors (e.g. battery voltage, current sensor on energy harvester); and the Hardware Abstraction Libraries (HAL) used to interface the specific hardware. Detailed description and interaction of the components follows.

5.3.1 Core Components

The EBM core component represents the core of the whole infrastructure. The only application entry-point is called the engine; it performs coordination of sub-components and runs the decisional algorithm that determines how output sub-system gets modified. All the other components cooperate together without interacting with the user application. The EBM core components are the following:

- **Engine:** The EBM main loop coordinates other sub-components and performs analysis of stored data in order to run the decisional algorithm that dictates how to act on platform subsystems. It receives inputs from external sensors (such as battery, temperature, accelerometers etc) and decides how to manage registered outputs, in order to provide the best energy performing configuration of the platform. User applications will interact

only with this component, which is made asynchronous by means of a software timer and can be implemented as a separate thread.

- **Logger:** Logger stores and retrieves data registered from external sensors, providing a standardized interface to the engine in order to abstract from the underlying memory technology. It implements a circular buffer or a more complex data storage algorithm, and interacts with a hardware specific backend in order to have data stored in volatile memory, FLASH memory or external storages (even remote).

- **Input Manager:** Handles the input sensors controlled by the engine, managing setup and data sampling operations. The user application can register hooks to the input manager in order to provide a set of functions necessary to properly interact with the hardware (DMA setup, peripheral configuration, data sampling operations etc.).

- **Output Manager:** Handles the registered outputs controlled by the engine, such as General Purpose Outputs, CPU frequency scaling and other output devices. As per input manager, it requires the user application to register proper hooks to properly manage the actual hardware components.

- **Hooks:** Functions defined by application developer (or pre-shipped with the EBM) used to interface the engine with external and internal input sensors and platform dependent outputs. These functions will be registered to the engine by means of input or output manager facilities. Their main function is to separate the logic of the operations from the actual mechanisms needed to setup and interact with the underlying platform details.

- **Libraries:** Hardware Abstraction Libraries (HAL). All the platform dependent code resides here. As any other HAL library, it provides a common programming interface that makes the EBM easily portable between different platforms. Both the core components and the provided hooks have to use the HAL programming interface.

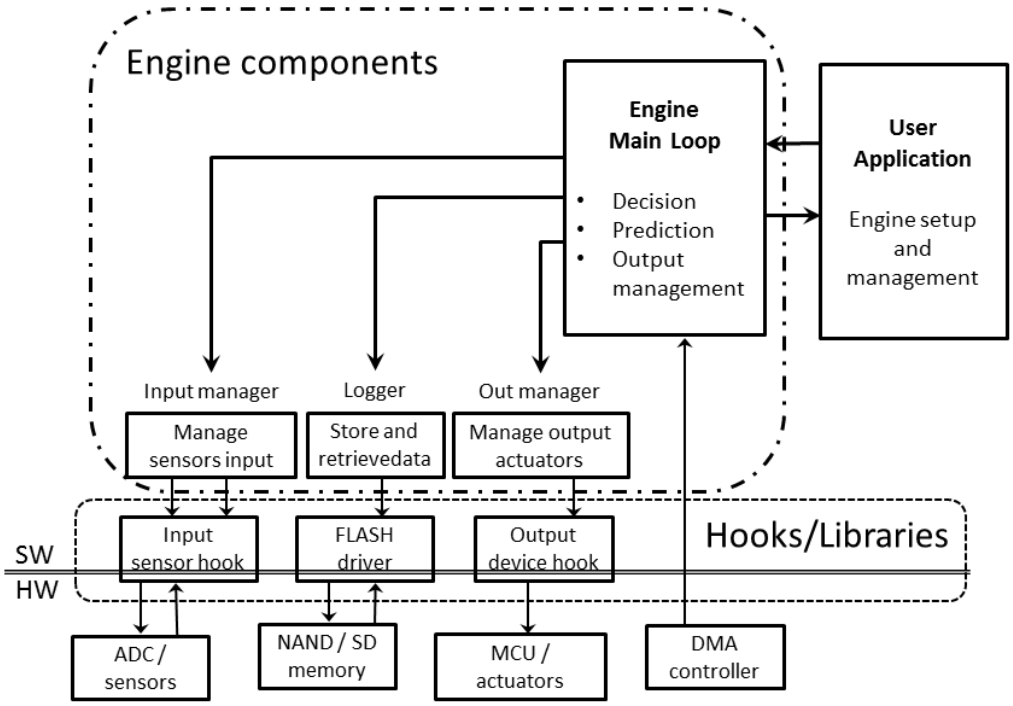


Figure 5.1: EMB Architectural overview. The engine components interact with hardware by means of libraries and application provided hooks. Hooks themselves can rely upon hardware abstraction libraries.

5.3.2 Components interaction:

The engine presents itself to the application as a structure of pointers to functions (engine objects), and the application employs this single interface to interact with all the EBM components.

This design admits the application to decide at run-time which engine to use and to dynamically switch to more or less aggressive policies. Hooks for the engine to handle inputs and outputs are registered to the EBM by means of the engine interface, by simply providing specific structures with pointers to hooks and libraries functions.

Once an input or an output is registered to the EBM, input or output managers will use these structures to interface with the hardware. Which function has to be called by the managers is determined by the engine main loop that will decide, after performing an analysis of the stored data, the way the subsystem has to be modified.

The interaction diagram presented in figure 5.2 illustrates how a DMA-operated input behaves and how data transfer is performed by the EBM.

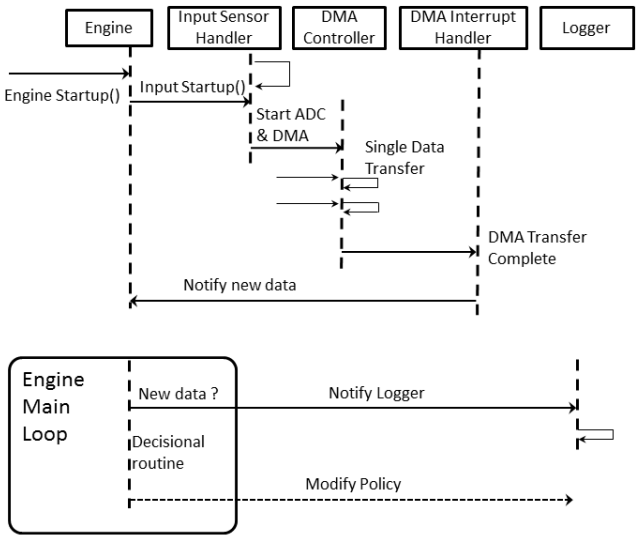


Figure 5.2: DMA input and data transfer for EBM

Once the engine has been started, it setups all the registered sensors, which starts producing data that are transferred in RAM memory through the DMA engine. Once the desired number of samples has been produced, the engine is notified about the availability of new samples. Those samples are then notified to the logger that stored them in FLASH memory; they can be read by the Engine main loop for Historical data analysis. The presence of new samples triggers a decisional loop used to manage outputs.

5.3.3 Engine integration in FreeRtos.

The describe architecture of the middleware can be easily integrated in a RTOS like FreeRTOS, since EBM can be implemented as a single task. EBM task should be able to read input from Hook functions (to be aware of system energy availability, or system behavior). The core in EBM task then take decision on the power management policy. A typical output of EBM task is the configuration of the IDLE task. Such task is called by FreeRTOS scheduler when there

is no other task to execute and it is usually used to put the system in low power mode. EBM core can decide which low power mode to use, which component to turn off or on, and control them in the IDLE task or setting parameters on applications tasks.

5.4 FreeRTOS Implementation

Along this section we describe the implementation of an example application, based on the EXLs1 platform. The case proposed can be common to typical Body Area Networks applications.

This application will serve as a validation of the flexibility of the engine mechanisms and to test how different energy management policies can be easily implemented

The basic tasks of the application chosen can be summarized to be the following:

- Sensor Reading: the node has to periodically read the external sensor set to collect data that can be sent or stored after being elaborated or in a raw format.
- Communications: the node must accept external commands in order to modify its behavior. Commands are sent from external applications using the Bluetooth interface.
- Visual Feedback: the installed LEDs are used to provide a visual feedback to the user about the operating mode and the battery level.
- Data Streaming: data collected from external sensors have to be stored or sent outside using the Bluetooth interface.

These basic operations can be encoded in single tasks, built on the FreeRTOS mechanisms, without any specific knowledge about energy management.

In order to define energy aware operation modes, all EBM components have been updated to comply with the new architecture, based on FreeRTOS. This guarantees the separation and automation of data collection and background processing, performed to provide effective data

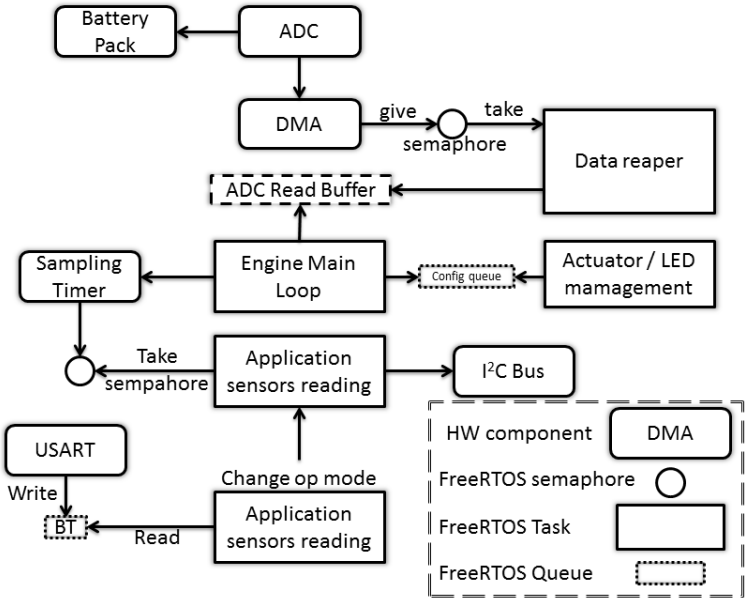


Figure 5.3: FreeRTOS system architecture for Cupid node implementation.

on which energy consumption can be reduced, in conjunction with system performance, when the available power in battery storage decreases under a critical level or when a more energy efficient operation mode can be selected to provide the desired functionalities.

Transition between different configurations can be triggered by critical thresholds established on battery power, or through environmental sensing, through an external component that specifies to the system which is the current use case for the Body Sensor Node.

sectionPolicy definition and performance analysis

5.4.1 Testbed implementation

To measure the performance of the new design, we have developed a modified version of the engine main loop task, which produces data continuously ranging from 4.0V to 3.4V. The current system voltage is decremented by 0.1V each 5 seconds and the system parameters are modified using simple thresholds, based on the battery pack voltage. Reducing the sampling frequency is the simplest strategy to increase the time the node spends sleeping, thus it is supposed to reduce the overall power consumption, reducing the system performance and accuracy [CFB13].

Another parameter we have simulated and tested is the ability of the firmware architecture to setup the system configuration in relation to the context. With context we refer for example to the knowledge of the motor activity of the user (sitting, walking, standing, cycling, running) or to other situational information that can influence the use of the device (e.g. if the system is thought to be active only indoor the information on where the user is can be used to appropriately change the configuration). This information can influence parameters such as the sampling rate, transceiver and other components active state, microcontroller frequency, etc.

To test the mentioned context-aware capabilities, a simulated software component (a use-case prediction box) dictates to the system which use case has currently been selected (this could be done in real word use cases through environmental sensing techniques, or through explicit user interactions). We define a set of policies that resume platform configurations that can be applied in different usage scenarios.

- Performance: all sensors are active; platform uses all the available sensors at maximum frequency.
- Balanced: Non vital sensors are turned off; CPU frequency can be decreased when non-critical tasks are run.
- Powersave: Only components specifically required by the running tasks are active, CPU speed is reduced and external devices, such as NAND storages, are deactivated.

Using this configuration, as the battery voltage decreases, the system is able to automatically decrease the absorbed energy through performance degradation techniques (reducing sampling rate), and then using the simulated prediction box to configure the platform with the more energy convenient policy (according to the user activity and context); the whole system is then able to configure itself without any intervention from the application developer. In order to quantify the reduction of energy consumed, we have experimentally measured the current absorbed by the node, while it is varying its working parameters following the data produced by the modified engine and the simulation of different use cases.

5.5 Performance analysis

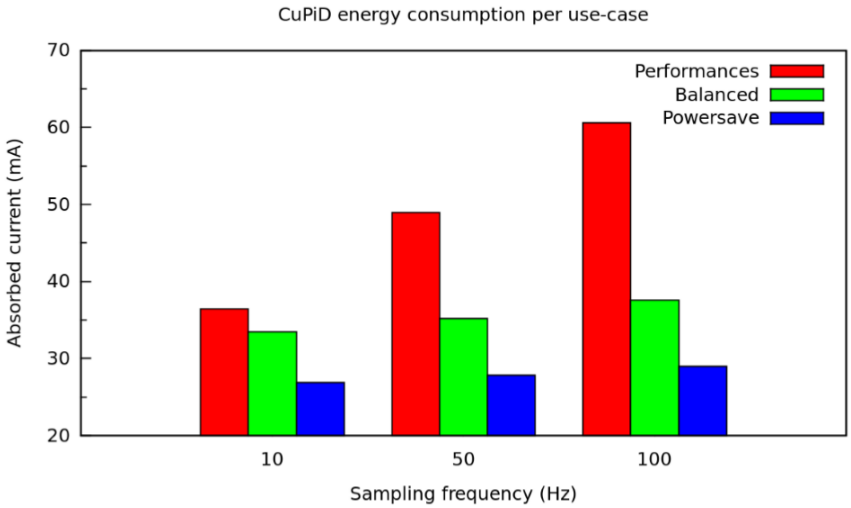


Figure 5.4: Power consumption for different use cases scenarios and battery levels, as the battery level decrease, sampling rate is reduced; use case activity control the policy (Performance, Balance, and Powersave).

The new firmware has been deployed on a test node, and its performance has been tested in different contexts. We have performed different runs, with 3 battery levels, and for each of them a different use case has been simulated. The graph in figure 5.4 summarizes the experimental results obtained simulating typical usage scenarios for the selected application.

Based on the constraints specified for each operating state, the EBM adopts the most energy convenient policy, which guarantees the lower power consumption without compromising the application needs.

As last result, we measured memory footprint of our EMB: we measured MCU stack occupation with EBM and without EBM, finding that our module occupies at most 120 bytes of RAM, which added to 230 bytes used by FreeRTOS kernel result in a RAM occupation of 350 bytes, comparable to the 350 bytes used by the core of TinyOs.

5.6 Conclusions

We have developed an EMB trying to keep architecture as simple as possible. This resulted in very little memory occupancy and in the possibility to integrate EMB in an existing operating system without altering the general design of the firmware, our module can be considered as an additional layer between the application and hardware layer. We have then showed a practical implementation on an inertial sensor node with FreeRTOS.

We simulated a use case scenario with different policies; the EBM was able to choose the most convenient policy respecting application constraints. It is possible to further develop this work adding more complexity to the core component, in order to have a more advanced logic to switch among power management policies.

Chapter 6

Activity aware power management for wearable sensors

6.1 Overview

In recent years, Body Area Networks (BAN) have emerged as an important class of distributed embedded systems capable of addressing a variety of challenging personal monitoring and assistance problems in a number of application domains, ranging from healthcare applications to fitness [L+11]. BAN technologies have recently been adopted to support the emerging personalized at home assistance, known collectively as wireless health. Wireless health merges data, knowledge, and wireless communication technologies to provide healthcare and medical services, such as prevention, diagnosis, and rehabilitation, outside of the traditional medical enterprise.

Such sensor systems have a high potential to significantly improve the quality of life for large segments of the population and enabling conceptually new types of applications. However, it is important to note that such systems are currently mainly used for research purposes and are not widely adopted; a key factor to improve usability and success of BAN are both device size and battery duration [Sar12]. The adoption of local dynamic power management (DPM) strategies to recognize and minimize the impact of wasteful and inefficient activities [BBDM00b] is an

approach that can be applied in BAN for node level power optimization.

Energy efficiency for BAN nodes has been addressed in at least two main classes of approaches in the literature: trough development of energy-efficient communication protocols (MAC, routing and self-organization protocols) that take the peculiarities of wireless sensor networks into account have been proposed [S+08]; and the adoption of local dynamic power management (DPM) strategies to recognize and minimize the impact of wasteful and inefficient activities within an individual node [BBDM00b].

However, it is important to note that a path to industrial realization has been more elusive than initially expected due to a variety of issues, including system and operational complexity, cost and energy sensitivity, semantic complexity, and the need for often revolutionary changes in consumer behaviour. Ever increasing opportunities in health care have thus motivated researchers to develop technologies that can be adopted in the medical fields to serve the growing demand of low cost and widely accessible health care services.

Among DPM techniques, an interesting approach consists in using context information to choose among the most effective power management policies, we will refer to this approach as Context Aware Power Management (CAPM). CAPM poses different research challenges when applied to BAN, since it must be able to detect user activity with good accuracy, levering on a limited number of sensors and energy budget [CFB14].

The CAPM is a popular approach for optimizing smartphone energy efficiency, where the presence of multiple sensors and the availability of significant computational resources allow the use of complex classifiers for activity recognition [M+13] [B+12].

Starting from similar premises, the work in [F+12] proposes an opportunistic classifier to optimize power consumption in a wearable movement monitoring system.

In our work, we compare different classifiers for activity recognition in search for the most energy efficient one to be applied within the CAPM strategy. It is thus important not only to evaluate activity classification accuracy, but also to evaluate end eventually reduce the energy overhead of the activity/context detection algorithm. Moreover, the platform that we target is a

wearable sensor node for continuous monitoring, which typically has limited resources in terms of memory and computation capabilities. Therefore, in our case, it is essential to assemble a lightweight activity recognition chain.

6.2 Related Works

Power management can be addressed at several levels, from hardware to firmware [7], optimizing single components and subsystems, up to application of distributed power optimization strategies of systems such as wireless sensor networks.

In typical BAN applications, healthcare in particular, the number of nodes is limited and there is often no possibility of placing redundant nodes, due to the need of enhancing wearability and usability. Furthermore, once the sensor node has been assembled or in case of commercial nodes use, the choice of the radio protocol is obliged and therefore there is no possibility to count on protocol optimization, but only to play with existing protocol configuration options. Given these considerations, power management of the BAN mainly overlaps with node-level power management.

At a very general level, several approaches can be exploited alone or combined to reduce power consumption at node-level. Two main techniques are duty cycling and data driven approaches [13]. Duty cycling is often based on sleep/wakeup scheduling algorithms and protocols. Dynamic power management (DPM) is a duty cycling based technique that decreases the energy consumption by selectively placing idle components into lower power states. The device needs to stay in the low-power state for long enough (the break even time) to recover the cost of transitioning in and out of the state [YY06].

While duty cycling techniques are not aware of data content, data-driven approaches can be a complementary way to save energy in a smart node [21]. Data sensing can impact on energy consumption (i) because the sensing subsystem is power hungry [21], (ii) because sampled data have strong correlation (spatial or temporal [22]), so that there is no need to communicate redundant information. CAPM combines a data driven approach trough an analysis of sensor's

data and DPM approach trough duty-cycling of unnecessary device components during the detected activity. Context awareness has been extensively studied, authors in [18] tailor the information such as location, time, season, temperature and so forth into several aspects of user's context. Due to the variety of available sensors, and the possibility to interact with different devices, mobile phones are very suitable devices for context recognition [3]. In some occasions sensors present on mobile phones have also been used to monitor user activity [4] [5]. The possibility to detect device's usage context also lead to algorithms capable to lever on such information to conserve energy on mobile phones [6]. A BAN is constituted by multiple sensors and is not practical to wear more than one smartphone. Moreover due to physical size constrain, BAN sensors have usually limited computational resources and many of the proposed algorithms for smartphones cannot be implemented on a resource-limited sensor node.

More related to BAN area, the work in [7] proposes an opportunistic classifier to optimize power consumption in a wearable movement monitoring system. In this case authors had the possibility to exploit features computed for the needs of the application. A novel system architecture has been proposed [8] for monitoring neuro-motor activity of Parkinson's disease patients, and for detecting epileptic seizures. Such system implement power optimization policies based on sensors computed features. Such approaches come usually with a computational cost that must be taken in consideration. Our work proposes node level optimization capable to extend battery life of BAN nodes. We propose a power optimization policy that relies on the possibility to switch off board components with no loss of relevant information from sensors. Context detection always comes with a cost; in this paper we also trade off between the usage of accurate classifiers and the need to minimize the energy cost of the classifier itself.

6.3 Activity recognition process

In the literature many different methods have been proposed for retrieving activity information from raw sensor data. The main steps can be summarized as preprocessing, segmentation, feature extraction, dimensionality reduction and classification [9] (figure 1). In this section we

present the most widely used algorithms and methods for each of these steps.

6.3.1 Filtering

Due to the nature of inertial sensors, the acquired sensor data should first pass a pre-processing phase. Almost always, high frequency noise in acceleration and gyroscope data needs to be removed. Therefore, non-linear, low-pass filters should be employed for removal of high-frequency noise [10]. Nowadays digital sensors integrate this type of filtering [11].

6.3.2 Segmentation

Retrieving important and useful information from continuous stream of sensor data is a difficult issue for continuous activity and motion recognition. Several segmentation methods for time series data have been proposed. We choose sliding window since it is simple and on-line algorithm. A sliding window algorithm starts with a small subsequence of time series and adds new data points until the number of data in the window is greater than the threshold which is defined by the user. This kind of algorithms work with a complexity of O(nL), where L is the average length of a window; the value of L also greatly influences next phase which is features extraction.

6.3.3 Features extraction

The purpose of feature extraction is to find the main characteristics of a data segment that accurately represent the original data. In other words, the transformation of large input data into a reduced representation set of features greatly simplifies classification work, giving advantages in terms of computation time. The feature vector includes important cues for distinguishing various activities and features are then used as inputs to classification algorithms [12]. Features can be grouped in the following types:

- Time-Domain: they are directly derived from a data segment. Most widely used features are: Mean, Variance, Std. Dev, RMS, Zero or mean crossing rate, Derivate, Peak counter and Amplitude, data range etc... This class of features have a computational complexity in the order of O(n)
- Frequency domain: Frequency-domain features focus on the periodic structure of the signal and often require the computation of Fourier Transform, among them are [14]: Discrete FFT Coefficients, Spectral Energy, Spectral Entropy Frequency, Range Power; to compute frequency domain features has a complexity of at least O(nlogn)
- Time-frequency domain: used to investigate both time and frequency characteristics of signals and they generally employ wavelet techniques.

6.3.4 Classification

The selected features that create feature sets are used as inputs for the classification and recognition methods, in literature is possible to find a variety of classifiers [15], among them we choose to compare the most common [12]:

- Nearest Neighbor: algorithms used for classification of activities based on the closest training examples in the feature space. These algorithms have a complexity of O(mnlog n) where m is the number of neighbor.
- Support Vector Machines (SVMs) are supervised learning methods used for classification. Complexity depends on the specific implementation and Kernel characteristic; it can vary between O(n2) and O(n3)
- Nave Bayes: a simple probabilistic classifier based on Bayes' theorem. (complexity O(n))
- Linear Discriminant: find a linear combination of features which best separates two or more classes of objects or events. (complexity O(n)).
- Linear Discriminant: find a linear combination of features which best separates two or more classes of objects or events. (complexity O(n)).

- Decision Tree: uses a tree-like graph of decisions. Each branch represents outcome of test, each leaf represents a class label. Complexity for the last 3 classifier is in the order of O(n).

6.4 Power management policies

The system on which power management policies is an inertial sensor module developed to assist Parkinson's disease patient in motor rehabilitation. Our purpose is to extent battery life of the nodes from a few hours to a whole day, guaranteeing same performance during rehabilitation exercises. We now briefly describe the main hardware components and software tasks in the BAN sensor node for this class of applications; this will give an overview of the main sources of power consumption.

- *Microcontroller (MCU):* is the brain of the sensor node, it interacts and communicates with other components. Most of the modern MCUs architectures (from 8 bit AVR to the more powerful Cortex M4) enable some of the internal components to be turned off when the system is in idle.

- *Radio Interface:* the communication system can use different protocols (Zigbee, Bluetooth, ANT) each of them features one or more power saving policy (e.g. Sniff mode for Bluetooth).

- *Sensors:* are the sources of the data, can be digital or analog, digital sensors usually have a low power mode.

Operations performed by a typical BAN node are: sensor sampling, data processing, data transmission.

Strategies to reduce energy consumption have been focused their scope on two different aspects: efficient MAC protocols and (DPM), developed to recognize and minimize the impact of wasteful and inefficient activities within an individual node.

DPM is usually implemented through a Power Manager (PM) or a scheduler, both with complete knowledge of the tasks that should be executed and the type of hardware resources they require. Our work focuses on the PM implementation. PM is an abstract layer that can detect user activity and adjust power saving policy accordingly.

Transition between sleep states of sensor node's components comes with a cost in terms of energy and delay. For simplicity we will now analyze the cost due to the transition from off to on state: if we call t_{off} the time that the hardware component spend in a dormant state and P_{off} the power consumption in such state. The transition time $t_{off \to on}$ from the dormant to the active state will thus have a power consumption of $P_{off \to on}$ whereas during on state power consumption is P_{on}. Transition to dormant state will be convenient only if the following condition is satisfied:

$$P_{on} \cdot (t_{off} + t_{off \to on}) \geq P_{off} \cdot t_{off} + P_{off \to on} + t_{off \to on} \tag{6.1}$$

Or in other word the transition to the dormant state is energetically convenient if the energy spent in the active state is greater that the energy spent in sleep state plus energy spent to wakeup the component. It is thus identifiable a minimum sleep time under which it is not convenient switch state:

$$t_{off} \geq \left(\frac{(P_{off} - P_{off \to on}) \cdot t_{off \to on}}{P_{off} - P_{on}} \right) \tag{6.2}$$

If the entire future of the system is known, it would be possible to define a DPM strategy capable to maximize energy efficiency carefully managing transition between low power states of the system components.

A typical scenario with known future is the need to periodically sample a set of sensors; in this case the PM defines a power management policy capable to obtain optimal power consumption [ref Patmos]. When the future of the system is unknown the PM can adopt different strategies [17], we choose to use a context aware strategy, in which the PM detect activity and chooses

appropriate policy.

6.5 System design

The context in which we are working is motor rehabilitation of patients thus we use inertial and magnetic sensor to detect limbs movements. Such data can be processed on the sensor node, logged in to internal memory, or transmitted through a radio device, according to clinical needs.

Among inertial sensors, accelerometer is the least power hungry, and accelerometer data are the most used for motor rehabilitation [15]. We thus designed a classifier capable to detect user activity using only accelerometer data. This classifier has been integrated in the PM. Moreover, the PM itself brings an overhead in terms of power consumption: while accelerometer data comes with no cost (they are needed for the application), features extraction and classification increases the energy spent during the processing phase. The system on which we tested or PM is a sensor node, constituted by the same hardware components of the CuPiD node, described in section 2.1.2 and listed here for reader's convenience:

- STM32 Cortex M3 Microcontroller.
- Bluetooth 2.0 radio module.
- tri-axial accelerometer, gyroscope and magnetometer.
- 1GB NAND FLASH memory.
- battery and circuitry needed to power and connect different components.

The set of features chosen to be analyzed together with computation clock cycles are shown in Table 6.1

From results in Table 6.1 it is evident that frequency domain features require much more energy (the computation time is one order of magnitude greater w.r.t. time domain features), for this reason we choose to find a classifier capable to discern activities using only time domain features.

Table 6.1: MCU clock cycles spent to compute features on a 500 elements array

Feature	**Clock cycles**
Mean	6636
Offset	12515
Variance	16166
RMS	10682
Max	10939
Min	10939
Mean Crossing Rate	Mean + 15986
FFT	226773
Spectral Energy	FFT+ 10940
ABS	26964

6.5.1 Classifiers comparison

In section III.D we have enumerated some of the most common classifiers, each of them has different complexity and accuracy; both this characteristics of the classifier can influence PM efficiency. A very complex classifier may require heavy resources in terms of memory utilization and computational capabilities, which might not be available in a low power embedded platform. Classification time, as well as features extraction time is affected by window size. We have compared different classifiers execution time with different window sizes.

Results of Figure 6.1 refer to time spent by the classifier to classify the same amount of features vectors; the window size is instead referred to the training phase. Classifiers trained with smaller window sizes may result in a more complex classifier structure (i.e. a higher number of Support Vectors for SVM classifier) resulting in a higher classification time. It is also evident from Figure 6.1 that our SVM classifier is not appropriate for our purposes, since it requires computation time not compatible with our real-time application, thus it will not be further analyzed.

6.5.2 Classifier accuracy

Our PM uses the classifier to choose the correct power management policy, each policy differs from the other in terms of energy consumption, sampling rate frequency and the use of the radio device. A misclassification will thus result in the adoption of an incorrect policy; this will result in higher power consumption or in a loss of significant data for a given activity.

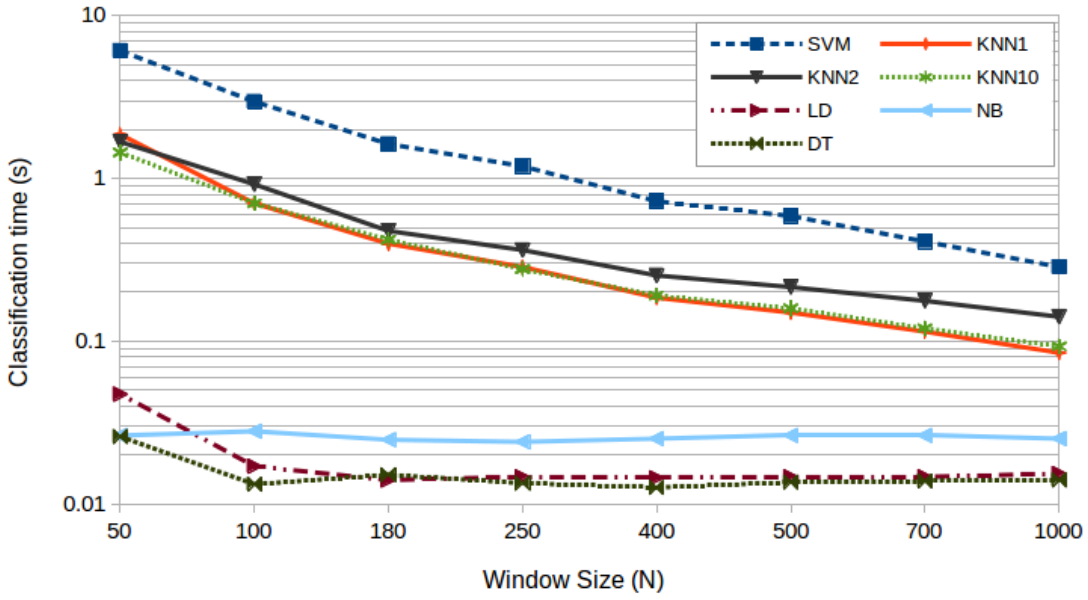


Figure 6.1: Execution time for different classifiers, compared with same training set (referred to an array of 1000 instances)

Table 6.2: Energy policy for each user activity

Activity	**Sampling Frequency**	**Active Sensors**	**Send/Log**	**Node Average Power Consumption(mA)**
Run	200	Acc + Gyr + Mag	Send	28.19
Walk	100	Acc + Gyr + Mag	Send	20.45
Stair Up	200	Acc + Gyr + Mag	Log	16.74
Stair Down	200	Acc + Gyr + Mag	Log	16.74
Bicycling	50	Acc + Gyr	Log	11.37
Sit	50	Acc	Log	3.10
Lie	30	Acc	Log	2.80

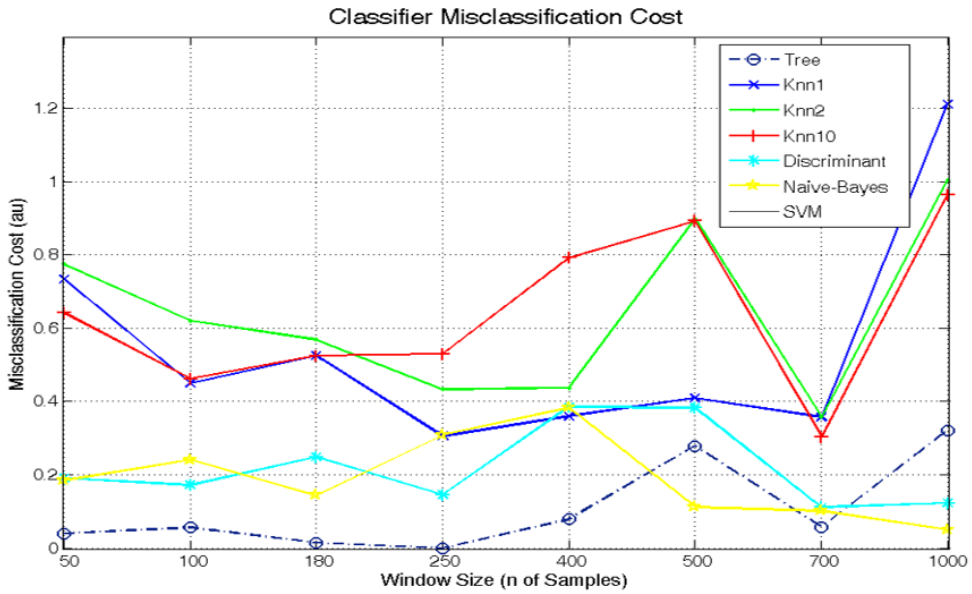


Figure 6.2: Misclassification costs for different classifiers expressed in penalty point

Choices made for policy implementation are dictated by the type of application for which the sensor is used: it is a system used for gait rehabilitation and patient monitoring, during rehabilitation sessions; data must be transmitted wirelessly to another device for further real-time analysis, whereas during other activities data can be stored onboard and analyzed offline by clinicians. This explains why radio is active only during run and walk activities, as the activity becomes less dynamic or clinically important, the sampling rate, or the set of active sensor is reduced accordingly. Sampling frequency of the sensor has been kept above 30Hz to correctly detect gait [10] and to avoid an increase in classification error.

To evaluate the cost of a misclassification we have given a penalty score proportional to difference of power consumption between correct and misclassified activity (e.g. if the classifier does not recognize the activity run correctly and interpret it as walk, a penalty proportional to 28.19mA 20.45mA will be assigned). Using this criterion we have evaluated penalty for the loss of data or higher power consumption due to an incorrect classification. It can be seen from Figure 6.2 that the tree classifier is favorable with smaller window size. This result, together with measurements presented in Figure 6.1 lead our choice to the **classification tree** as best classifier for context-aware power management in our application scenario.

6.5.3 Classifier characteristics

Classification trees evaluate responses feature vector. To assign a response, the tree is followed from the root (beginning) node down to a leaf node. The leaf node contains the response. Each step in a classification involves checking the value of one variable. The implementation in an embedded system can be very simple: once features have been computed, a series of nested if then else instruction checks for the value of some features, till a leaf is reached. The structure of the tree and the value of each feature used to select a branch are determined during the training phase. In our case the number of nodes (decision points) in the tree was dependent on the window size: a smaller window on which the features are computed resulted in a more complex tree structure.

During training, all features are used by the classifier, but after classification not necessarily all features are used for classification, once the model has been built it is thus possible to instruct the PM to compute only a subset of features.

Another advantage of using classification tree, is that it is possible to prune it, removing branches of leafs with lower importance; it is in fact possible to evaluate the error due to pruning of certain branches and eventually accept an error to reduce classifier complexity. In our case this was not necessary since time spent by the classifier resulted to be negligible.

Classification accuracy varies from subject to subject and if the classifier is trained on one subject data or multiple subjects. Using 10 fold cross validation we measured an accuracy of 98% using one subject data and an accuracy of 90% when we trained the classifier using data from 3 different subjects. It is possible to train the classifier on each subject in medical applications; where a doctor in the first phase of the therapy supervises patient's activities.

6.6 Experimental results

Without power management policy power consumption of the node is 28mA. Using our PM we have been able to extend battery life of a sensor node to more than a whole day in a scenario

Table 6.3: Node power consumption and typical daily activity duration

Activity	**Hrs. per day**	**Total Power consumption per day (mAh)**
Walk	3	61.38
Run	0.5	14.10
Stair Up	0.25	4.19
Stair Down	0.25	4.19
Bicycling	0.5	5.69
Sit	11.5	35.63
Lie	8	22.33

where the node is used to monitor user gait and running, during those activities sampling rate is kept high (Table 6.3) and data is sent to an external device to assess performance. We have evaluated power consumption during one day of typical activity of a subject. From results shown in Table 3, we have that a battery of 160mAh would least more than one day, compared to the 5 hours when no policy is adopted and thus maximum sampling frequency must be used.

The flexibility of the system makes it possible to use it also for different applications scenarios. An example is the use of the inertial sensors in an assistive system. Sensors are being used to assist patients during exercises. This is the case of the CuPiD Project, where people with Parkinson's disease perform gait and posture rehabilitation for one hour a day. The patient is asked to perform outdoor walking session wearing inertial sensors. A smartphone connected to sensors tutors the patient trough an audio feedback.

In this scenario we leveraged on the possibility of the classifier to detect user activity. This information can be used to choose the appropriate power management policy and reduce power consumption during activities different than walking. Context information derived by our CAPM can also be used to trigger the execution of the exercise by the patient. Once the walk activity has been detected, a message is sent to the patient to ask whether he wants to start rehabilitation gait session. When 1 hour of exercises has been performed, no further requests will be sent. The output of the classifier thus can be used not only for power management purposes, but also for application needs. Using the proposed power management policies is possible to wear the sensors all day and kept them responsive; our approach reduced power consumption to 80mAh per day, extending battery life to two days; moreover the output of the

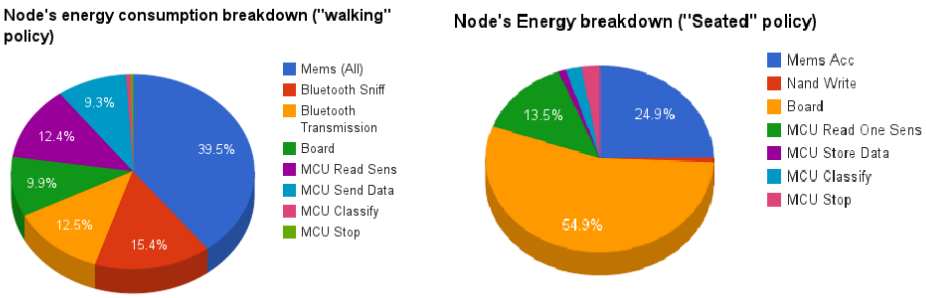


Figure 6.3: Node's power consumption breakdown

classifier can be used to suggest the patient to start exercises.

Our power management layer has been applied to a sensor node that was designed for continuous streaming at high sampling rates (100-200Hz), greatly increasing its flexibility and battery duration. The impact of our layer in terms of resources can be considered negligible: in Figure (down) it is possible to notice that for walk policy, power consumption of different components is balanced, with classifier accounting for only 0.6% of power consumption. Whereas when a low power policy is adopted much of the power is used by the board itself (power regulators and the components that cannot be switched off) evidencing that the system is not designed to operate in ultra-low power mode.

6.7 Decision Tree benefits

From results of the previous section it is clear that Decision Tree is a good candidate for integration in an activity based PM. Hence we focus on DT and show techniques to further reduce energy consumption with respect to other classifiers. Optimization is performed on two different scenarios: single subject and multiple subjects.

6.7.1 Dataset

Once the classifier has been defined, an external dataset has been used to test it on a relatively large number of subjects. We decide to use dataset collected in [A+12]. This dataset is a

collection of inertial data (3 axis accelerometer and gyroscope) collected from 30 subjects using a smartphone sampling data at 50Hz. This dataset already contains computed features, but we used only raw data to test classification tree to be embedded in our PM.

6.7.2 Single subject

Intra-subject classification requires a less complex tree structure and delivers better accuracy w.r.t. activity recognition applied to the variability brought by multiple subjects. The choice of decision tree as classification method enables our PM to save more energy with respect to other classifiers. The features computation requires indeed a non-negligible amount of energy. We optimized this process using the following approach: during the training phase, all the features are extracted and provided in input to the training algorithm; once the tree structure has been defined, it is possible that some of the features are not used. In our case we trained our classifier using all the 27 features listed in Table. 6.1 (one feature per each of the 3 accelerometer axis, except ABS that was applied to FFT). To avoid the use of computational intensive features, the algorithm tries to classify data using features with less computational effort and if not able to achieve good accuracy (above 99 % on training data) more features are taken into account. We computed the number of features actually used by the tree generated for each subject and we measured the classification error using 5 fold cross validation. Results are shown in Fig. 6.4 where using box plot we show number of features used for each patient and average number of features.

When the segmentation window is large (above 400 samples) it is possible to classify activity using very few features (one to three), but error rate of the classifier is increased (Fig. 6.4). The case with best accuracy (97%) has been achieved using only one seventh of the training features (4 features out of 27). The features extraction algorithm we use results 7 times more efficient than the case when other classifiers are employed, since in our case it is sufficient to compute only one seventh of all the features. Moreover, as we privileged low-complexity features, none of the frequency domain features was used by our classification tree, bringing further MCU clock cycles saving (according to Table 6.1).

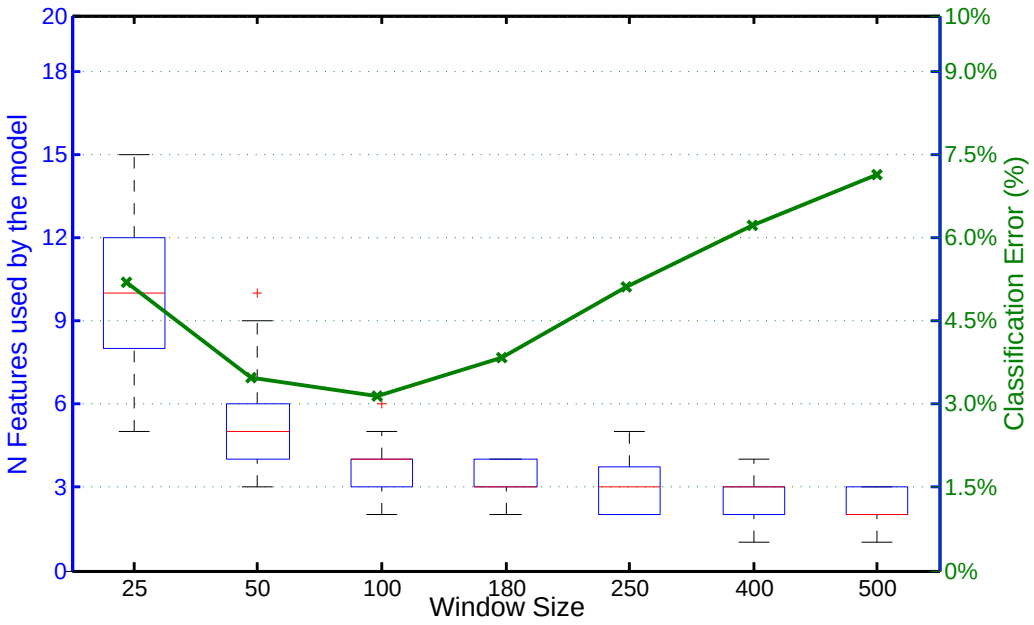


Figure 6.4: Analysis of features needed by decision tree for each patient (Box Plot) and classification error (green line)

6.7.3 Multiple Subjects

When used on multiple subjects, the decision tree usually has a more complex structure and it often uses much more features to correctly classify data. In our test the average number of used features was 19 out of 27, with a maximum value of 21, thus it is possible to avoid the use of frequency domain features that requires much more clock cycles than time domain features. When trained with data from different subjects, decision trees are affected by over-fitting problem. To mitigate this issue, we imposed that each leaf of the trees must have a minimum number of occurrences during the training phase. The minimum number of occurrences was chosen to be 1/900 of the total number of training features (30 subjects x 6 activities x 5 leaves = 900 nodes). The complexity of decision tree has also impact on PM power consumption, according to [Knu71] sorting algorithm has a complexity in the order of $\Omega(n \log_2 n)$ where n is the number of nodes. To further reduce the number of nodes, we applied a pruning algorithm to the tree, and verified how classification error is affected by pruning.

As shown in Fig. 6.5 it is possible to prune decision tree up to 70% with no increase in error rate and great savings in terms of clock cycles used by PM. We computed that given the complexity of decision trees for multiple patients (from 900 to 100 nodes), the classification phase requires

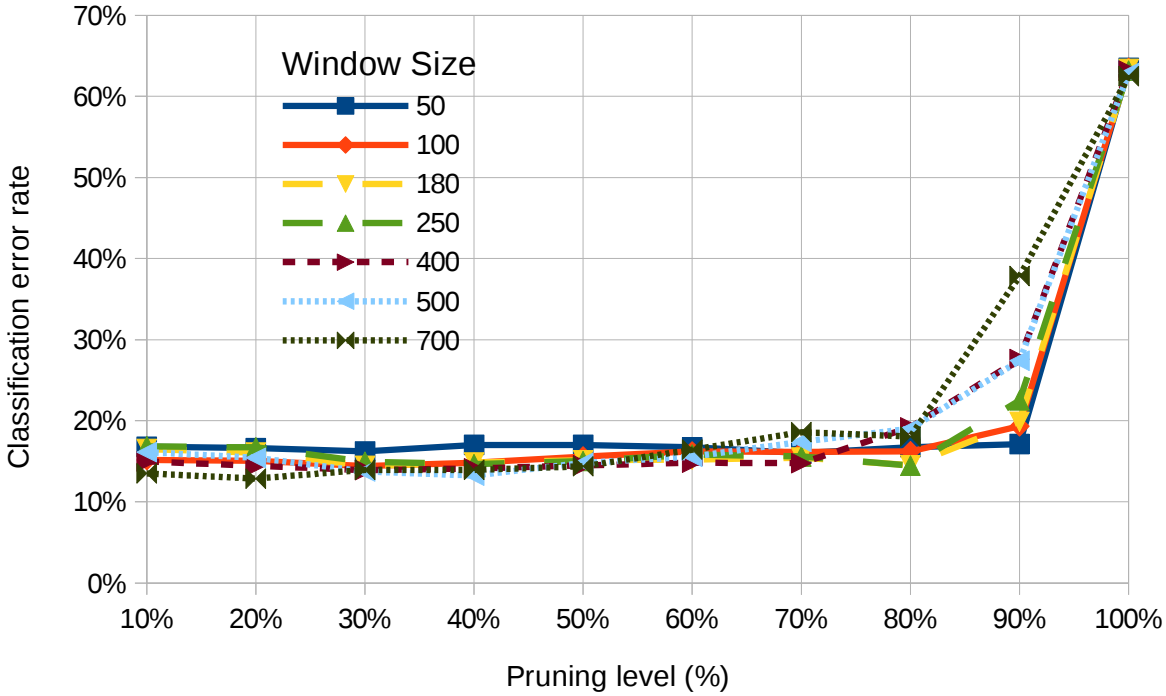


Figure 6.5: Classification error rate on multiple subject with different pruning levels

only 25% of the time needed if no pruning algorithm is applied. Another aspect that must be considered is that during the transition period between two activities, some miss-classification may result. Although, according to [N+03] there are around 100 activity transition in a 12h period, if we chose a 100 as sliding window size we may have miss-classification for 333s in a 43200s period (0.78%).

6.8 Experimental result

To evaluate the effectiveness of the proposed method our PM has been tested on a sensor node and power consumption has been measured. The reference platform is constituted by the following components:

- *Microcontroller*: a Cortex M3 (STM32L151VC) low-power 32-bit microcontroller capable to run up to 32MHz
- *Radio Interface*: Bluetooth Low Energy has been chosen due to compatibility with smartphones and low power consumption. The chip used is TI CC2541

- *Sensors*: Invensense MPU 9150 is an IMU, which embeds in a single chip triaxial accelerometer, magnetometer and gyroscope. The power consumption of the chip is affected by sampling rate and active sensors.

- *NAND Flash Memory*: a Micron 8Gbit memory stores sensor data. The size of the memory would allow to save up to 5 days of data sampled at 100Hz

The power management policies adopted for the different activities are shown in table 6.4

Table 6.4: Energy policy for each user activity

Activity	**Sampling Freq (Hz)**	**Active Sensors**	**Radio**
Run	200	Acc + Gyr + Mag	Send
Walk	100	Acc + Gyr + Mag	Send
Stair Up	200	Acc + Gyr + Mag	Off (Log)
Stair Down	200	Acc + Gyr + Mag	Off (Log)
Bicycling	50	Acc + Gyr	Off (Log)
Sit	50	Acc	Off (Log)
Lie	30	Acc	Off (Log)

In Fig. 6.6 we report the power consumption of the platform without optimization (dashed lines) and the percentage of microcontroller clock cycles utilization by the PM. It is possible to notice that our optimization reduced MCU occupancy required by PM task, optimisation can be quantified in 70% MCU clock cycles reduction for single subject and 50% reduction for multiple subjects scenario. The energy consumption of the system in multiple subjects scenario is in average 12% higher, mainly due to misclassification errors (i.e if activity "lie" is identified as "walk" a penalty equal to the difference of power consumption of the two state is given). Measurements have been performed using the "single subject" scenario, due to the difficulty to recruit a large number of subjects. Results of Figure 6.6 completely agree with simulation results of section 6.7.2 and shows the efficacy of the proposed PM optimization.

6.9 Conclusion

In this work we have used CAPM concept to build a Power Manager capable to reduce energy consumption of BAN sensor nodes used for subject continuous monitoring. In our work we com-

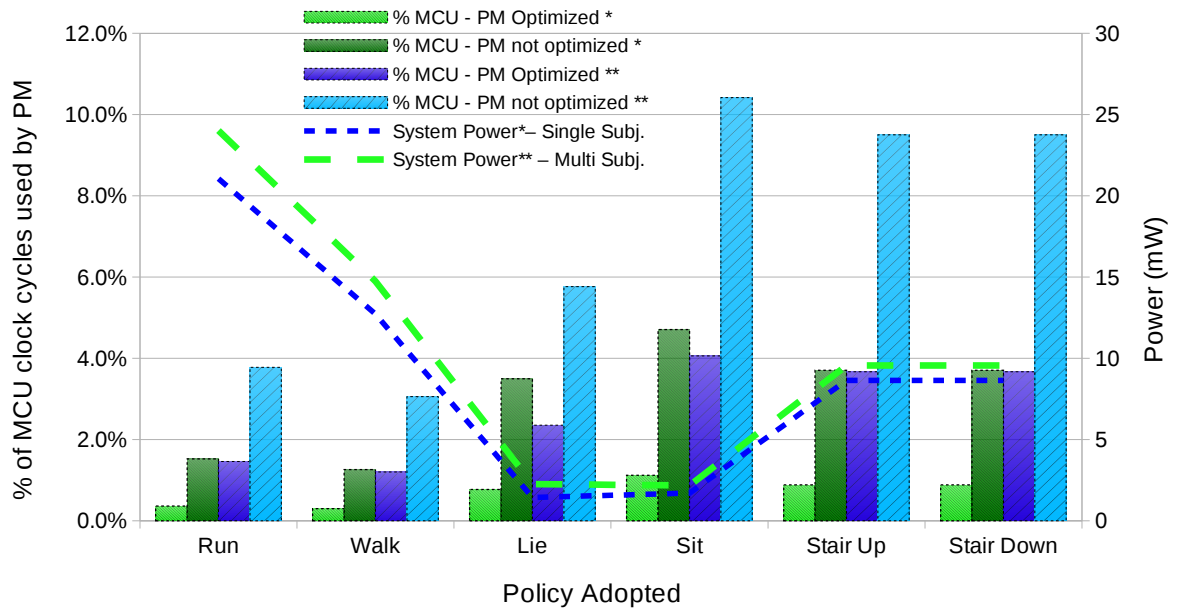


Figure 6.6: Representation of test platform power consumption and percentage of MCU occupancy by the PM with and without optimization. Classifiers has been trained for (*) single subject and (**) multiple subjects scenario

pared different activity classification approaches, among them the use of Decision Tree seems the most suitable; it has same accuracy as other classifiers, but requires less computational effort. We then moved one step further introducing a selective pruning algorithm that is capable to reduce the number of computed features or simplify the tree structure. These improvements resulted in a lightweight power manager that does not impact significantly on platform computational capabilities but increase battery life thanks to its policies. Measurements on a typical platform show that it is possible to achieve energy savings up to 70%.

Chapter 7

Networking operations for activity aware sensor nodes

7.1 Overview

Together with wearables and smartphones, Wireless Body Area Networks (WBANs) are demonstrating to be useful in improving health-related habits and reaching fitness goals. New devices like activity trackers have appeared in myriad, bundled with appealing apps motivating people to care for their health. In motor rehabilitation, information from different body segments is often required, thus WBANs can play an important role, since each single node can be placed in the point of interest of the body [CFM+13]. Together with all wearables, WBANs share the need to prolong their lifetime as long as possible, to enhance usability, maintenance and mobility, while keeping the form factor reduced. WBANs have unique challenges in energy efficiency because, differently from wireless sensor networks (WSN), they often have to provide continuous data streaming. In particular, in rehabilitation kind of applications, the WBAN is also asked for real time processing and feedback provisioning.Furthermore, the application imposes strict requirements in quality of service, i.e. accuracy of the measurements, timing, avoidance of data losses, etc.

Power management for WBAN has been proposed in several ways and shares challenges typical

of networked embedded system (i.e. WSN). Software/firmware approaches focus on nodelevel power management such as dynamic voltage and frequency scaling, power-aware scheduling, dynamic power management by using low-power modes, etc. Alternative approaches are based on hardware choices such as use of low-power radios or custom ultra-low power hardware components or subsystems [UK12]. The best benefits can be reached by combining hardware and software techniques and this is the approach followed in the present paper.

To address power management challenges in WBAN, the development of activity aware power management approaches that can be generalized to other contextual information (from which the name CAPM context aware power management). Depending on user activities andapplication goals, the device switches between different power management policies corresponding to different setup of the unit components (i.e. sensor sampling rate, microcontroller configuration or transceiver power states, etc.). In another work [CFB14], CAPM is tested at node level. However, the knowledge of the context can influence more than one node in a WBAN and in particular, in this work, we focus on the possibilities offered in a multi-node scenario, augmenting our previous work. In this work a software strategy based on CAPM is combined with the availability of an ultra-low power hardware component, the radio trigger [KKM$^+$14] enabling to selectively wake-up nodes at the best convenience. The use of the wake up radio in WBAN has been proposed in previous works [MP11], but the combination with an activity aware power manager in a rehabilitation scenario has not yet been attempted.
In this chapter we therefore analyze the benefits of using the CAPM approach augmented with a wake-up radio in a gait rehabilitation scenario, where the user wears a WBAN of 3 inertial nodes. Two nodes on the feet are controlled by a third node on the trunk via the wake-up radio trigger, which is activated by the CAPM. This method can benefit from the fact that in the selected rehabilitation scenario nodes can be activated only when the user is walking for periods longer than one minute and remain in a low power state for the rest of the time. We compared the power consumption in absence of power management with the use of the CAPM alone and with the combination of CAPM and radio trigger. We demonstrate that we can augment the lifetime of the WBAN up to 20 times for a typical usage scenario.

7.2 Proposed Scenario

The purpose of the system is to assist elderly people in gait rehabilitation through a closed loop feedback. To perform such task, it is necessary to have one wearable sensor node on each foot and a central node for coordination, data collection and feedback restitution. Our system is thus constituted by the following components: one Master Node (MN), two Slave Nodes (SN) capable to collect inertial data and perform basic data processing to extract gait features.
The slave nodes are worn on the shoes whereas the master node is worn on the chest of the subject. Due to its position on the chest, the MN is able to compute subjects trunk posture and to detect subject activity. The algorithms used to compute gait features are derived from OpenShoe Inertial Navigation System [NSHH12]. Thanks to these algorithms, it is possible to extract and to compute step features like: step duration, step length, step velocity, foot elevation, etc The system can work in two modes: SN collect sensory data and extract gait features that are sent to MN where results from both feet are combined to generate a feedback. SNs can also send calibrated data and thus perform most of the computation on the MN. This solution is useful not only to offload SN MCU, but also to combine sensorial information from both feet and improve algorithms accuracy. While the MN has to be active to monitor Subjects activity, SN can be turned off when the person is not walking, thus we implemented a context detector on the MN to detect when the subject is walking and a radio wake up to send wake signal to SN also when they are in deep sleep condition. The system is constituted by existing sensor nodes (Section 2.1.2), and a radio wakeup connected to them trough I/O pins.

7.2.1 Sensor Nodes:

Each sensor node is equipped with a tri-axial accelerometer, a magnetometer and a gyroscope, a Bluetooth 2.0 radio transceiver, a microcontroller unit, an external FLASH memory; the node includes a nano-power wake up radio, which is able to detect wake up messages and generate interrupts for the microcontroller. Due to the presence of the wake up radio, the Bluetooth radio can be completely switched off when the communication is not needed. In this state the

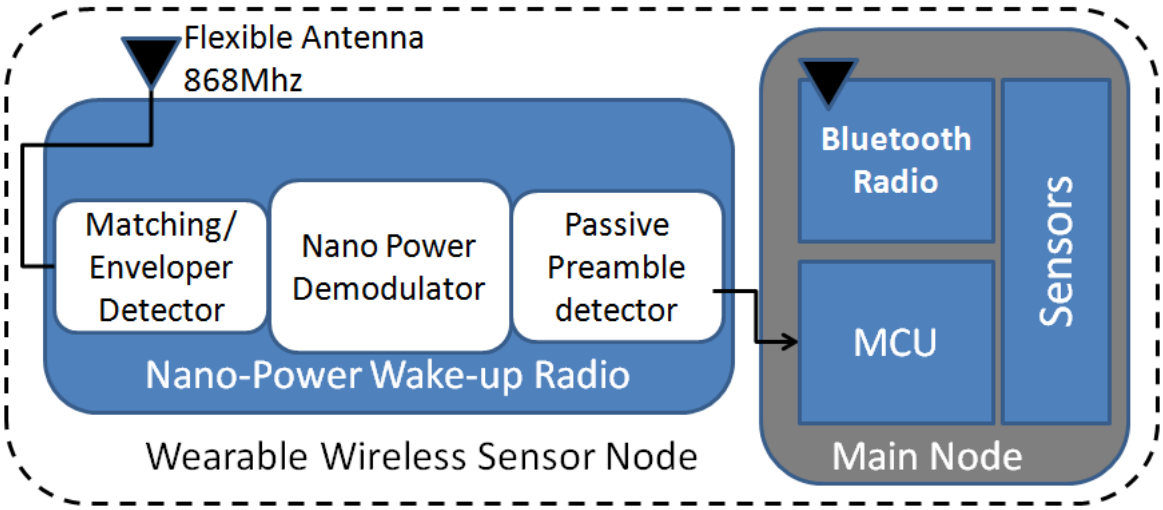


Figure 7.1: Wearable Wireless Sensor Node Architecture.

node is in listening mode only on the wake up radio channel. In this section, we will present the hardware architecture of the main node and the wake up radio. The two subsystems will comprise together the whole slave node.

The differences between the slave nodes and the master nodes are: Master node (MN) has a more powerful microcontroller (Cortex M4 instead of Cortex M3), the battery size of the MN is 5 times bigger than the slave nodes (1000mAh instead of 200mAh) and MN is equipped with radio wakeup transmitter whereas the slave nodes mount the receiver. Slave nodes have a lower battery capacity to reduce overall weight and size.

7.2.2 Nano Power Wake up Radio

As mentioned before, the main role of the wake up radio is the detection of radio messages when the Bluetooth is switched off. The primary aim of the wake up radio subsystem, is the reduction of the overall power consumption due to the idle listening of the Bluetooth. In fact, usually the Bluetooth radio wastes energy in Sniff low power mode. The wake up radio can completely remove the power consumption of the radio due to that mode. Thus, to achieve a real benefit, the power consumption of the wake up radio must be order of magnitude smaller than the power consumption of the main radio.

For this reason, the first goal in the design of the wake up radio is the very low power consump-

tion usually in the range of microwatt or better nanowatt. A second goal is to allow the data recognition to reduce the number of false wake ups and incorporate an addressing mechanism to wake up only the node intended to be waked up.

Furthermore, the design of the wake up radio requires careful consideration of design issues in RF, analogue electronics, and digital and system design to carefully evaluate the following trade-offs:

- wake-up range vs. energy consumption;
- wake-up range vs. delay;
- same-band vs. different-band wake-up radio;
- Addressing or without addressing.

Moreover, in [GS04] Gu and Stankovic, who can be considered pioneers of the wake up radio concept in wireless sensor networks, presented the following design goals for WURs, which are still valid:

- Low power consumption;
- High sensitivity;
- Resistance to interference;
- Fast wake-up.

The whole architecture of the wake up radio implemented in this work is shown in Figure 7.2, and it can be considered as a very simple and ultra low power radio receiver. The modulation chosen for the messages is the On Off Keying (OOK) modulation due to the low power consumption of the radio demodulator. In fact, as Figure 7.2 shows, beyond the matching network, the RF receiver is comprised by a passive envelope detector and a nano-power comparator with its reference generator (an RC filter in the V- pin).

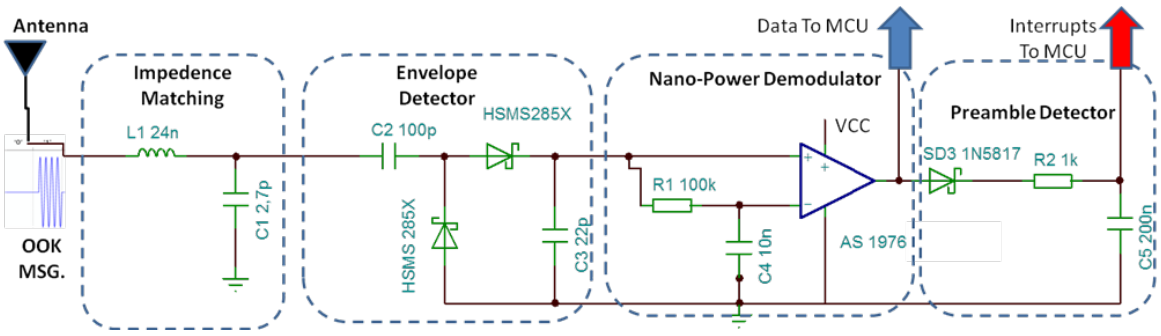


Figure 7.2: Wake up radio diagram.

The matching network has been tuned at 868MHz and it has the important role to allow the maximum energy transfer between the antenna (50ohm) and the rest of the circuit. Furthermore, the matching is able to filter also the interferences due to the other frequency (i.e. 2.4GHz of Bluetooth). The envelope detector continues the radio front end; it has been designed with the Avago HMSM 285X Schottky diodes optimized to achieve -56dBm of sensitivity at 868Mhz. These diodes are the ones available in market with the lowest sensitivity. After the envelope detector the signal is rectified and a comparator is needed to generate the data and the interrupt signals. For this work, we used the AS1976 from Austrian Microelectronics, which is able to guarantee -42dBm of sensitivity with only 200nA of current consumption.

With this sensitivity we were able to cover an range around 5meters with a 1.8dBi flexible antenna worn on the body, which meets our needs. For the scope of our application in fact, the MN with radio transmitter and the SN node with wake up radio are planned to be placed not farther than 1.5 meters (distance between the feet and the trunk), a distance much smaller than the maximum allowed by our wakeup technology.

The power consumption results to be order of magnitude lower than the sleep current of the Bluetooth modules. Latency measured for the first interrupt to the microcontroller can be generated in 5seconds when the first bit '1' of the message is detected. To avoid false positive a preamble detection is needed to generate an interrupt only when a long message of following '1' is detected. The preamble Detector has been designed using only a diode and a RC filter. The RC filter can be tuned to detect different length of the preamble. The length of the preamble affects the reactivity of the wake up so that the trade off between latency and false positive

has to be selected. In this work, we selected four consecutive bits as preamble. The bit rate selected was 10Kbps so that the overall latency to generate the signal was 400s. This time has been selected according to the application algorithm, which does not need higher latency.

7.3 power consumption using the WUR

Measurements have been performed on the two separate components of the system (sensor nodes and radio wake up) and then put together to simulate a single device. The functionality of the full system has been tested connecting a radio wake up to sensor nodes trough general purpose IO of the sensor node. We measured power consumption for three different scenarios:

1. Power consumption of the nodes without wake up radio and without power manager: The system was designed to be used only during specific training sessions whose duration can be as long as few hours. Nodes were thus not equipped with a context power manager and a wake up radio. In this first case, the nodes sample data at the maximum frequency and are active all the time. In particular, we analyzed two different configurations:

    - The SNs continuously stream raw data to the Master node where most of the processing occurs. This configuration reduces MCU load of each SN but increases the radio power consumption. As result, we measured a power consumption of 64.8mW for the single SN and 59.4mW for the Master Node.
    - The processing occurs on the SN, and only step features are transmitted to the MN. This configuration reduces data transmitted, but charges the MCU. In this second case, we measured a power consumption of 65.2mW for the SN and 26.3 mW for the MN.

    Since the Master node has a higher battery capacity, it is more convenient to stream data and let the processing occur on the MN. The latter configuration is also advantageous since it allows to combine data from both SNs and improve foot position accuracy [26].

2. Power Consumption without wake up radio, with Power manager When the MN is used to perform also activity recognition, the SN does not need to be active all the time, and they can use different sampling frequencies according to the users activity. In any case, the SNs need to have the Bluetooth connection active, in sniff mode, with a minimum power consumption of 3.5mA.

   During the Not walking scenario, the SN do not sample any data, the MCU is in low power mode, the Bluetooth link is active but in low power mode, so that the MN can send a signal to wakeup the SN. MN instead performs only activity classification. 3) Power consumption with the radio trigger and Context Power Management: When the subject is not walking, the MN and the SNs consume a relevant amount of power. This is mainly due to the fact that the Bluetooth in low power mode (sniff), also without transmitting data, needs an average value of 3.5mA to keep the connection alive.

   Furthermore, the SNs need to keep the serial peripheral active to read wakeup signal from the Bluetooth module. If radio trigger is used instead, the MCU can be set in deep sleep mode, and Bluetooth can be completely switched off. In this scenario power consumption is shown in Table 7.1:

Table 7.1: Power consumption for different policies

Activity	**Sensor Sampling rate**	**SN power**	**MN power**
Run / stair	200Hz	87.7 mW	99.3 mW
Walking	100Hz	64.8 mW	59.4 mW
Not walking CAPM	0Hz	15.8mW	18.5 mW
Not walking CAPM + Radio Wake up	0Hz	0.5 mW	5.8 mW

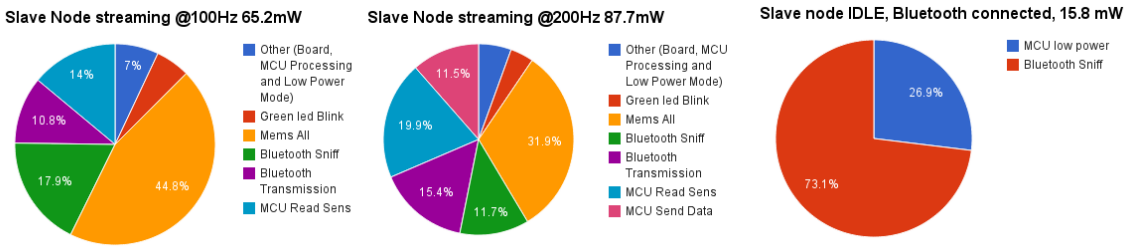


Figure 7.3: Slave node power consumption breakdown for different configurations.

Using the results shown in Table 7.1 we can thus estimate battery duration for a typical usage scenario, where a person runs for 15 min and walks for 1Hr every day: in this case, a 200mAh battery can last one entire week for the SN and up to 2 weeks for the MN.

7.3.1 Error introduced by Power Manager:

The CAPM and Radio Wakeup introduce a delay in detecting that the subject is walking. This delay can be easily quantified as the sum of: Sliding Window delay, computation delay, radio wakeup delay, slave nodes wakeup delay, Bluetooth connection delay. The most significant contribution is given by Sliding Window delay (1.66 sec) and the Bluetooth delay connection that requires 2.10.3 sec. This gives an average delay of 3.8 sec on average and we measured a maximum value of 4.5 sec. As result, the system is not able to detect the first 4-5 steps. However, this is not an issue for our application, since walks with duration less than one minute are not relevant for rehabilitation purposes and losing the first minute of a walk does not affect the overall evaluation and rehabilitation of a longer walk.

If the classifiers does not correctly detect the activity, two different cases can occur: the nodes consumes more power than necessary, and collect more data than needed; the node use less power, but lose relevant data. We tested the classifier with 4 subjects and computed the confusion matrix (CM). In the CM of Table 7.2 it is possible to locate 3 different areas: on the diagonal occurrences with correct sampling rate are present; the upper triangular section shows the occurrences where the system sampled data at a higher frequency than necessary and thus extra-power consumption occurs; the lower triangular section represents instead occurrences where sampling rate was lower than application requirements. We can further analyze occurrences where we have a data loss situation. Two cases can be outlined:

- data rate is reduced. This is the case when the person was running and the output of the classifier was walk. As result, the step estimation algorithms are less accurate, foot position estimation error grows from 1% to 5% each step [NSHH12].
- no data is collected. In this case the SN are switched off and gait events are loss. In our

dataset we had no such occurrences. It is in fact very unlike that the person is moving and no motion is detected.

Table 7.2: Confusion Matrix for classification results

Predictet \Actual	**Walk**	**RUN**	**Not Walk**
Walk	2558	105	14
Run	61	1881	0
Not Walk	0	0	2625

7.4 Conclusions

Power management in WBAN context can greatly improve usability and ease of wearable sensors. We tackled this topic combining a context power management software strategy with a radio wake up, which is a hardware component. Results show that for gait monitoring, this strategy is effective in the purpose of prolonging battery life from few hours (8hr) to an entire week.

This allows less frequent battery charges improving usability mainly for elderly people. We also showed that our strategy does not affect the quality of data received by the application.

Chapter 8

Low power dual core architecture for wearables

8.1 Overvirew

Wearable electronic devices exist as prototypes since few decades, and they remained as a market niche for many years. During last few years, wearable experienced an impressive market growth and gained lot of attention both from industry and academia.

Wearable sensors are now widely used for fitness, health-care and even military applications, an important task that wearable devices need to be able to perform is human activity recognition. For instance recognizing activities such as walking, running, climbing stairs can be useful to provide feedback to the caregiver about the patients behavior. In tactical scenarios, precise information on the soldiers activities along with their locations and health conditions, is highly beneficial for their performance and safety. Wearable devices can also be used to track user's physical activity an stimulate population toward better physical behavior and well being [DLC14]. Together with activity recognition, power efficiency is an important aspect of wearable devices, lower power consumption means smaller batteries, smaller devices and longer intervals between battery charging, leading to a more user friendly devices.

In this work we constructed a typical activity detection algorithm, using decision tree as activity classifier, using a dual core NXP development board. We used inertial sensors embedded on the board to detect the activity, and leveraged on the unique dual core architecture of the NXP microcontroller to reduce power consumption respect to a single core platform. We show thus how the NXP dual core platform behaves respect to a single core platform for a typical application of a wearable device. The explored applications are two: an activity recognition algorithm and a median filter. We show that for different configurations, the dual core architecture is advantageous respect to the single core, reducing power consumption up to 37% The paper is organized as follows: after the introduction we describe hardware and software architecture of the system, then we show which methods we used to reduce power consumption on the dual core platform. Next two sections describe the two applications (activity recognition and median filter) and the power comparison with a single core device. The section 6.3 describe in detail how the activity detection algorithm works, which classifier we used and why. We finally draw conclusions.

8.2 System Description

The system is based on the NXP Sensor Processing Solution and the Bosch BXS lite sensor fusion library.

8.2.1 Hardware

The LPC54102 is a microcontroller (MCU) with a full set of peripherals, SRAM and FLASH memory it also integrates two 32 bit ARM cores. The two cores are a Cortex M0+ and a Cortex M4 that can work independently or in parallel. Both ARM cores are implemented as AHB (Advanced High Performance Bus) master and both have full control of the available resources on the LPC54102. Both Cortex M0+ and Cortex M4 are AHB masters, together with DMA.

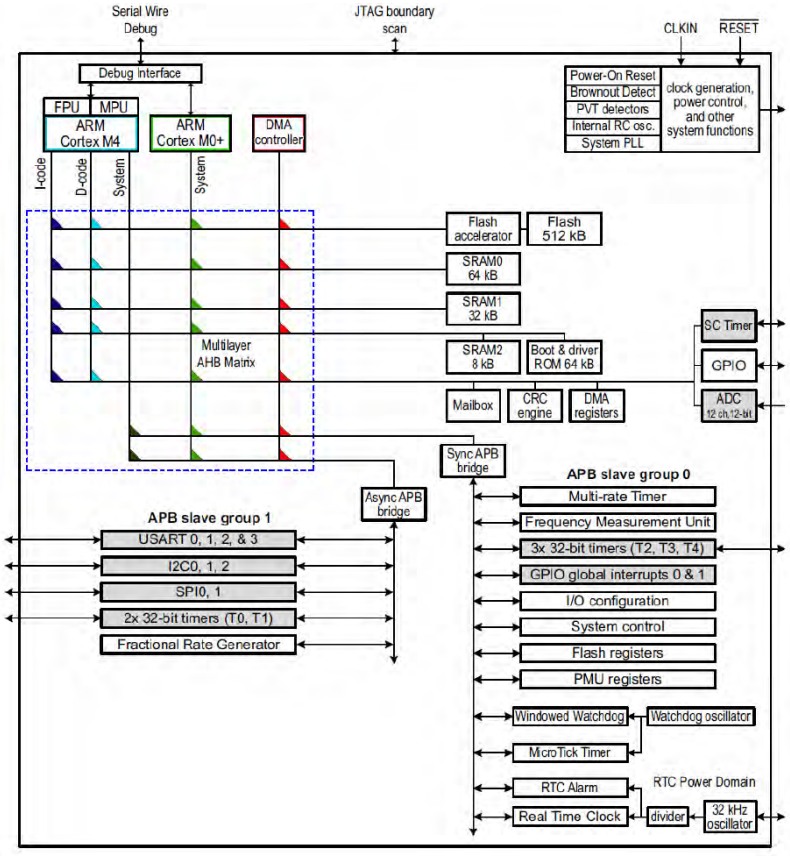


Figure 8.1: Advanced Hugh Performance Bus multilayer matrix for the LPC54102

It is thus clear that accessing to the same resource may result in bus arbitration. To avoid bus contention it is important to plan access to resources (e.g. avoid that both cores try to access same SRAM bank at the same time). The Cortex M4 is equipped with MPU that can prevent unauthorized access to certain memory areas, but DMA and M0 do not have such hardware block and can access to memory areas designed for M4.

The AHB bus on LPC54102 is connected to 3 separate SRAM banks and one Flash accelerator, given this configuration if both cores run in parallel with code located in Flash memory, bus contention may occur. This is one of the reason why we decided to let Cortex M0+ execute code from SRAM1, whereas Cortex M4 executes code from Flash and uses SRAM0 and SRAM2 to store variables. Access to peripherals in our project is usually performed by M0+. We used a demo board with the LPC54102 and debug interface (OM13077). The demo board is connected to a Bosch Sensor Shield which is equipped with a variety of sensors:

- Bosch Sensortec sensors: BMI055 inertial measurement unit, BMC150 digital compass, BMM150 magnetometer, BMP280 pressure/temperature sensor
- Murata pressure/temperature sensor
- MAX44000 ambient light and proximity sensor
- ACKme AMS0002 Bluetooth LE module
- IR remote control driver/receiver
- Dual Knowles digital microphones

8.2.2 Software

The tasks common to almost all wearable nodes can be grouped in: data collection, data processing, transmit results. As sample application we decided implement a Human Activity Classification algorithm based on Decision Tree classifier. As lowest abstraction level, to interface our code with the Microcontroller, we used LPCOpen library delivered by NXP, an

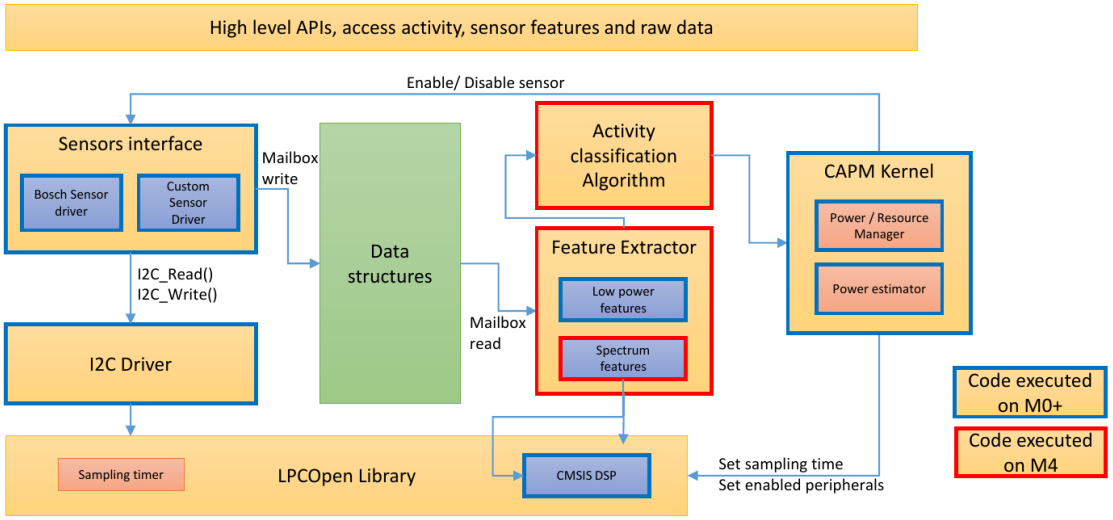


Figure 8.2: Frirmware functional groups

open source library 8.2. On top of the LPCOpen library we developed a I^2C driver for serial communication with sensors, and we used a customized version of the Bosh Sensor Library (BSL), to manage the different configuration and settings of the inertial sensors.

Data collected from sensor are stored in a circular buffer structure, so that only N newest samples are present in the buffer (where N is the size of the buffer). Data buffers are accessed for read operations by the Features Extraction block. This block has access to CMSIS DSP libraries which are delivered by ARM, and contains highly optimized signal processing functions for the specific Cortex M architecture in use.

Once feature vectors have been extracted from signal, are transferred to the classification algorithm. As classifier, we decided to use decision tree, since according to our previous work exhibit good trade-off between accuracy and classification time. The Decision tree used in the C5.0 implementation, which is an improvement respect to the older ID3 and C4.5 algorithms. The last block CAPM Kernel, is used for control purposes (manage active sensors, collect interrupts, manage power state).

Methods

Chapter Intro

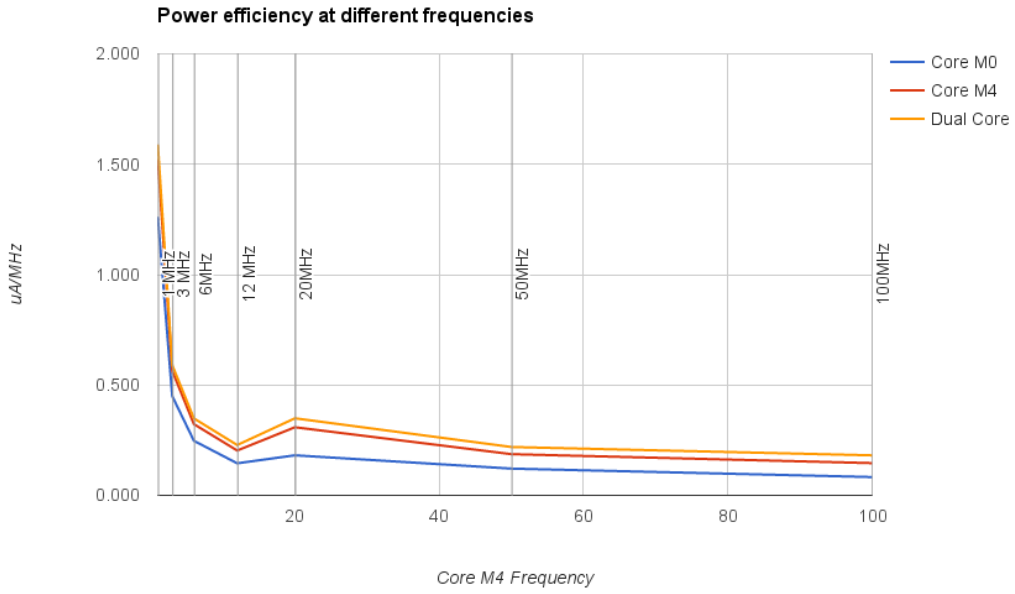


Figure 8.3: Power Efficiency at different operating frequencies (lower is better)

8.2.3 Power Efficiency at different frequencies

To better choose task assignment among the two cores,we performed some simple measurements of computation time and current consumption. We designed a project where both cores performed a for loop where a LED was toggled and different low power modes at different frequencies were used.

We measured the efficiency of the microcontroller for different frequencies (Figure 8.3). The efficiency is expressed as the ration between the current consumption and the clock frequency $\mu A/MHz$. This metric of efficiency is independent from the instruction set of the specific architecture, but it does not deliver information on instructions per seconds per watt. It can be used to find the best operating frequency of the specific platform. From Figure 8.3 it seems that the best operating frequency is close to the maximum frequency of 100 MHz, as also reported by [Fuk15].

8.2.4 Task allocation

As briefly described in the Software section 8.2.2 the tasks allocated to the microcontroller are:

- Kernel this task detect interrupts from sensors, RTC, and decide whether to use one of the available sleep states. This task is executed every time the MCU detect an interrupt or after the completion of another task.
- Sensors communication: this task is deputed to the communication with sensors using the I^2C interface. After a "one time" sensor configuration, the task performs periodic data collection. The frequency of this task depends on the data collection rate.
- Features Extraction: this task processes data from sensors and extract features. Typically this task is executed with much lower frequency than the previous ones (1-2 order of magnitude lower)
- Classification: the classification algorithm is executed, the current activity is predicted, this task is executed with similar frequency of the features extraction task
- Bluetooth Communication: This task receive and send data from remote devices using the Bluetooth Low Energy transceiver

Task distribution among the two cores

We have tested the aforementioned tasks (but the last two) on both cores, to measure energy required for each task, on each core, this way it was possible to decide which task allocate on what core.

Measurements of Figure 8.4 have been performed with both cores in the same condition: clock frequency 12MHz, both running from SRAM, same peripheral configuration. Similar figures can be obtained at different frequencies. Results of Figure 8.4 shows that is more efficient in terms of energy consumption to assign Sensor communication and MCU control tasks to core M0+ and Features extractions to core M4.

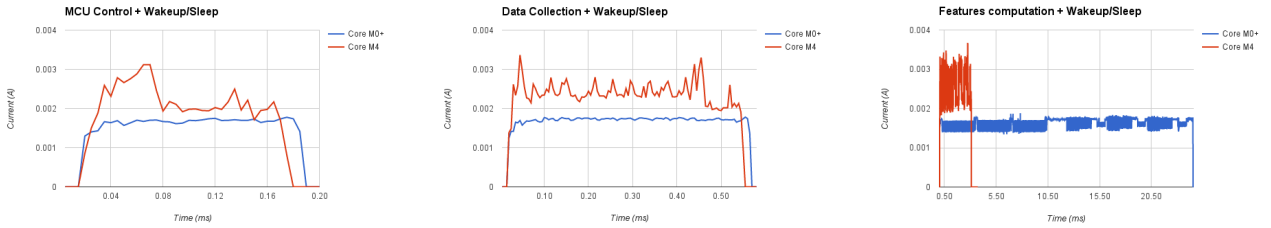


Figure 8.4: Current consumption of the two cores, to perform same task, running frequency 12MHz

8.2.5 Low Power Modes

When in Idle the LPC54102 can use any of the 4 available low power modes:

- SLEEP: this is the softer sleep state, wakeup from this state only requires few clock cycles. This state can be independently used by one of the two cores. All peripherals are active.
- DEEP SLEEP: in this state some peripherals cannot be used, in our configuration the MCU requires around $90\mu s$ to wakeup and go to sleep
- POWER DOWN: this is a state with higher wakeup time and lower current consumption respect to Deep Sleep. Time necessary to wakeup and go to sleep for our configuration is around $120\mu s$
- DEEP POWER DOWN: this state does not allow SRAM retention, thus had not been used for our application during data collection, since it be necessary to fill data buffer with new data, and this operation would require a couple of seconds.

The latter 3 low power modes affect both cores, thus only one core can decide when to enter in Deep Sleep, Power Down and Deep Power Down; to exit from such states there is no particular limitation, the core that has the proper interrupt configuration can be woken up, leaving the other core in SLEEP state.

This means that if the M0+ has control on Low Power modes, only M0+ can decide when to enter low power mode, but the M0+ or M4 can be independently activated by two different interrupt events. In our implementation the kernel task estimates the sleep time and decide which low power mode shall be used.

8.2.6 Core to Core Communication

The LPC54102 is equipped with an advanced communication system to share data among the two cores, it is equipped with mutex for memory exclusive access and Mailbox for variables communication. Mailboxes generate an interrupt on the recipient core, and is possible to send a 32 bit data which can be a pointer and thus virtually any type of data. Mutex are used to avoid concurrent data writing on the same memory location. The core that need to access the shared variable takes the mutex until memory operations are completed.

In our implementation we noticed that if a Mutex was active on the M4, the M4 core wakes up also for interrupts intended for M0. For this reason we used another strategy: we used a GPIO pin connected to External Interrupt line of one core, and driven as General Purpose I/O (GPIO) for the other core. For instance when the M0 need to wakeup the M4, it needs to toggle a GPIO. To transfer data, both cores knows the location of a common memory address (defined at compile time) containing a pointer to data structures.

8.2.7 Data collection using DMA and Interrupts

A possible argument of using Core M0+ for data collection could be that such task could be performed by DMA, relieving cores from such task. Starting from the premise that for other type of sensors, results might be different (i.e. a single sensor with SPI burst communication capabilities may use DMA more efficiently) we evaluated energy consumption for the Sensor communication task with and without DMA.

When DMA was used, it was necessary to configure and enable it, this requires an additional time, this is the reason why in Figure 8.5 the DMA implementation requires more time. The low power mode used in DMA implementation is "Sleep", the MCU goes to Sleep during I^2C communication, and wakes up to manage I^2C events (cannot be done automatically by the peripheral). The lower current consumption using DMA is evident, unfortunately deeper low power modes could not be used since there is not enough time to enter and exit sleep mode between two interrupts. As a result the DMA implementation and Polling implementation had

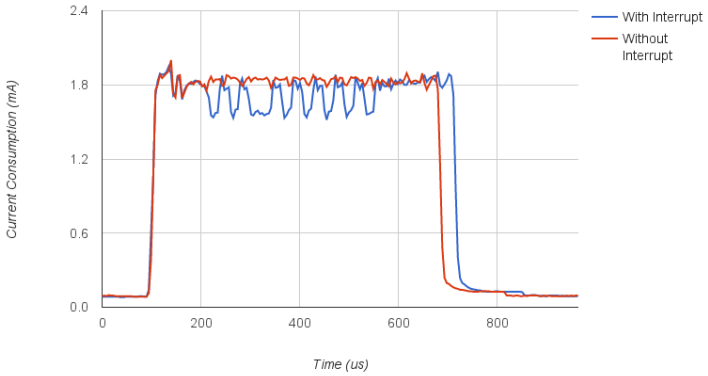


Figure 8.5: Current consumption for the Sensor communication task using DMA and polling

almost the same energy consumption. We measured 3.380 mJ using DMA and 3.383 mJ using polling.

8.3 Single-Dual core comparison

We used the unique dual core Cortex M characteristics of the LPC54102 MCU to compare power consumption for a single versus dual core architecture. All tests have been performed in the same conditions, using the same functions executed on the different cores. We've created 3 different projects two for the single core and one for the dual core. Every project does the same set of operations: data collection from sensors and features extraction.

8.3.1 Common to all configurations

The following parameter were common to all configurations:

- MCU Clock Frequency: 12MHz for all cores
- Sleep mode used: mainly Power Down, except during I^2C communication
- Sensor Communication: I^2C interface with clock set at 400kHz

- Active Sensors: Accelerometer, Gyroscope, Magnetometer
- Sensor sampling rate: Accelerometer 60Hz, Gyroscope 100Hz, Magnetometer 25Hz
- Buffer size on which features are computed: 128 samples, for each axis of the sensors
- Features computed: Mean, Standard Deviation, Maximum, Minimum value, range of the Data, spectral power using Fast Fourier Transform (FFT), Zero Crossing Rate

8.3.2 Core M4 Only

For this project the core M4 performs all tasks. Code of M4 is executed from FLASH memory. Running code from SRAM would reduce current consumption of 8%, but to compare it with the dual core implementation (where M4 Executes code from FLASH) we collected data when code is executed from FLASH. We also verified that current consumption is the same when the Core M0+ is not booted, or when Core M0+ is in *WFI();* state (Wait for Interrupt).

8.3.3 Core M0+ Only

For this project the Core M4 is booted first, it gives the control of low power modes to M0+ and then it boots the M0+. The core M0+ executes code from SRAM. We did not boot directly M0+ because wee need a fair comparison with the Dual core implementation, and the boot process in the dual core implementation is the same.

Dual Core Serial

For the dual core implementation we decide to have the following task division (based on results of Figure 8.4: Core M4: first boot and features extraction; code executed from FLASH Core M0+: Sensors Communication and Kernel; code executed from SRAM Communication among the two cores is done using shared SRAM variables and External Interrupts. The Core M0 is woke up by RTC or sensors Interrupt, it detect which sensor has the data ready and the it goes

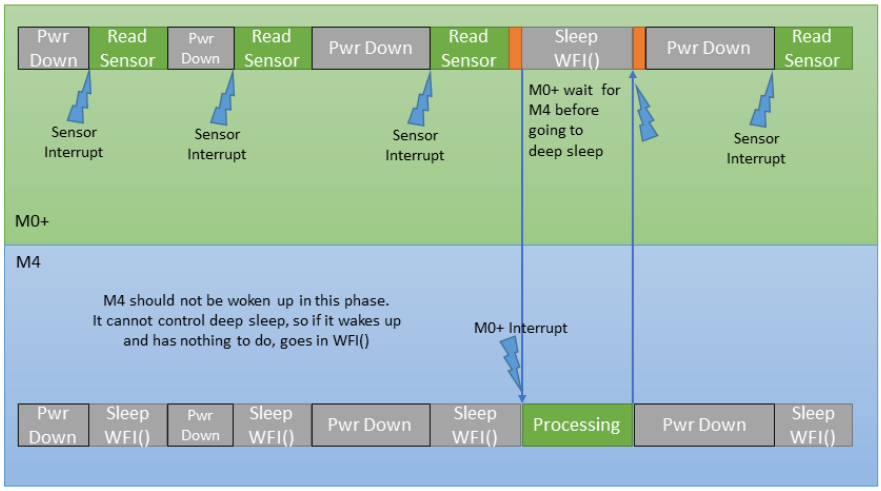


Figure 8.6: Dual core serial implementation time diagram

to low power mode(Figure 8.6). Once the M0 has collected enough data, it wakes up the M4 that performs the features extraction. Once also M4 completed its task, the M0 set the MCU in low power mode.

8.3.4 Dual Core Parallel

The dual core parallel implementation has similar architecture of the previous case, with the difference that the M0+ let start the features extraction by M4 while it is collecting data. Using this method the processing is executed with one sample delay, but both cores work together for a limited amount of time. In both cases the M0 after data collection goes in sleep and wait for a signal from M4 telling that processing is complete and the chip can be set in low power mode.

8.3.5 Measurement results

To analyze performance of the single and dual core configuration, we measured current consumption of the LPC54102 chip, using the embedded current measurement tool. We also verified the measurement using an oscilloscope. The embedded current measurement tool has

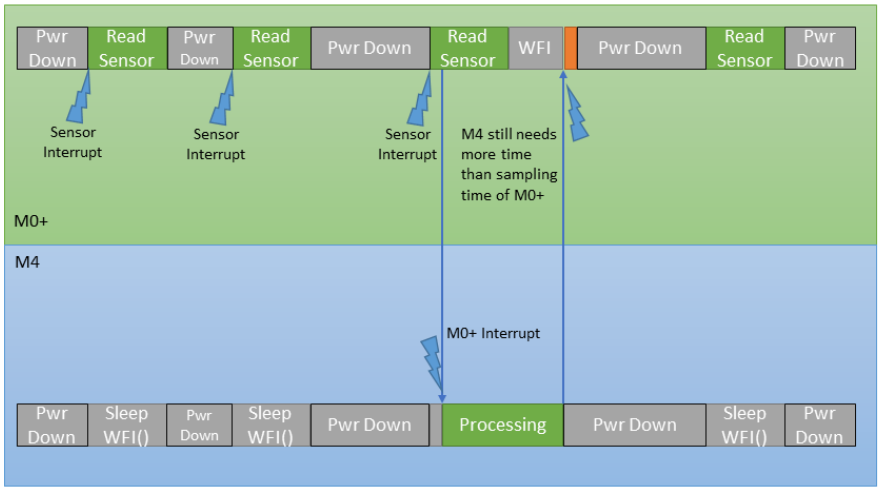


Figure 8.7: Dual core parallel implementation time diagram

a sampling frequency of 200kHz and a resolution of $4\mu A$. We collected data for 5 seconds (1 Million samples per measure) and averaged them.

In Figure 8.8 we show results of our measurements for the two dual core configurations and the two single core configurations. One parameter that we used in our tests is the processing frequency, that is the frequency with the M4 is woke up to perform features extraction. For the higher processing rate (30Hz), the M0 was not able to complete computation before the arrival of new data, so the real-time constraint was not satisfied.

It is possible to notice that the dual core implementation consumes less power than the single core implementation for the analyzed scenarios. Moreover the parallel implementation is more advantageous respect to the single core, but this is evident at higher processing rates, since the time in which both cores are active is higher.

In this scenario we analyzed the duty cycle for the different projects at a processing rate of 30Hz, results are shown in Figure 8.9

As we mentioned before, the time for which both cores are active is really limited, this is mainly due to the fact the M4 remains active more time than the sampling time of M0. Moreover the M0 is active also to sample other sensor, even if we do processing at 30 Hz (1 processing every two accelerometer samples), the M0 has to collect also other sensors.

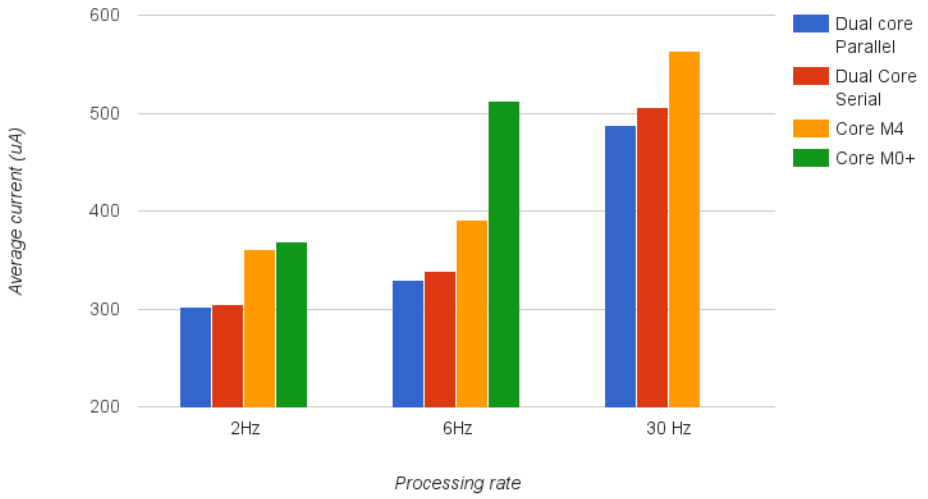


Figure 8.8: Comparison of the average current consumption for different processing rates. M0+ did not complete the computation on time for the 30Hz processing rate

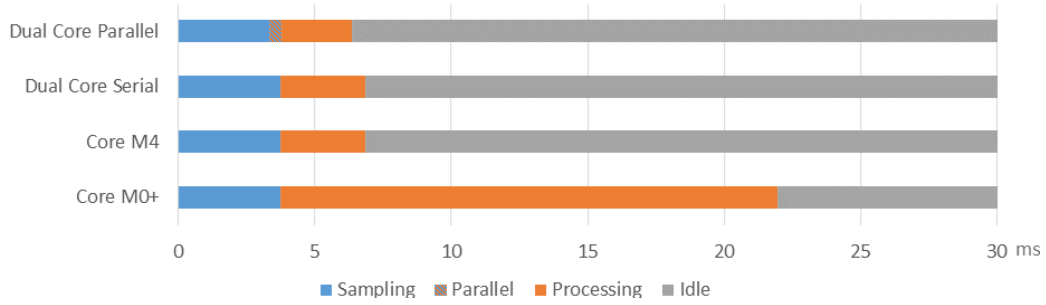


Figure 8.9: Duty cycle of the various core when processing at 30Hz

8.4 Dual Core time maximization

As is possible to notice from Figure 8.3 the power consumption of both cores is significantly lower that the sum of power consumption of the sum of the single cores. This is mainly due to the fact that in the LPC54102 many components are shares (Peripherals, clock generator, etc). To verify the advantages of such dual core architecture in a real case scenario we also prepared 3 different firmware for the single and dual core scenarios, where we maximized the time in which both cores are contemporary active.

8.4.1 Firmware structure

The firmware for this case is very similar to the section 8.35rr55d/dd//5d. The main differences are:

- Gyroscope and magnetometer are switched off
- Sampling rate of the accelerometer increased to 500Hz
- Data Processing is performed with higher frequency (every 1,2 or 4 new samples
- The data processing routine is much more simple and does not perform features extraction but a simple average filter that computes the average value over 128 samples for the 3 axis of the accelerometer. This filter requires almost the same time used by M0+ to read accelerometer.

8.4.2 Measurement Result

Also for this scenario we collected average current consumption for a 5s window with a sampling rate of 200kHz, results are shown in Figure 8.10. In this case the advantage of the dual core platform is much more evident than in the previous application. Moreover the possibility to use both cores in parallel further reduce power consumption.

We measured that the dual core serial implementation reduces power by 27% compared to M4 and the parallel implementation reduce power consumption by 37% respect to the M4. The core M0+ could complete the processing before the arrival of new data only when the processing was performed every 4 new samples.

We also measured Duty cycle for this scenario, results are shown in Figure 8.11. It is possible to notice that for the parallel implementation, cores are contemporary active for a much longer time respect to Figure 8.9, this allows a much longer idle time and lower current consumption. We want to stress out that this is different from a benchmark, it is a real world application, in

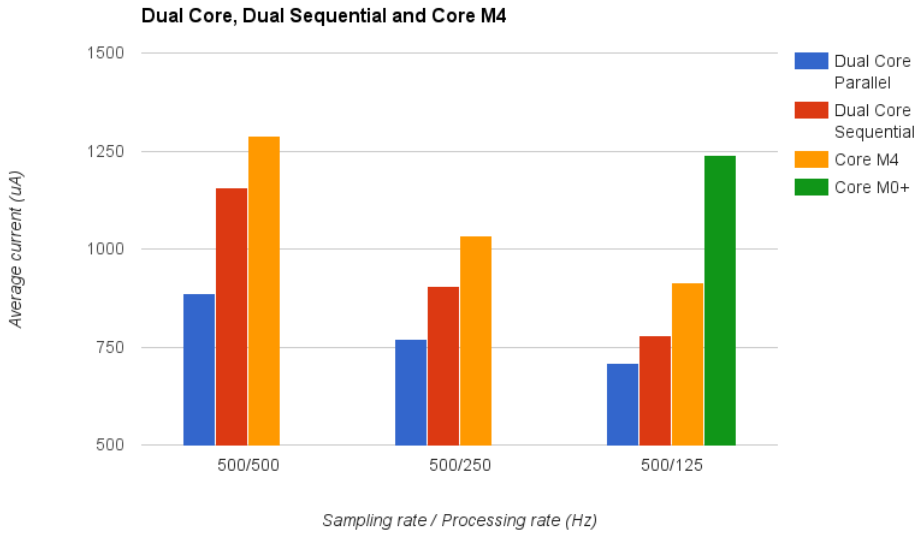


Figure 8.10: Average current consumption for the filtering application, in this case the core M0+ was able to process only 1 sample every 4 new data (sampling rate 500Hz, processing rate 125Hz)

fact it was not possible to run everything in parallel, the kernel task for instance was executed only by M0+.

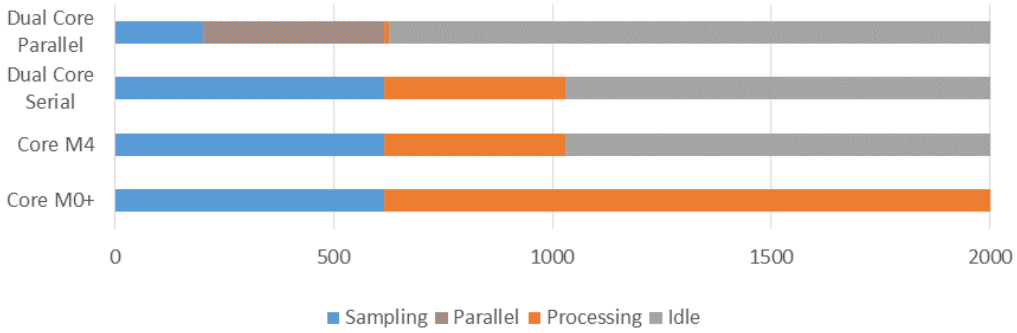


Figure 8.11: Duty cycle for the filter application

8.5 Activity detection Algorithm

The general structure of an Activity detection algorithm can be found in literature [BBS14], and has been also described in section 6.3.

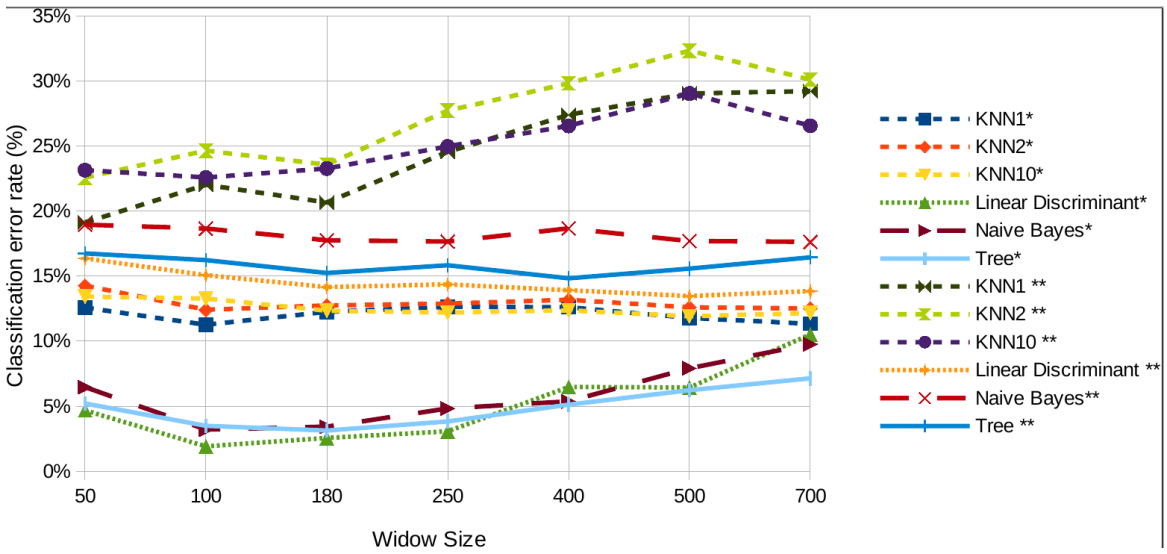


Figure 8.12: Accuracy for different classifiers in single and multi subject scenario

8.5.1 Choices for implementation

We've simulated the execution time and accuracy for different scenarios, using different classifiers, and different window size. Results for Execution time are shown in Figure 6.1. It is possible to notice that the Bayesian, Decision Tree and Linear discriminant classifiers, are the ones wit lower computation time. In Figure 6.1 the size of the window also affect classification time, since a smaller window corresponds to a less complex classifier, but also slower response time.

We also evaluated accuracy of the classifier, using an online dataset [AGO+13]. The accuracy has been evaluated for the single subject case (one classifier trained and tested on one subject) and multi-subject (one single classifier is trained and tested on all subjects). Results for different classifiers are available in Figure 8.12. It is possible to notice that all classifiers have worse performance for the multi-subject scenario, this is due to the fact different subject may perform the same movement in different ways. Also in this case, Bayes, Decision tree and Linear Discriminant have better performances (in terms of accuracy). We decided to use decision tree since it allows easy interpretation of the classifier structure and is possible to tailor some parameters (e.g. pruning) to further reduce computation complexity.

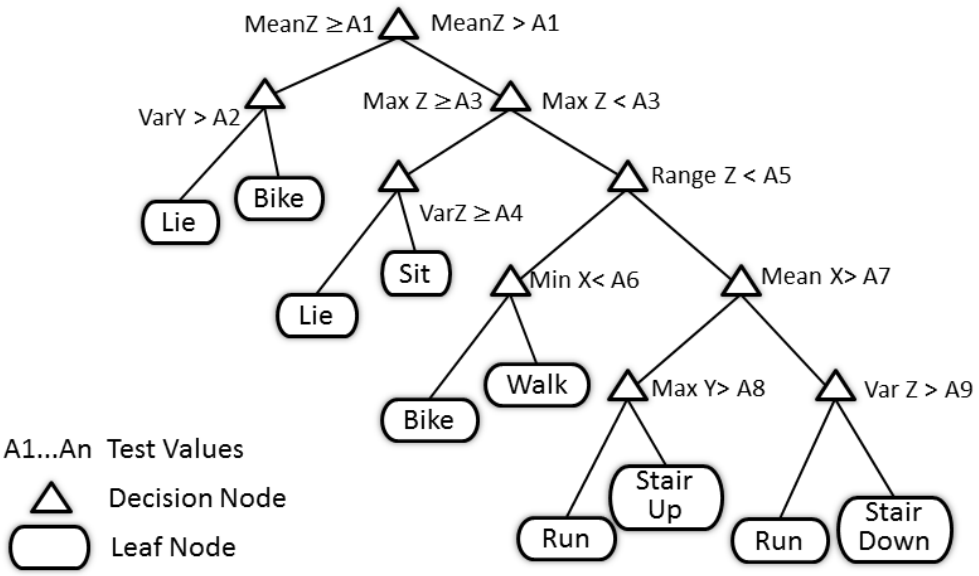


Figure 8.13: Example of a simple decision tree structure

8.6 Decision tree implementation

Decision Trees are widely used in machine learning, since, respect to other classifiers, they are easy to understand an interpret.

We can imagine a dataset as a matrix whose rows are instances, each instance is a feature vector. The Decision Trees examines the feature of given instance and comes to a conclusion on what label to assign based on the values present for the various features of that particular instance, in our case the label is the subject's predicted activity. Each node in the decision tree can be a decision node or a leaf node. The leaf is a terminal node, when the algorithm reaches the leaf node, computation is complete. Decision node is a node where one particular feature is tested, according to the value of the test feature, a different branch of the tree is selected (Figure 8.13).

Different version of Decsion tree Algorithm exists, we'll list the more common:

8.6.1 ID3

The ID3 is an iterative algorithm, it iterates through every unused feature of the set S and calculates the entropy H(S) of that feature. It then selects the feature which has the smallest entropy (or largest information gain) value. The set S is then split by the selected feature (e.g. age ¡ 50, 50 ¡= age ¡ 100, age ¿= 100) to produce subsets of the data. The algorithm continues to recur on each subset, considering only attributes never selected before. Recursion on a subset may stop in one of these cases:

- Every element in the subset belongs to the same class (+ or -), then the node is turned into a leaf and labeled with the class of the examples
- There are no more features to be selected
- There are no examples in the subset, this happens when no example in the parent set was found to be matching a specific value of the selected attribute, for example if there was no example with age ¿= 100. Then a leaf is created, and labeled with the most common class of the examples in the parent set.

8.6.2 C4.5

C4.5 is an evolution of ID3, it uses same splitting criteria, but made a number of improvements to ID3. Some of these are:

- Handling both continuous and discrete attributes - In order to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it.
- Handling training data with missing attribute values
- Pruning trees after creation - C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes.

8.6.3 CART

Classification and Regression Tree (CART) algorithm, similarly to ID3 uses a recursive splitting algorithm, but, instead of entropy it uses a generalization of the binomial variance called the Gini index. First it grows an overly large tree and then prune it to a smaller size to minimize an estimate of the misclassification error. CART employs 10-fold (default) cross-validation, whereas C4.5 uses a heuristic formula to estimate error rates.

8.6.4 C5.0

Like C4.5, it has tree and rule-based versions and shares much of its core algorithms with its predecessor, the program source code, which was made available to the public in 2011. C5.0 offers a number of improvements on C4.5. Some of these are [KJ13]:

- Speed - C5.0 is significantly faster than C4.5 (several orders of magnitude)
- Memory usage - C5.0 is more memory efficient than C4.5
- Smaller decision trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- Support for boosting - Boosting improves the trees and gives them more accuracy.
- Weighting - C5.0 allows you to weight different cases and misclassification types.

We decided to implement on the microcontroller the C5.0 algorithm, since it is the newest algorithm, and has less computational resources requirement respect to the other classifiers.

8.7 Conclusions

In this work we showed how, a wearable device, using the NXP dual core platform can deliver significant power savings respect to a single core platform. As an example we analyzed two

applications typical of wearable devices: an Activity recognition Algorithm and a filtering applications. In both cases the dual core platform allowed significant power savings respect to a single core platform.Moreover the possibility to run both cores in parallel, enable further reduction of power consumption. We measured reduction in power from 10% to 40% . In general we can extend the advantages of the dual core platform for all the applications that share the two tasks: peripheral communication and data processing.

Chapter 9

Conclusion

Energy consumption in embedded system is a challenge since many years, astonishing results have been achieved with new low power architecture. Nevertheless the continuous miniaturization of electronic components paves the way for smaller devices that can be worn or even implanted on our body. Energy density in batteries does not evolve at the same rate as silicon component do; moreover nowadays application demand always connected devices, always on sensors and continuous monitoring, especially for healthcare applications. To reduce energy consumption of a system is necessary to combine different fields of electronics, from components selection, to board design, radio management and software techniques.

9.1 Summary of Thesis Achievements

In this work we were able to combine different techniques and apply them for several platforms in order to reduce power consumption of wearable devices. We started with the description of the CUPID platform where a combination of low power sensors, low power microcontroller and radio transceiver,allowed us to have a reliable device whose battery duration was within the application requirements.

Nevertheless using with the knowledge fo the platform it was possible, through accurate management of the low power states of the component develop a firmware capable to maximize

energy savings without losing performances and reliability.

The concept has been later on generalized making it independent from the specific platform, thanks to the use of an embedded operating system. The OS allows to export one technique to a variety of platforms. It allows an higher abstraction level, and the programmer does not need to know detail of every single platform to apply low power techniques. We were able to create an infrastructure within the OS that enables it for low power management.

The hardware platform of the CUPID project although showed some weaknesses: the device had to be turned on and off manually by the patient, which usually is an old person. Why not to do it automatically? And this is what we have done: we were able to add a passive radio wake up that allowed to completely turn off the sensor node when not used, and almost instantaneously turn on when the patient needed to be monitored.

Another weakness was the fact that it was not possible for instance to use deed sleep states of the microcontroller because this would have lead to data loss and higher power consumption due to the break-even time.This lead us to go one step further in to the hardware modification: we completely changed platform, using a dual core architecture, one of the cores was able of high computational performance, the other was characterized by ultra low power consumption. The two cores were managed so to have performance when needed and great energy savings in low power mode.

9.2 Future Work

As next work it will be interesting to generalize the concept of Context Aware Power Management, we tried it on the CUPID node, on a modified version of it and on a dual core platform. But we used only a limited set of sensors (inertial sensors) to detect the context. Future works are in the direction of integrating more and more sensors to make the CAPM more accurate and more versatile for use in different application. It will also be possible to create a distributed version of the CAPM where multiple sensors data, coming from different sources can be aggregated to detect the user's context. Very interesting is the use of microphones and low power

cameras that can augment the number of detectable context and thus tune with more efficiency the power consumption of a wearable sensor node.

List of Tables

List of Figures

Bibliography

[A+12] D. Anguita et al. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care*, pages 216–223. Springer, 2012.

[AGO+13] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, 2013.

[APH13] Anne Felicia Ambrose, Geet Paul, and Jeffrey Hausdorff. Risk factors for falls among older adults: A review of the literature. *Maturitas*, 2013.

[ARP06] Ankur Agarwal, Saeed Rajput, and Abhijit S Pandya. Power management system for embedded rtos: An object oriented approach. In *Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on*, pages 2305–2309. IEEE, 2006.

[B+12] S. Buthpitiya et al. Hermes – a context-aware application development framework and toolkit for the mobile environment. In *(WAINA), 2012 26th International Conference on*, pages 663–670, March 2012.

[BBDM00a] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(3):299 –316, june 2000.

[BBDM00b] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *(VLSI) Systems, IEEE Transactions on*, 8(3):299–316, June 2000.

[BBS14] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[BCD+05] Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. *Mobile Networks and Applications*, 10(4):563–579, 2005.

[Blu12] Bluegiga Technologies. *iWrap 5.0 User Guide*, August 2012.

[C+10] Michael P. Caligiuri et al. Handwriting movement kinematics for quantifying extrapyramidal side effects in patients treated with atypical antipsychotics. *Psychiatry Research*, 177(12):77 – 83, 2010.

[CFB13] Filippo Casamassima, Elisabetta Farella, and Luca Benini. Power saving policies for multipurpose wban. In *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2013 23rd International Workshop on*, pages 83–90. IEEE, 2013.

[CFB14] F. Casamassima, E. Farella, and L. Benini. Context aware power management for motion-sensing body area network nodes. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE, 2014.

[CFM+13] Filippo Casamassima, Alberto Ferrari, Bojan Milosevic, Laura Rocchi, and Elisabetta Farella. Wearable audio-feedback system for gait rehabilitation in subjects with parkinson's disease. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing*, pages 275–278. ACM, 2013.

[Cir10] Cirrus Logic Inc. *CS43L22 - Low Power, Stereo DAC w/Headphone & Speaker Amps*, 03 2010.

[CL10] K.K. Chui and M.M. Lusardi. Spatial and temporal parameters of self-selected and fast walking speeds in healthy community-living adults aged 72–98 years. *Journal of Geriatric Physical Therapy*, 33(4):173, 2010.

[CLCC09] H. Cao, V. Leung, C. Chow, and H. Chan. Enabling technologies for wireless body area networks: A survey and outlook. *Communications Magazine, IEEE*, 47(12):84–93, 2009.

[CWS08] Benedykt Cichy, Magdalena Wilk, and Zbigniew SLIWNISKI. Changes in gait parameters in total hip arthroplasty patients before and after surgery. *Medical science monitor*, 14(3), 2008.

[DGV04] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[DLC14] Sean T. Doherty, Christopher J. Lemieux, and Culum Canally. Tracking human activity and well-being in natural environments using wearable sensors and experience sampling. *Social Science and Medicine*, 106:83 – 92, 2014.

[ERR05] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: an energy-aware resource-centric rtos for sensor networks. In *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, pages 10 pp.–265, Dec 2005.

[F+12] Fraternali et al. Opportunistic hierarchical classification for power optimization in wearable movement monitoring systems. In *(SIES), 2012 7th IEEE International Symposium on*, pages 102–111, June 2012.

[FFMK11] Pamela Fok, Michael Farrell, Joan McMeeken, and Yi-Liang Kuo. The effects of verbal instructions on gait in people with parkinsons disease: a systematic review of randomized and non-randomized trials. *Clinical rehabilitation*, 25(5):396–407, 2011.

[Fuk15] A. Fuks. Sensor-hub sweet-spot analysis for ultra-low-power always-on operation. In *VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pages C154–C155, June 2015.

[GASS15] H. Ghasemzadeh, N. Amini, R. Saeedi, and M. Sarrafzadeh. Power-aware computing in wearable sensor networks: An optimal feature selection. *Mobile Computing, IEEE Transactions on*, 14(4):800–812, April 2015.

[Gou09] Brian Gough. *GNU Scientific Library Reference Manual - Third Edition*. Network Theory Ltd., 3rd edition, 2009.

[GS04] Lin Gu and John A Stankovic. Radio-triggered wake-up capability for sensor networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 27–37, 2004.

[ISH12] Dave Zachariah Isaac Skog, John-Olof Nilsson and Peter Hndel. Fusing the information from two navigation systems using an upper bound on their maximum spatial separation. 2012. In proc. IPIN 2012.

[Jan08] J. Jankovic. Parkinson's disease: Clinical features and diagnosis. *Journal of Neurology, Neurosurgery and Psychiatry*, 79(4):368–376, 2008.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[KHL+07] Kevin Klues, Vlado Handziski, Chenyang Lu, Adam Wolisz, David Culler, David Gay, and Philip Levis. Integrating concurrency control and energy management in device drivers. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 251–264. ACM, 2007.

[KJ13] Max Kuhn and Kjell Johnson. Classification trees and rule-based models. In *Applied Predictive Modeling*, pages 369–413. Springer New York, 2013.

[KKM+14] Pouya Kamalinejad, Kamyar Keikhosravy, Michele Magno, Shahriar Mirabbasi, Victor Leung, and Luca Benini. A high-sensitivity fully passive wake-up radio

front-end for wireless sensor nodes. In *Consumer Electronics (ICCE), 2014 IEEE International Conference on*, pages 209–210. IEEE, 2014.

[Knu71] D.E. Knuth. Optimum binary search trees. *Acta Informatica*, 1(1):14–25, 1971.

[KRK03] O. Karjalainen, S. Rantala, and M. Kivikoski. A comparison of bluetooth low power modes. In *Telecommunications, 2003. ConTEL 2003. Proceedings of the 7th International Conference on*, volume 1, pages 121 – 128, 2003.

[KS15] Ki Joon Kim and Dong-Hee Shin. An acceptance model for smart watches: implications for the adoption of future wearable technology. *Internet Research*, 25(4):527–541, 2015.

[L+11] B. Latré et al. A survey on wireless body area networks. *Wirel. Netw.*, 17(1):1–18, January 2011.

[LGR13] Sue Lord, Brook Galna, and Lynn Rochester. Moving forward on gait measurement: Toward a more refined approach. *Movement Disorders*, 28(11):1534–1543, 2013.

[LMM+04] Susan E Lord, Kathryn McPherson, Harry K McNaughton, Lynn Rochester, and Mark Weatherall. Community ambulation after stroke: how important and obtainable is it and what measures appear predictive? *Archives of physical medicine and rehabilitation*, 85(2):234–239, 2004.

[LMP+05] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer, 2005.

[LP11] Jung Keun Lee and Edward J Park. Quasi real-time gait event detection using shank-attached gyroscopes. *Medical & biological engineering & computing*, 49(6):707–712, 2011.

[LPSS10] Rafael Lajara, José Pelegrí-Sebastiá, and Juan J Perez Solano. Power consumption analysis of operating systems for wireless sensor networks. *Sensors*, 10(6):5809–5826, 2010.

[LVW+10] D.E. Lieberman, M. Venkadesan, W.A. Werbel, A.I. Daoud, S. DAndrea, I.S. Davis, R.O. MangEni, and Y. Pitsiladis. Foot strike patterns and collision forces in habitually barefoot versus shod runners. *Nature*, 463(7280):531–535, 2010.

[M+13] M. Moghimi et al. Context-aware mobile power management using fuzzy inference as a service. In *Mobile Computing, Applications, and Services*, pages 314–327. Springer, 2013.

[MHP08] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 53(5):1203–1218, 2008.

[MNF+12] Bojan Milosevic, Roberto Naldi, Elisabetta Farella, Luca Benini, and Lorenzo Marconi. Design and validation of an attitude and heading reference system for an aerial robot prototype. In *American Control Conference (ACC), 2012*, pages 1720–1725. IEEE, 2012.

[MP11] Stevan Marinkovic and Emanuel Popovici. Nano-power wake-up radio circuit for wireless body area networks. In *Radio and Wireless Symposium (RWS), 2011 IEEE*, pages 398–401. IEEE, 2011.

[N+03] B. Najafi et al. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *Biomedical Engineering, IEEE Transactions on*, 50(6):711–723, June 2003.

[N+11] John G Nutt et al. Freezing of gait: moving forward on a mysterious clinical phenomenon. *The Lancet Neurology*, 10(8):734 – 744, 2011.

[NDCB10] Jad Noueihed, Robert Diemer, Samarjit Chakraborty, and Stefanie Biala. Comparing bluetooth hdp and spp for mobile health devices. In *Body Sensor Networks (BSN), 2010 International Conference on*, pages 222 –227, june 2010.

[NRMS09] Alice Nieuwboer, Lynn Rochester, Liesbeth Müncks, and Stephan P Swinnen. Motor learning in parkinson's disease: limitations and potential for rehabilitation. *Parkinsonism & related disorders*, 15:S53–S58, 2009.

[NSHH12] J-O Nilsson, Isaac Skog, P Handel, and KVS Hari. Foot-mounted ins for everybody-an open-source embedded implementation. In *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, pages 140–145. IEEE, 2012.

[Nut14] Gregory Nutt. Nuttx: a real-time operating system (rtos) with an emphasis on standards compliance and small footprint. http://nuttx.org, 2007–2014.

[P+08] Mirthe M. Ponsen et al. Impairment of complex upper limb motor function in de novo parkinson's disease. *Parkinsonism & Related Disorders*, 14(3):199 – 204, 2008.

[PD+92] Jacquelin Perry, Jon R Davids, et al. Gait analysis: normal and pathological function. *Journal of Pediatric Orthopaedics*, 12(6):815, 1992.

[PHH+11] J. Plomp, M. Heiskanen, M. Hillukkala, T. Heikkilä, J. Rehu, N. Lambert, V. van Acht, and T. Ahola. Considerations for synchronization in body area networks for human activity monitoring. *International Journal of Wireless Information Networks*, 18(4):280–294, 2011.

[PMFC11] Marco Pirini, Martina Mancini, Elisabetta Farella, and Lorenzo Chiari. Eeg correlates of postural audio-biofeedback. *Human movement science*, 30(2):249–261, 2011.

[PSC05] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369. IEEE, 2005.

[RBH+10] Lynn Rochester, Katherine Baker, Victoria Hetherington, Diana Jones, Anne-Marie Willems, Gert Kwakkel, Erwin Van Wegen, Inge Lim, and Alice Nieuwboer. Evidence for motor learning in parkinson's disease: acquisition, automaticity and

retention of cued gait performance after training with external rhythmical cues. *Brain research*, 1319:103–111, 2010.

[RCM+13] Ely Rabin, Jason Chen, Lisa Muratori, Joanne DiFrancisco-Donoghue, and William G Werner. Haptic feedback from manual contact improves balance control in people with parkinson's disease. *Gait & posture*, 2013.

[RGL04] Olof Rensfelt, Richard Gold, and Lars-Åke Larzon. Lunar over bluetooth. In *Proceedings of the 4:th Scandinavian Workshop on Wireless Ad-Hoc Networks*, 2004. Internet Official Protocol Standards Request for Comments (RFC) 3828.

[RR07] M. Ringwald and K. Romer. Practical time synchronization for bluetooth scatternets. In *Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on*, pages 337–345. IEEE, 2007.

[S+08] R. Senguttuvan et al. Multidimensional adaptive power management for low-power operation of wireless devices. *Circuits and Syst. II: Exp. Briefs, IEEE Transactions on*, 55(9):867–871, Sept 2008.

[Sar12] R. Sarpeshkar. Universal principles for ultra low power and energy efficient design. *Circuits and Syst. II: Exp. Briefs, IEEE Transactions on*, 59(4):193–198, April 2012.

[SBV+13] Arash Salarian, Pierre R Burkhard, François JG Vingerhoets, Brigitte M Jolles, and Kamiar Aminian. A novel approach to reducing number of sensing units for wearable gait analysis systems. *Biomedical Engineering, IEEE Transactions on*, 60(1):72–77, 2013.

[SNH10] Isaac Skog, John-Olof Nilsson, and Peter Händel. Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–6. IEEE, 2010.

[Sta] Staedtler. Digital pen 990 device. http://www.staedtler.com/.

[T+97] Hans-Leo Teulings et al. Parkinsonism reduces coordination of fingers, wrist, and arm in fine motor control. *Experimental Neurology*, 146(1):159 – 170, 1997.

[TDMM12] Dawn Tan, Mary Danoudis, Jennifer McGinley, and Meg E Morris. Relationships between motor aspects of gait impairments and activity limitations in people with parkinson's disease: a systematic review. *Parkinsonism & related disorders*, 18(2):117–124, 2012.

[TTT+09] Ryo Takeda, Shigeru Tadano, Masahiro Todoh, Manabu Morikawa, Minoru Nakayasu, and Satoshi Yoshinari. Gait analysis using gravitational acceleration measured by wearable sensors. *Journal of Biomechanics*, 42(3):223 – 233, 2009.

[UK12] Sana Ullah and Kyung Sup Kwak. An ultra low-power and traffic-adaptive medium access control protocol for wireless body area network. *Journal of medical systems*, 36(3):1021–1030, 2012.

[VHZ+12] Rosemarie Velik, Ulrich Hoffmann, Haritz Zabaleta, Jose Felix Marti Masso, and Thierry Keller. The effect of visual cues on the number and duration of freezing episodes in parkinson's patients. In *Engineering in Medicine and Biology Society (EMBC)*, pages 4656–4659. IEEE, 2012.

[WAC] WACOM. Inkling device. http://www.wacom.com/.

[WOL11] J. Whslen, I. Orhan, and T. Lindh. Local time synchronization in bluetooth piconets for data fusion using mobile phones. In *Body Sensor Networks (BSN), 2011 International Conference on*, pages 133 –138, may 2011.

[Y+99] Victor Yodaiken et al. The rtlinux manifesto. In *Proc. of the 5th Linux Expo*, 1999.

[YPP+07] Galit Yogev, Meir Plotnik, Chava Peretz, Nir Giladi, and Jeffrey M Hausdorff. Gait asymmetry in patients with parkinsons disease and elderly fallers: when does the bilateral coordination of gait require attention? *Experimental brain research*, 177(3):336–346, 2007.

[YSGBH12] Galit Yogev-Seligmann, Nir Giladi, Marina Brozgol, and Jeffrey M Hausdorff. A training program to improve gait while dual tasking in patients with parkinson's disease: a pilot study. *Archives of physical medicine and rehabilitation*, 93(1):176–181, 2012.

[YY06] Guang-Zhong Yang and Magdi Yacoub. Body sensor networks. 2006.