PhD Dissertation

UNIVERSITÀ DEGLI STUDI
DI TRENTO

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Cyber-Physical Systems: Two Case Studies in Design Methodologies

Luca Rizzon

Advisor:

Prof. Roberto Passerone

Università degli Studi di Trento

April 2016

Abstract

To analyze embedded systems, engineers use tools that can simulate the performance of software components executed on hardware architectures. When the embedded system functionality is strongly correlated to physical quantities, as in the case of Cyber-Physical System (CPS), we need to model physical processes to determine the overall behavior of the system. Unfortunately, embedded systems simulators are not generally suitable to evaluate physical processes, and in the same way physical model simulators hardly capture the functionality of computing systems. In this work, we present a methodology to concurrently explore these aspects using the METROII *design framework. The methodology provides guidelines for the implementation of these models in the design environment. To demonstrate the feasibility of the proposed approach, we applied the methodology to two case studies. A case study regards a binaural guidance system developed to be included into a smart rollator for older adults. The second case consists of an energy recovery device which gets energy from the heat dissipated by a high performance processor and power a smart sink able to provide cooling or to serve as a wireless sensing node.*

Keywords
[Cyber-Physical Systems, Design Methodology, Binaural Synthesis, Energy Harvesting]

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 CPS Design Challenges

Embedded Systems (ESs) are computing devices specifically designed, and optimized to perform a specific task. This term highlights how ESs are in opposition to general computing platforms, which are designed to execute a variety of different tasks that can differ significantly one to the other. PDAs, smartphones, car ECUs, avionics, industrial automation systems, home automation, among many others are all examples of instances of ESs. Those applications encompass different domains, therefore they differ in requirements and attributes, however they share common design challenges. For example, mobile devices require low energy consumption profile to guarantee an acceptable battery lifetime without compromising the device size or weight. In the case of systems from which the individual's safety depends, such as for example cars ABS, or avionics system, the execution time of a certain task is crucial, therefore the execution time of a task, and its probability of being interrupted from other events, are of fundamental importance, as opposed to the case of personal computers.

Recently, ES applications made of software and hardware components that work together to *monitor and to control physical quantities* have received growing attention [55]. These applications, which exhibit a tight coupling between the computation domain and the physical world, are referred to as Cyber-Physical System (CPS). In a typical CPS, a physical parameter is sampled with an appropriate sensor, the data is processed by the cyber component (that is the computing unit), a control algorithm determines the parameter to apply to an actuator, and the actuator can manipulate, or influence the physical parameter subject to control. This cross-relation complicates the design, especially because the computational and the physical components are handled by professionals with different profiles and are generally studied with models which differ significantly. Moreover, computational world and physical world belong to two very different domains and are generally studied with models which differ significantly. The cyber domain is made of sequential operations, that are punctuated in discrete time (i.e. the clock), physical process can be concurrent and modeled with expression in the domain of real numbers.

For engineers developing CPS it is of fundamental importance to design the cyber as well as the physical part of the system taking into account both the computational unit features, and the results of the interaction with the physical world. Both aspects, in fact, affect important system metrics, such as energy efficiency, efficient utilization of limited computational resources (memory, processors, speed), real-time constraints, and predictability of the outcome. Embedded systems modeling can be used for verification, for power consumption estimation, and to select the best hardware architecture for the design. Typically, these tools are event–based since this approach is more practical to simulate hardware interrupts, or message exchange between components. On the contrary, to model physical phenomena we need tools which are able to handle a different domain to include models typical of the physical world, such as for example differential equations. Thus, due to the intrinsic difference that exists between the cyber and physical domain, design tools and simulators designed to model one world are typically not suitable to model the other, and vice versa [36]. Therefore, the current practice is to study the two worlds separately, with the risk of not properly modeling how the two interact. Moreover, only very few description of studies applied to real cases have been presented in the literature [32].

In this thesis, we propose a methodology to integrate computational models and physical models using a design strategy inspired by the function/architecture co-design paradigm: while cyber and physical components are conceptually separated, they are integrated over two mapping steps to result in an overall system model. We support the methodology using the METROII design exploration tool [15]. In particular, we explore the functionalities offered by the METROII design framework and how to use them in practice by studying how to analyze two very different case studies, to identify and outline best design practices that can be adopted to express and combine the cyber/physical features in METROII. The proposed practices can be adapted or extended to model other, equally diverse systems.

We explore the functionalities of the METROII design framework by studying how to simulate two very different systems, with the objective of outlining practices that we adopted to be able to express system features through the elements already defined in METROII. To illustrate the approach, we will describe the design of two systems very different one to the other. These systems were chosen because the author has been actively involved in their development, and they have some characteristic features of the CPSs.

We describe the design of an auditive guidance interface developed to be used in a smart walker for the older adults. The auditive guidance system generates spatial 3D sounds to give instructions to the user of the walker regarding the displacement of the suggested path to follow stimulating higher level of cognition. To provide a convincing effect, while keeping the computational requirement acceptable for the application, we developed a sound generation software based on a simplified sound propagation model and we take advantage of user head movement to increase the level of realism. However, the interface must generate sound within a time limit, otherwise it will not be effective from a perceptual viewpoint. Since the interface software cooperates with other software components of the assistive device, and the device is equipped with several computing units, we adopted the design methodology to evaluate the execution time for different configurations.

Then, we will describe the design of a energy recovery system for data centers CPUs. The system scavenges the energy from the heat dissipated by a data center processor to power a smart heat sink. The smart sink is equipped with a microcontroller system with radio capability and a convention fan. According to the configuration, the smart sink provides two functionalities. It can work as a wireless sensor network node, to provide environmental data center monitoring by sending data it samples from its on–board sensors to a data collector. Alternatively, it can power the fan to provide active cooling to the data center CPU. The fan is activated following a strategy that allows the CPU to not overheat during a period during which the CPU is overclocked. With this system it is possible to improve the data center management, but the behavior of the smart sink depends on the amount of energy it can scavenges. Although the tasks of the data center CPU and of the smart sink are independent, they influence each other through a physical interaction.

1.2 Structure of the Thesis

This thesis is organized as follows.

Chapter 2 illustrates the state of the art in Cyber-Physical Systems design methodologies, and design tools. Moreover, it describes the main features of the design tool we have chosen for this research, and described the proposed methodology.

In **Chapter 3** we introduce the first case study: a auditive guidance interface based on 3D sound synthesis. We define the objective of the project, illustrate the behavior and technology of the interface. We illustrate how the design methodology has been applied to the case study and discuss the results we obtained.

Chapter 4 illustrates a second case study: the design of a thermo energy harvesting system that scavenges energy from high–performance processors to supply a very low–power device. The initial part of the chapter is dedicated to the description of the parts that compose the system and their characterization that allow us to get models useful to describe the system. The second part of the chapter illustrates the application of the design methodology to the design of the system and discusses the results.

Finally, **Chapter 5** concludes this thesis drawing conclusions on the work done, and outlining some possible future developments.

Chapter 2

Design Methodology

2.1 State of the Art

Several design methodologies have been presented in the literature and applied successfully to cope with the design challenges of CPS. In Platform-Based Design (PBD), the design moves from the application space to the architectural space through a mapping process [52]. The approach is based on the separate definition of the functionality of the system, and of the possible implementation platforms. The two are then combined by *mapping* the functionality on the different platform architectures to assesses the performance of interest in a structured way [18]. The PBD paradigm can be enforced using contracts [47]. In this case, the design methodology is carried out in two steps. The first consists in the definition of the cyber and the physical architecture, followed by an exploration of the target platforms. In our work, we follow the PBD paradigm and extend it to a two-step mapping process to separately specify and then combine cyber and physical components.

Model-Based Design (MBD) addresses the problem of designing complex systems starting from a model that represents the system behavior using numerical methods, models or measurements. Jensen et al. discuss the MBD approach in the context of CPS, by dividing the design process into steps, from physical modeling, to simulation, software synthesis and validation [29]. The authors exemplify the methodology with Ptolemy II, a modeling and simulation environment for heterogeneous embedded systems which supports MBD [12, 53]. Our approach is more structured, and defines precise roles for the architectural and the functional elements of the models. The METROII design tool, which we use in our work, has been also used in combination with Ptolemy II, originating the design tool for timing verification called *Metronomy* [23]. Also based on Ptolemy II, PTIDES focuses on the design of event–triggered real–time distributed systems, and includes a programming model and the relative toolchain that supports MBD [73]. The PTIDES approach combines the construction of a model with a simulation of the system and network, and estimates the behavior of the system mapped onto a target platform. The implementation platform introduces delays that may affect the execution of real-time systems. The tool integrates the simulation of Discrete Events –typical of cyber components– with different physical time models, to perform timing verification, and generate executable software. An alternative approach, proposed by Noyer et al., includes the adoption of three different design tools in the workflow: 1) a tool for timing analysis, 2) a tool that supports Model Driven Development, and 3) a tool for Requirement Management (RM) [46].

The *Matlab* design environment, which follows the MBD approach, supports CPS development through *Simulink* or *SimScape.* Here, simulations and implementations are coded in two different languages. On the contrary, we aimed to develop a design environment that can reuse the same C/C++ code for both implementations and simulations. An alternative approach to the one presented in this work is based on the adoption of *Modelica* [24]. *Modelica* is a powerful tool for modeling and simulation of mechanical, electrical, thermal, control and hydraulic systems. In its standard version, it does not provide simulation of computing systems, but since it is designed to be domain neutral, it is flexible enough to support the description of such kind of systems. Recently, a support library called *Modelica_EmbeddedSystems* was released, which represents an extension of *Modelica* from the physical domain toward the digital domain. A design approach that combines *Modelica* and METROII has been developed [54], but it keeps the two designs separated and makes them communicate via *CORBA*.

Another framework for building complex systems by composition of components is Behavior-Interaction-Priority (BIP) [8]. In BIP, the system specification is organized in three layers. The first layer describes the components in isolation, the second layer is used to specify the interaction of components activation through *connectors*, and the top layer is used to express the priorities that characterize the overall system architectures. To support CPS design, *BIP* can interact with *Simulink* physical models by using translators to convert *Simulink* blocks into *BIP* specifications [65].

To address the problem of CPS design, recent technologies are devoted to the joint modeling and simulation of domain specific software [19]. Since the intrinsic multidisciplinarity of the design process involves the study of concurrent, heterogeneous models, a design cannot be studied in an unique framework due to the different semantics in use. Some design methodologies involve the structured exploration of the cyber and physical worlds by using two different design tools, and then applying iterative algorithms to converge towards a solution that meets the design requirements [30, 3]. Other design flows use two different design tools and a new specification language to bridge programming languages that otherwise could not communicate. One example is the use of an Intermediate Format (IF) to exchange models between *Matlab/Simulink* and the C language [71]. Standards can alternatively be used for tool interoperability. The Functional Mockup Interface (FMI) is a tool-independent standard aimed to ease the collaboration in design projects where different tools and workflows are used. Using FMI, the designer distinguishes between the functionality of the component (code or binaries), and its interface data. Tools supporting FMI allow engineers to import models and co-simulate them to perform design validation, promoting the exchange of simulations model across tools [7]. Our methodology is orthogonal to this approach, which could be conveniently used to extend the integration capabilities.

2.2 MetroII Design Framework

A generic CPS can be seen a system where embedded computers can monitor and manipulate physical world, and the physical quantities can affect computation, usually with feedback loops. For a CPS engineer, it is important to consider and to model both the behavior of computation processes and the evolution of physical quantities. From the cyber side, for example, it is important to understand the time required to perform a task, the amount of memory required, or the physical size of the device. From the physical side, to model thermal trend, the mechanical, or to capture the behavior the system under control. The physical architecture under control can be composed of a hydraulic, electrical or mechanical components. However, the cyber and the physical world belongs to two different domain, in fact, most of cyber processes can be model in discrete time and evolve in a sequential manner; on the contrary the physical world is modeled as real processes that can be concurrent in time. Systems resulting from the combination of the two heterogeneous world exhibit a superimposition of the two domains, and must be modeled accordingly.

Generally, CPS properties are evaluated in terms of power consumption, cost, weight, size; but also robustness, accuracy of the results, and battery lifetime. Those quantities are influenced by architectural features of the design (such as the choice of components), and from the functional aspects (software complexity) concurrently. A CPS can usually be made of many devices, sensors and actuators. The cyber side can be made of many software components that execute on one or many processing unit. Operating systems schedule and coordinate events with different criteria.

2.3 MetroII building blocks

In our work, we adopt the METROII design framework [15]. METROII is a design framework which support PBD. It is written in SystemC which is a C++ library that provides an interface for event-driven simulation, meaning it allows a designer to simulate concurrent processes written in plain C++ syntax. A METROII system design is made of a *Functional* and an *Architectural* model, to promote reusability and separation of concerns. The Functional Model expresses what the design does in algorithmic terms, such as the control strategy. The Architectural Model describes how the system is implemented, the hardware platform where the control algorithm executes, including embedded processors, sensors, actuators, communication primitives, operating systems, firmware, and drivers. While the functional model can be used to perform control validation, the architectural model contains values and formula used to model the physical quantities of interest, to simulate physical time, power consumption, associated costs, and other features of interest.

In METROII, functional and architectural models are described as netlists of *components* that communicate through *ports*. A *component* is an object that contains imperative code and may contain processes. Processes are made of threads that can execute concurrently with other processes in the design. A component contains an internal behavior and an interfaces. The internal behavior of the component is used to specify the functionalities that it provides to the system. The interface allows a component to interact with others. A interface is made of ports. Each port is associated to a set of methods defined by the port interface. A port can be either *required* or *provided*. Required ports specify a set of methods that the component requires from other components. Provided ports specify the methods that the component implements and that it can perform when requested by others. For example, a functional component can contain the functionality of a piece of software. In the binaural guidance case, a functional component representing the audio processing algorithm contains the code to process 3D sound. For simulation purposes it is possible to remove the functionality code. Typically a functional component has a component in the architectural part that represents the functional component functionality as an executable thread of the architectural model. This component is used to capture the execution of the functionality into the architecture. An example of an additional architectural component can be a microprocessor, or a memory. The microprocessor is used to model the hardware component of a system, in this case the microprocessor that can execute the architectural thread. The microprocessor component can contain the functions that other components require it to perform, and those functions are exposed to other component through a provided port. If the processor requires access to the memory, it must contain a required port to interface to a memory architectural component.

The basic behavior of components is initially specified in *SystemC*, then encapsulated by wrapping the code to provide METROII interfaces. The components of a METROII system are concurrent, and synchronize through events. Each method is associated to a pair of *begin* and *end* events, which mark the start and end points of the procedure execution. The execution of a METROII system is orchestrated by an *event scheduler*, which triggers the methods of the components. Additional events are used to characterize *quantities*, which model non-functional aspects such as execution time and power consumption. *Annotators*, which are similar to event schedulers, are used to label quantities with values corresponding to the event triggering, providing the means to conduct performance analysis. In particular, end events are associated with tags expressing time the annotator uses to evaluate the execution time in the target architecture. The event scheduler can be used to specify the order of execution of operations. Moreover, through the event scheduler it is possible to specify if a component can execute only a function at a time or, on the contrary, if it can execute more operation concurrently.

Event synchronization is specified through *constraints*. In particular, architectural components are synchronized with the functional components through *mapping constraints*, which schedule functions and their respective architectural implementations (tasks) simultaneously. This mechanisms implements a function/architecture co-design paradigm: functions synchronize with the architecture, which progresses according to the modeled quantities, to produce an estimate of the system performance. Likewise, abstract functional communication primitives, such as a blocking FIFO, can be synchronized to an architectural implementation, such as a shared memory.

2.4 Proposed Design Methodology

The building blocks provided by METROII to model ESs can be used to model CPSs. To model the cyber part of a system, designers implements the typical workflow for embedded systems design. This consists of modeling the functionality of a system in the functional model, and defining the architecture that can execute the functionality in the architectural model. Then, the tool provide a matching of the functionality onto the architecture to perform the functional-architecture co-design.

In this thesis, we extend the function/architecture approach to perform cyber/physical co-design, adopting the procedure illustrated in Figure 2.1. We follow a paradigm similar to the function/architecture co-design approach. However, we first decompose the computational and the physical part of the system under consideration. Since physical quantities are handled and modeled in the architectural part of the system, to model the

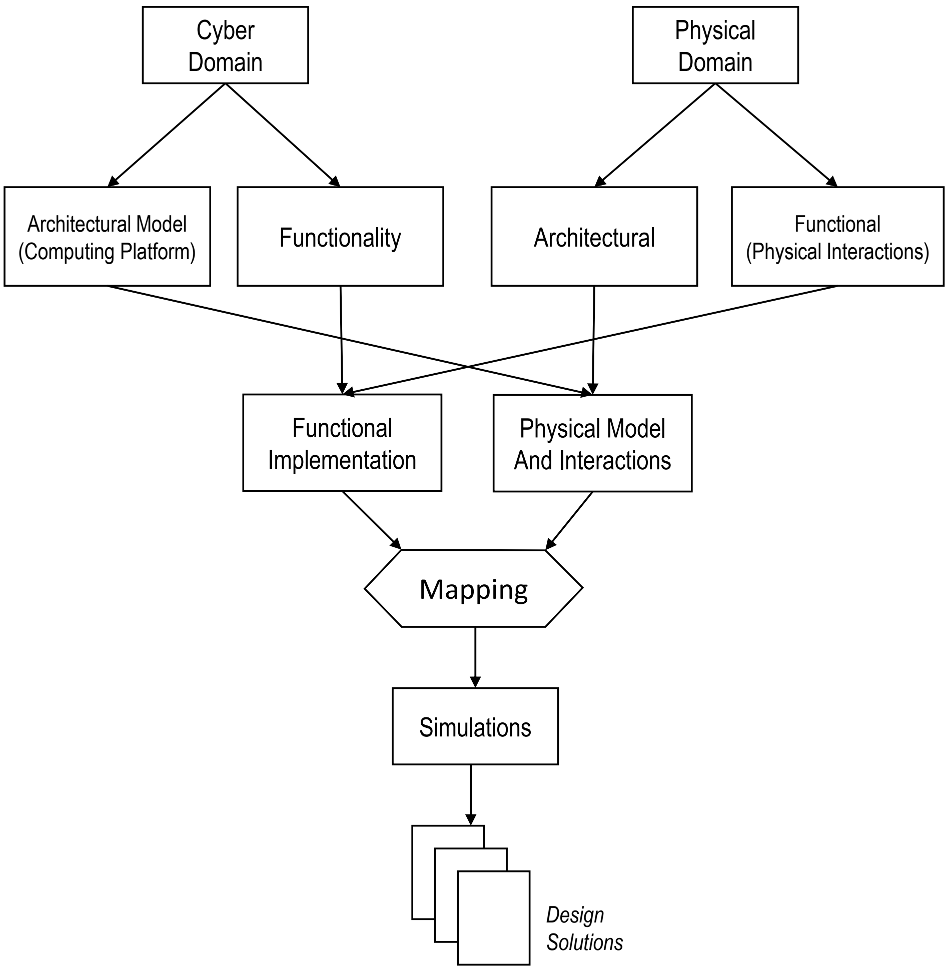

Cyber Domain
Physical Domain
Architectural Model (Computing Platform)
Functionality
Architectural
Functional (Physical Interactions)
Functional Implementation
Physical Model And Interactions
Mapping
Simulations
Design Solutions

Figure 2.1: Proposed workflow to model cyber-physical systems by separating functional and architectural models of the computing and physical parts of the system and conducting the simulations with the same design environment

physical part of a CPS the designer focuses on the architectural model of the system first. Mathematical models that describe the physical behavior of the system are implemented in C/C++ inside the architectural components. These models may involve differential equation solvers (e.g., the Euler method), when a discretized continuous time description is required. Interfaces are used to forward physical quantities to other components according to the interaction that exists between architectural elements, whether they represent a physical or a computational process. In this phase, the designer analyzes physical interaction between architectural components to identify the component organization. The interaction is implemented through ports. Then the designer analyzes the functional model of the physical part, to identify components that interact with the functionality of functional functions. A physical process that influences (or is influenced by) the behavior of the functionality of the system is modeled as a functional component. A special provision must be made if a physical process is defined only in the physical functional model. In that case, the designer must include also a virtual (or counterpart) component in the physical architectural model, to allow the *mapping constraints* to schedule and capture its execution, even if the component is isolated from other components from the point of view of the architecture.

In the architectural model of the design, the message exchange between components is used to connect provided and required objects as in traditional ESs design. In addition, interfaces are used to forward physical quantities between components that physically interact. For example, to model how the temperature of a component A influences the actions of another component B in one of our case studies, component A contains a required port to a shared memory, and B has a provided port to the same memory. The interface is used by B to receive temperature values from A.

Once the design is completed, the overall functional model contains the functionality of the computing part of the system, with components and ports that model the interactions of physical processes with the system functionality. The architectural model contains the definition of the computing platform together with the physical models and interactions between physical and computing worlds. The METROII scheduler is used to coordinate the execution, the same way as in the design of embedded system. However, events that are triggered by physical processes (i.e., a value reaches a threshold) are ignored by the METROII scheduler, but are fired from direct, concurrent function calls performed by an architectural component. In this way it is possible to define the order of execution of code portions according to the behavior of physical quantities that is not known before execution. In fact, the events logical ordering handled by the scheduler cannot be conditioned in runtime. Generally, the *physical time annotator* is used to evaluate the execution time of a design. To simulate the evolution in time of physical quantities, the *annotator* is used to define the time resolution of the simulation. Since the physical time annotator is used to tick the simulation time, and the logical sequence of events may change depending on the design, if the designer wants to monitor the time between two generic events the developer must insert additional code to extract time values. Designers can use the tool to evaluate the features of many implementations mapped into several architectures to identify the solution that best matches the application constraints. At that point, having studied the design by separating the functional and architectural aspects of both computation and the physical domains, the selected solution can be implemented into the target architecture reusing the functional code.

Chapter 3

Audio Guidance Interface

In this chapter we describe how the proposed design methodology has been applied to the design of a auditive guidance interface developed for a smart walker. The first part of the chapter is dedicated to the description of the implementation of the interface, while the second part focuses on the description of the design model. Subsequently, we illustrate the analysis of the quality of the generated sounds from a technical viewpoint, and an experimental evaluation in which the acoustic guidance is compared with other type of interfaces implemented in the smart walker. The smart walker we considered mounts several computing units and several software components are needed to properly implement the functionalities. The software components can be executed from any of the computing devices, so it is possible to implement different combinations of hardware and software that will exhibit different performance. The objective of our research work is to identify the combination that allows us to obtain a guidance signal effective and pleasant to hear for the user. For this purpose, we have extended the functional architecture analysis to accommodate the presence of the user in the design model.

3.1 Context of the Project

Medical literature suggests that reduced mobility of older adults can have a detrimental effects on their health, and accelerates the process of aging [69]. Reduced mobility can be caused by physical or cognitive impairments, and people with reduced ability may gradually perceive daily activity inside crowded places as intimidating [33]. Therefore, to tackle the problem of aging, technology must provide a tool that permits the older adults to do physical exercise, keep the mind active, and get over everyday challenges across public space. To provide a physical and cognitive support to the older adults, researchers of the *DALi* project have developed a robotic device called the *c-Walker*. The device is based on a four-wheeled rollator equipped with sensors, computing boards, actuators and interfaces to provide navigation. Main goal of the project is to provide elderly with a tool that helps them to diminish the stress of carrying out daily activities, therefore benefits of social life and slow down the aging process.

While the mechanical (or physical) part of the *c-Walker* serves as physical support for people suffering from weakness of the lower limbs, its cyber part overcomes the issue of orientation, memory attention, and understanding of places by providing guidance stimuli to the user. Overall, the *c-Walker* main features are:

1. the identification of the path across the space that best supports the user preferences and goals;
2. self localization inside the environment;
3. the detection of anomalies along the way, and the possibility to locally reshape the path accordingly;
4. provision of guidance to the user with active, and passive user interfaces.

The system has been designed to work in indoor environment, for example in shopping malls, airports, hospitals, or medical centers.

The user of the *c-Walker* selects the places he/she wants to visit using a touchscreen mounted between the handlebars. The system computes the optimal route using information stored in a map database. When the user starts walking, the *c-Walker* use different sensors to localize itself inside the environment, and compares its real–time position with the planner route. Planned route may change along the way according to the dynamic of the scene. If the user does not deviates from the planner route, the system does not provide guidance, and behave as a conventional walker. On the contrary, if the user deviates from the optimal route, the system intervenes by actuating the User Interface (UI): the touchscreen, a pair of wearable haptic bracelets and a headphone. If the user deviates significantly, the system intervenes using active actuators: the motorized steering front wheel and the brakes on rear wheels.

3.1.1 System Functionalities

The data processing flow of the *c-Walker* can be summarized in three steps. First, environmental information is captured using sensors, stored and processed on embedded boards. Then, the data is processed to decide the best route. Lastly, the system communicate to the user information about the route through different media.

The first step features different components: system localization, people detection and tracking, detection of person of interest and anomalies, map representation. Those components sense the environment by means of low level sensors (i.e. wheel encoders, Inertial Measurement Unit (IMU), RFID reader), and high level sensors (camera and depth sensor). To process the best route, the system includes a long–term and a short–term planners. The long–term planner finds a path that connect the desired destination with the current location inside the map, satisfying user preferences. The short–term planner utilizes real–time information, such as the presence of anomalies or crowded areas. The short–term planner introduces a “local” modification of the long–term plan required to react to potential risks detected using cameras and depth sensors (i.e. Kinect).

The UI includes: a visual interface, haptic bracelets, acoustic interface and the mechanical guidance. The visual interface shows information of navigation via the touchscreen by drawing a map of the surroundings, highlighting the current position, the proposed path and showing an arrow pointing to the direction the system suggests to follow. The touchscreen is also used to take the user inputs. The haptic and acoustic interfaces raise the user attention in correspondence of turns or significant deviations from the optimal trajectory. To signal to the user to take a left turn (or a right turn), the haptic bracelet worn on the left forearm (or right forearm, in the other case) vibrates. To indicate to the user to turn left, the headphones reproduce a positional sound in the corresponding location. To synthesize positional sound we implemented different software, including a binaural rendering algorithm based on sound propagation model. Using these interfaces, the *c-Walker* has just an assistive role, in fact, the user is in charge of the decision about the route to follow, and can override the device suggestions. We refer to the acoustic and haptic interfaces as *passive interfaces*. On the contrary, the mechanical guidance plays an *active* role, since it can force a change of the direction.

3.1.2 Related work

Robotic walkers have gained an undisputed popularity in the research community on ambient assisted living [34, 66, 37]. Closely related to *DALi* is the Assistants for Safe Mobility (ASSAM) project [4], where a system consisting of a set of modular navigation assistants deployed on several devices is used to encourage physical exercise. ASSAM has a major focus on the seamless transition from indoors to outdoors, while *DALi* specifically considers large indoors environment. Also, the behaviour of people in the surroundings is not considered in ASSAM.

The iWalkActive project [27] is more directly focused on the development of an active walker that facilitates and supports physical exercise, both indoors and outdoors. E-NO-FALLS [20] is another project focused on the development of a smart walker. In this case the main emphasis is on preventing falls. These two projects deal with physical aspects related to mobility of older adults. Although of interest, these aspects are complementary with those of *DALi*, whose main focus is on fighting cognitive decline.

3.1.3 Author Contribution

In the context of the *DALi* project, we focused on the design and development of the audio interfaces. To navigate the *c-Walker* user through sound stimuli, we decided to adopt an interface based on the synthesis of 3D sound to be reproduced via headphones. We also developed an alternative guidance system in which the direction information is encoded as a difference in amplitude of the left and right channels, and the amount of turn is encoded as the amount of stimuli in the unit time. A description of the sound synthesis software implementation for rendering spatial sound on the horizontal plane has been published in [58]. In particular, the paper addresses the evaluation of the computation time required to generate spatialized sound for two different versions of the model on two computer architectures (a laptop and an embedded platform). A deepen analysis of the sound synthesis algorithm, accompanied by a description of the reverberation algorithm, and an initial evaluation study performed through an on–line questionnaire and results has been published in [59]. An exhaustive description of the *c-Walker* can be found in [57] and in [49]. Those two articles include an explanation of all the guidance interface including the one based on binaural audio. The description of an experimental study designed to evaluate the performance of user traveling across a space using one of the different guidance interfaces and the analysis of the results can be found in [43]. The personal contribution of the author is related to the binaural guidance interface and the different audio interface implementations used for the experimental evaluation.

3.1.4 Chapter Structure

The rest of this Chapter is structured as follows. In section 3.2 we introduce the role of the auditive guidance interface and how humans can judge the position where sound originates. Reflecting the human ability of recognizing spatial sounds in real world, we then describe how to reproduce the same sensation via headphones using adequate algorithms. The propagation model (described in detail in section 3.2.2) we adopted combines monaural cues, useful to recognize the left/right displacement of sounds, with binaural cues, most useful to judge the elevation of sounds, and a reverberation algorithm to control the sensation of immersion. Section 3.3 is devoted to the description of the design exploration carried out using the proposed methodology with the design environment METROII. In Section 3.10 we describe an implementation details that allows the algorithm to handle the delay introduced with the propagation model without causing disturbances in the generated signal even in presence of moving sounds sources. A comparative study demonstrate the advantages of the approach we adopted in particular in presence of the Doppler effect. Finally, in Section 3.12, we summarize results of an evaluation study we conducted to estimate the performance of the system in guiding non–disabled people across an unknown environment.

3.2 Acoustic Interface

The primary role of the *c-Walker* is to convey information about the environment to the user through haptic, visual and acoustic interfaces, assisting the user to follow a path or to move within a safe area avoiding obstacles. Besides the aforementioned passive interfaces, the *c-Walker* includes active motion control implemented through front caster wheels and brakes on the back wheels. To improve user's comfort during movements, and ensure the user remains in control of the device, active guidance is activated only when the user approaches the boundaries of an area, or an obstacle. While the user is within the correct path, only passive interfaces are active.

Traditional acoustic navigation systems provide the user with instructions in the form of speech, such as "turn left", or "go straight". The limitation of this approach, especially for those who are visually impaired, is that the system does not an enhanced spatial representation of the surroundings. Our goal is to provide informations that can be easily interpreted to make an independent decision regarding the path to follow. Therefore, we have followed a different approach in which the audio stimuli is synthesized to convey to the user spatial location of the safest path and to stimulate higher level of cognition of the listener. This goal is achieved by processing signal with 3D sound techniques, which make sounds reproduced over headphones to appear as if they are originating from a precise point in space. In this way, sounds can be used to direct the attention of the user towards the path to be followed.

The ability of humans to interpret the position in space where a sound originates is called *localization*. In real world, localization relies on the body shape and displacement of the human ears. In particular, the external ears behave as directional sound filters. Thus, they introduce modifications of the incoming sounds, resulting in differences in amplitude and phase at the two ears (binaural cues), and spectral cues (monaural cues). The brain interprets the cues to determine the direction of arrival of sound waves. By reproducing the same phenomena involved in this natural process using signal processing technique, it is possible to generate synthetic sounds that give the sensation of distance and direction even if the signal are heard through headphones.

Many approaches can be used to create three-dimensional sounds. The most accurate technique is to measure the impulse response of the human hearing system for many position in space, and to determine the overall transfer function of the listener, known as the Head-Related Transfer Function (HRTF). However, HRTF measurements can be expensive, they require high quality speakers and microphones. Moreover, it is required to build a mechanical setup to hold the instrumentation in precise spatial points. Preferably, measurements are performed in anechoic chamber to minimize noise and reverberation, and recording sessions takes a lot of time during which the listener is required no to move his head because head movements would produce a loss in accuracy of the collected data. Even if several free databases of HRTFs exists, the impulse response are highly subjective because they are based on anthropomorphic features that change from person to person.

Another approach is based on modeling the sound propagation phenomena and synthesize HRTFs based on anthropometric information. This gives us more flexibility, since we are not tied to discretized measurement point as happens for measured HRTF, and we can tune the model parameters to fit the individual user. This solution is also convenient for an embedded implementation because it is considered faster to compute. This aspect is very important, given that the sound position must be updated in real–time as the user moves in the environment. Moreover, the interface delay must be limited to ensure a correct interpretation of the guidance stimuli while the user moves his/her head. This approach makes possible to synthesize a sound that provide a continuous reference point that the user can follow. Alternatively, the sound may be activated only when the user is requested to change direction, contextually with the haptic feedback implemented on the *c-Walker*.

In order to render more realistic sounds, and especially to counteract the perception of inside the head locatedness that afflicts synthetic 3D sounds, we introduced some amount of reverberation. This is accomplished using the Image Source Method which consist of replicating the sound as virtual images resulting from the reflections of the sound waves on the walls. The replicas are attenuated to account for the loss in energy due to the absorption and the longest path with respect to the direct sound. More complex reverberation algorithm are not necessary. Not only they are more computationally intensive, but it has been shown that only the first order reflections are useful to help with the localization process, while heavy reverberation on the contrary may cause confusion to the listener.

Likewise, localization can be considerably improved by tracking the position of the listener head, and adjusting the sound source virtual position accordingly. This is implemented by mounting an inertial platform on the headphone arches, which provides real–time information on the head orientation. The data is easily incorporated in the computational flow of the system by simply update the frame of reference. The improvement in localization is due to the ability of humans to integrate information in time as the head moves, and it is particularly useful, especially to reduce the front to back confusion that may arise with 3D sound engines. Moreover, humans use small head movements to discriminate ambiguous sound position in real world, therefore it is useful to recreate the same possibility even in the synthetic guidance interface to make it more natural.

3.2.1 Rendering spatial sounds

As discussed in the introduction of this chapter, the most accurate method to recreate 3D sound is based on measured HRTF, or their Fourier transform (HRIR). HRTFs represent the response of the human ears and body for a given direction of the incoming sound and individual anthropometry. Because this response changes from person to person, HRTFs have to be measured for each individual. Nonetheless, there exist databases of freely available HRTFs for research purposes [1, 70, 16].

The use of this method requires large memory to store the filter coefficients and the resulting filtering process is computationally demanding. Another disadvantage of HRTF–based sound rendering is that it lacks the sensation of distance and movement; therefore these effects have to be separately implemented by means of proper algorithms (e.g., Doppler effect, reverberation) [13, 22]. Moreover, measure responses are obtained in discrete spatial locations around the listener. To spatialize sounds in intermediate positions that were not measured, there is the need for techniques to estimate HRTFs by interpolation of adjacent contributions. Therefore, additional processing would introduce delay and approximations.

For mobility aids, a high accuracy of representation is not required, but a sensation of spaciousness and displacement must be provided. Our work is also motivated by the intent of lowering computational needs, in order to permit the implementation on an embedded system. For this reason, we have adopted a model-based approach in which we reproduce the sound wave propagation. The goal is to create a model with a reasonable good spatial resolution, and a low complexity. This also avoids having to measure the actual response.

Another disadvantage of measured HRTFs is that they include reflections that originate from the shoulders and the chest. Those parts of the body are far from the ear channel. Therefore, the echoes associated to body parts echoes arrive at the tympanic membrane with a large delay. As a consequences the measured responses that include shoulder contribution are longer in time (up to 200 taps at 44.1 kHz) making the filter processing slow and memory hungry. Generally, measurements are carried out with the listener always facing forward. Consequently, contributions from shoulders and chest are fixed at that position, and does not account for head movements. Meaning that, if the user tilts or rotate her/his head, the sound generated by the HRTF-based audio interface rotate accordingly, but conserves a contribution that does not rotate, and therefore may introduce error in the localization task.

3.2.2 Propagation Model

To provide an authentic spatial representation, sounds must be processed so that they give a sensation of direction and spaciousness of the scene. The sensation of direction depends on the relative position between the listener and the sound event, and on the listener's anthropometric features. The sensation of distance and spaciousness in the scene can be rendered by combining two methods. The primary cue of range comes from the amplitude of the incoming sound, which decreases linearly with the distance. However, this cue alone is not sufficient for accurate range judgments, and humans in daily life take advantages of reverberation cues in solving the task of range recognition. Reverberation depends on the room size, its geometry, the materials on the surfaces and the presence of other objects. Contrary to directional cues, this aspect is not subjective, so there is no need for individualization.

3.2.3 Interaural Time Difference

Analyzing how HRTFs are composed, it turns out thy are the result of the superimposition of multiple effects. Thus, it is possible to create each effect separately and sum the contributions to generate an artificial sound that contains all the cue to obtain a convincing effect. We first analyze the left/right displacement of sounds. One of the major cues for sound localization is the Interaural Time Difference (ITD) that is the transient time difference between the two ears due to the differential distance between ears and the sound source.

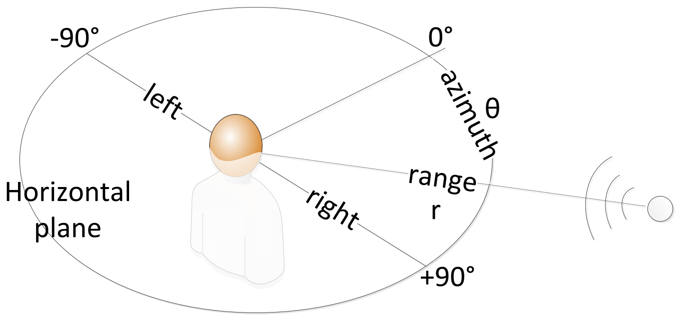


Figure 3.1: The vertical polar coordinate system.

We use a polar coordinate system as shown in Figure 3.1. The signal on the contralateral ear channel is delayed to account for the interaural time difference ΔT, given by

$$\Delta T = \frac{a}{c}(\sin\theta + \theta), \tag{3.1}$$

where a represent the head radius of the listener, c the speed of sound in normal condition and θ is the azimuthal angle, which takes value 0 in front of the listener, negative values on the left hand side and positive values on the right. For an average human head radius of 8.75 cm, the maximum delay at $\theta = \pm 90°$ is 660 μs. With a sampling frequency of $F_s = 44.1$ kHz, the maximum delay corresponds to about 29 samples.

3.2.4 Interaural Level Difference

Another cue for the localization of sounds lying on the horizontal plane is the Interaural Level Difference (ILD) or Interaural Intensity Difference (IID). In fact, sounds impinging the ear coming from different paths are attenuated according to the different length of the path, and due to the presence of the head of the listener. The effect of the path is easily inserted in the model considering the attenuation in amplitude is equivalent to the inverse of the path length. The shadowing effect introduced by the head is frequency dependent. However, this effect can be modeled as a single pole/single zero filter.

The interaural level difference is accounted by the shadowing effect introduced by the listener's head, computed using a low pass filter on the signal of the contralateral ear, with the transfer function:

$$H(\omega, \theta) = \frac{1 + j\frac{\alpha\omega}{2\omega_0}}{1 + j\frac{\omega}{2\omega_0}}. \tag{3.2}$$

The coefficient α, which depends on θ, controls the location of the zeros and is defined as:

$$\alpha(\theta) = \left(1 + \frac{\alpha_{min}}{2}\right) + \left(1 - \frac{\alpha_{min}}{2}\right) \cos\left(\frac{\theta}{\theta_{min}} 180°\right)$$

with the values $\alpha_{min} = 0.1$ and $\theta_{min} = 150°$ to produce a convincing approximation of a spherical head model [10]. In our implementation, the software computes for each given value of θ the correct coefficients that model the frequency dependent low pass phenomenon introduced by the listener's head. This model does not account for the reflection of sound on the shoulders, which are typically included in measured HRIRs, and which would require a larger number of taps. Measured HRIRs, however, only consider a listener facing forward, and are not accurate when the head is turned or tilted. Thus, not including these reflections in the model makes it less computationally demanding and excludes contributions that may introduce confusion when the listener head is not facing forward.

To summarize, ΔT represents the ITD while the ILD is expressed by the transfer function $H(\omega, \theta)$. The first is the primary cue for localization of sounds below 1.5 kHz while the latter is more prominent for higher pitch tones.

3.2.5 Rendering Sensation of Elevation

While the left/right sensation is based mainly on differences in amplitude and time at the two ears, the sensation of sound elevation is a monaural cue. To recognize the elevation of a sound source, the human brain bases the judgment on reflections that originated from the interaction of impinging sound wave with the external ear or pinna. In fact, the folds of the pinna introduce some "echoes" that depend on the direction of arrival of sound waves. To model the effect of the pinna reflections, the software introduces six attenuated and delayed replicas of the direct sound. The elevation angle, ϕ, takes value $\phi = 0$ for sound sources lying on the horizontal plane, and positive values for locations above the horizon. In this model, the components due to reflections that originate from the interaction of sound waves with the shoulder are ignored. In particular, the time delay for the n^{th} replica depends on both the azimuthal and elevation angles, and it is defined as:

$$\tau_n(\theta, \phi) = A_n \cos(\theta/2) \sin[D_n(90^\circ - \phi)] + B_n$$

where A_n represents the amplitude of the replica, B_n is a time offset, and D_n is a scaling factor. All values are expressed in number of samples at F_s. To implement the algorithm, we use the model values from the previous work done by Brown and Duda [10]. Said values have been obtained by experimental measurements of three subjects, and from the empirical study it has been shown that only the variable D_n must be adapted to the individual listener. We included in our sound synthesis software the model for a generic head including the common model values reported in the literature. In Table 3.1, we report variable values for the calculation of the time delay (τ) and the reflection coefficient (ρ_n) for the n replicas.

Table 3.1: Pinna model coefficients

n	ρ_n	A_n	B_n	D_n
2	0.5	1	2	1
3	-1	5	4	0.5
4	0.5	5	7	0.5
5	-0.25	5	11	0.5
6	0.25	5	13	0.5

3.2.6 Reverberation

Reverberation has the effect of decorrelating the signal at the two ears, and makes the sound more natural. Therefore, it increases the sensation of immersion and spaciousness, and reduces the "inside the head" phenomena, typical of dry sounds, that causes the sensation of perceiving the sound as coming from inside the listener's head instead of coming from the surrounding space [5]. There exist many reverberation techniques [42]. Many of them are very complex and are able to produce at the output an effect very similar to the reverberation of the real room [31]. Unfortunately, this translates into a time demanding and memory hungry implementation. Moreover, these reverberation algorithms are parameterized on room geometry, size and materials but do not take into account the position of sound source and observer inside the room. This means they do not increase the sensation of locatedness. Our choice for reducing the computational complexity of the system and in the meantime taking care of relative positions of sound elements in the room is to adopt the so-called Image Source Method (ISM) [9]. In the ISM each wall is equivalent to a reflective surface. A sound wave reflecting off a wall is equivalent to having a virtual source on the mirror image of the original, behind the wall (see Figure 3.2). In our work a simple ISM technique is implemented, considering only four contributions inside a virtual rectangular room whose dimensions are proportional to the distance of the farther objects to be rendered. This means that the resulting sound is made by the superimposition of five virtual sound sources.

3.3 Modeling the Audio Guidance Interface

The *c-Walker* is equipped with several interconnected computing platforms including a Beaglebone, a Beagleboard, and a laptop. The software application is made of several independent modules that carry out the different tasks that compose the overall application. Some of the modules can be active or disabled according to the functionality of interest. For example, the software that perform the guidance using acoustic signals can work without activating other guidance interfaces, or can be active when also the haptic guidance is active, or can be switched off is it is not needful. As a consequence, the computing requirements of the overall application may change according to which of the modules is enabled. Each application module can be mapped in any of the computing platforms, and the performance (i.e. the execution time) change accordingly. The objective of this analysis, is to use the design environment METROII to evaluate the execution time of the application to identify the optimal mapping of the software components into the computing platforms.

In this work, we focused on a subsystem of the *c-Walker*: the user audio interface [58]. The auditive interface software provides binaural sound, processed to be interpreted by the listener as coming from the point in space where he/she has to travel to. The interface software relies on data coming from the *c-Walker* regarding the spatial location of optimal route, and on information on the user's head orientation captured using a IMU mounted

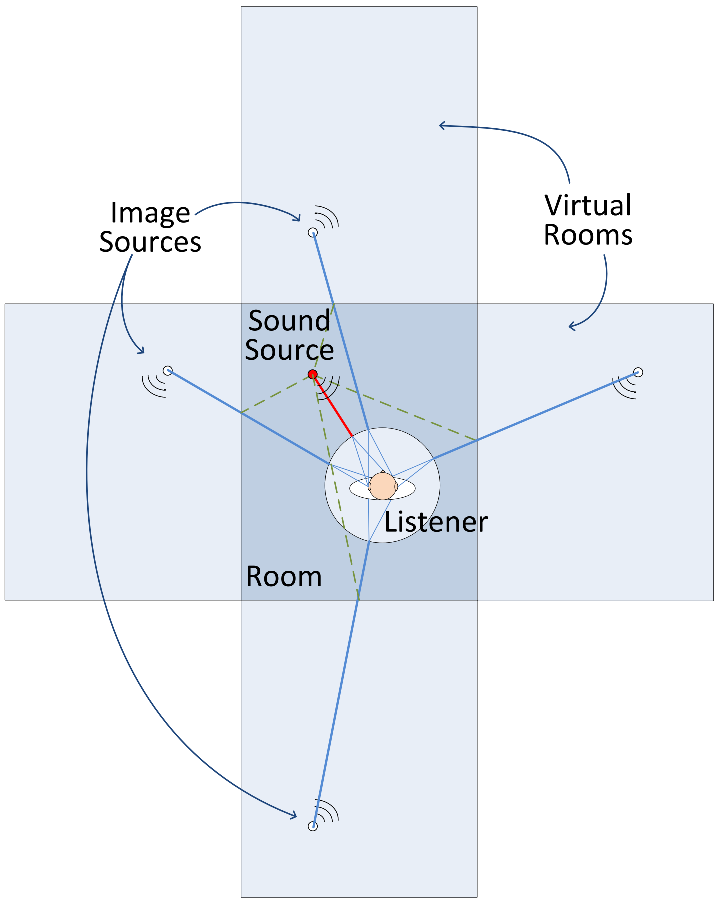


Figure 3.2: Intuitive graphical representation of the Image Source Method implementation.

on the headphone arches. The auditive guidance signal are processed using the algorithm described in previous sections.

It is important to emphasize that a person can correctly interpret binaural sound signals only if they are administered with a time delay smaller than 50 milliseconds, otherwise the human localization task may infer the perceived stimuli are associated to sound events that happens later than expected by the algorithm. For example, the human ability of recognize spatial location of sound improves if the listener is able to benefit of small head movements. In fact, moving the head makes it possible to collect more cues regarding an ambiguous signal (such as for example in case of sounds located within the so called cone of confusion) [6]. But, while the listener is moving her/his head, if the synthesized sound is emitted with a delay greater than 50 milliseconds, the sound may be associated with the final position of the head instead of being associated to the position where the head was initially. The interaction with the user (meaning the emission of stimuli and the reaction of the user which results in movements) closes the loop of the cyber part of the guidance system. Therefore, the time delay of the auditive interface is crucial to its

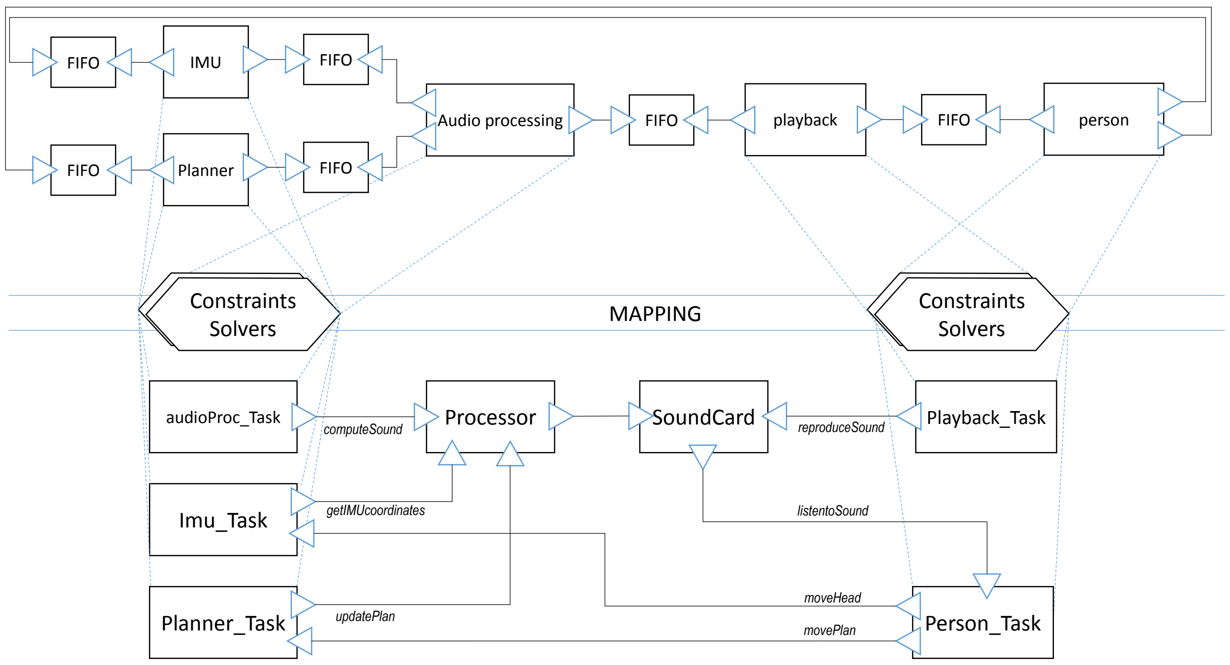


Figure 3.3: The METROII model for the binaural guidance system.

efficacy. Moreover, the audio processing and the planning algorithms have their periodicity and execution time which depend also on the computing platform. The the composition of the two software module into the target architecture can have performance such as to make the continuous reproduction of the guidance stimuli possible or not possible. We mapped several platform to identify the combinations that allows the system to reproduce the guidance stimuli without interruptions.

The general block diagram of the complete system is shown in Figure 3.3. Referring to the proposed methodology, we created a cyber model consisting of all the software components that express the functionality, as well as their corresponding architectural tasks. In addition, we have included a *Processor* and the *SoundCard* as architectural components to represent the computing platform. The physical part is needed to describe the physical time and the movements, or rather the variation of the spatial coordinates. The movements of the users are described by the *Person* component. To reflect its interaction with other components, and to map it correctly in METROII, the *Person* component is included in the architectural as well as in the functional model. The functional components are extended to include the description of physical interactions. In case a designer aims at modeling the precise human behavior [44], details regarding sound localization performance and degree of cooperation of the human can be implemented within the *Person* architectural component.

3.4 Physical Model

Since physical quantities of interest (i.e., time) are handled inside the METROII architectural model, physical model components belong to the architecture. The *audioProc_Task*, the *Imu_Task*, the *Planner_Task*, and the *playback_Task* have a dual role: they represent the architectural component of their functional counterparts, and they receive, modify, and forward physical data. The physical model includes the presence of the user of the *c-Walker* that is represented by the thread implemented within the *Person_Task* component. The presence of this component models the interaction of the user with the binaural guidance system. From one side, the user receives the guidance sound stimuli which convey spatial information encoded as the spatial location of the virtual sounds. On the other hand, the user movements change the position in space of the assistive device, and these variations are captured by the *Planner_Task*. The person can also move his/her head independently, and this is monitored by the *IMU_Task*. The physical model is completed including the interfaces to communicate with the *Processor* and the *SoundCard*.

From a functional viewpoint, the interface between the *playback* and the person transmits the sequence of samples representing the sound signal. But, from the physical point of view, the interface between the *SoundCard* and the *Person_Task* component models the exchange of spatial coordinates. In order to model the presence of the user, and to model the feedback between the sound reproduction and the subsequent movements of the user, a functional *Person* component is added to the cyber model of the system. Within the *Person* component it is possible to map the *Person_Task* behavior to close the loop of the system output toward the inputs.

3.5 Cyber Model

The cyber model consists of the software components of the binaural guidance system, and the hardware components that support their execution. In particular, the *Audio Processing* component contains the algorithm that transforms a monaural audio signal into a binaural stimulus. The computation works on a period buffer which is then fed to the sound card to be reproduced. The size of the period buffer matters, because it influences the latency of the computation. Once a number of samples of audio output equal to the *SoundCard* period buffer size has been produced, they are forwarded to the *playback* component. The *playback* component models the reproduction of the audio signal.

The *Audio Processing* component receives inputs from the *IMU* and the *Planner*. The *IMU* provides data regarding the orientation of the head of the user. The *Planner* computes the optimal trajectory for the user, and by knowing the actual position of the *c-Walker*, provides to the *Audio processing* component the spatial coordinates of the next position to reach, i.e., the spatial location of the virtual sound source the audio processing component has to compute in the subsequent time interval. The *Audio processing* aligns the absolute spatial coordinates according to the user head orientation and starts processing sound. The cyber model is completed including a *Processor* and a *SoundCard*. The *Processor* supports the execution of all the software components,

3.6 Functional Model

The functional model we considered includes the following components:

IMU the communication with the Inertial Measurement Unit, used to monitor the listener's head orientation;

Planner which represents the algorithms used to determine the suggested trajectory, this calculation depends on the planned path, the real user trajectory and dynamics of the environment.

audioProcessing this module represents the audio synthesis algorithm, and it is used to encapsulate the time required to synthesize a chunk of sound;

Playback representing the reproduction of the auditive stimulus, and it lasts for the same amount of time of the audio chunk size.

Person represent the presence of the user/listener; this component closes the loop from the Playback and the pair IMU and Planner and the person encloses the physical part of the system under investigation.

The IMU and *Planner* components are connected in parallel, because they can be executed independently; conversely, the other functional components are connected in series as being logically sequential. The IMU component model the requests of data from the inertial platform mounted on the headphone arches. If the system requires just the yaw angle, the time required to get the coordinate is lower with respect to the time required to get the coordinates in all the three directions (yaw, pitch and roll). In case the system implements the stereo guidance, instead of the binaural guidance, the IMU can be disabled, therefore the communication time is neglected. The planner component model the tasks executed to self localize the *c-Walker*and update the short term planner to decide the best trajectory for the user. A projection of the current location toward the planned trajectory determine the spatial coordinates of the virtual sound source location. The audioProcessing component model the execution of the algorithm described in the previous sections used for the generation of the navigation stimuli. The playback model the transfer of data from the memory location where samples were saved from the audioProcessing component, to the sound card buffer for the reproduction. The computation of the projected virtual sound source, the processing of the audio, and its reproduction follow a dataflow model, with the person movements expressed as consequence of the stimuli he/she listens to, and captured as inputs of the virtual sound source new coordinates. All the functional components communicate with each other through corresponding *FIFO* elements. The scheduler controls the execution of the system by firing events according to the availability of resource and connections of the elements.

3.7 Architectural Model

The architectural model of the system is composed of:

Processor : the processor is used to represent the execution element of the cyber part of the system;

SoundCard : representing the interface between the processor, which synthesize the sound, and the listener component.

In addition to the components listed above, the architectural model includes a *Task_component* for each one of the functional components (for a total of seven architectural components), to capture the implementation of functional components and map it into the architecture. Architectural components communicate each other through shared memory. The *Processor* features influences the amount of time required for the computation and the availability of memory. In fact the Processor provides all the functions required from other components and the tag values associated to the functions model the execution time required for the function on the target architecture processor. Moreover, to simplify the design the processor component models also the memory. Alternatively, it is possible to design a model with a component to model the processor and another component to model the memory. The *SoundCard* buffer size determine the maximum amount of memory available for the playback, therefore the reproduction delay. The architectural model includes tasks that are architectural representation of functional components (implemented as threads), used to map functional elements towards the corresponding architectural element (or elements) that can execute the required functionality.

3.8 Mapped Systems

We have estimated the performance of the system onto two different architectures. The first is a traditional laptop with a 2.53 GHz dual core CPU and 4 GB of RAM, to have a baseline performance model on a relatively high power device. The second platform is BeagleBoard–xM, equipped with a 1 GHz ARM Cortex–A8 core and 512 MB of RAM, which we have considered as representative of a class of small size, low power devices. Because of its small dimension, the BeagleBoard would be preferable if the system were to be engineered to be wearable, rather than being carried on the *c-Walker*.

The mapping of functional onto architectural components is labeled with execution time values, which were measured on the target platforms through profiling. Besides the two platforms, we also mapped four different configurations for the Planner, which differ in terms of the minimum dimensions of the grid used to represent the environment map. This affects both the accuracy of the route determination, as well as the computation time. The communication with the IMU can be configured in two ways: 1) considering only azimuthal angles, or 2) using values of roll, yaw and pitch. The first is associated with the rendering of binaural sounds laying on the horizontal plane only, while the second is associated to the reproduction of full 3D sounds. Their implementations differ, as well as, of course, the performance. The sound processing time depends on the size of the sound buffer, which determines the period of the computation. Longer buffers allow for longer computation time, but introduce latency. Finally, the playback time depends on the semantics of the playback system call: it is considered to be equal to the duration of the processed sound chunk for a blocking call, or is considered equal to 0 if the playback is called by a non-blocking function (we neglect the call and return time).

3.9 Results

We analyze the results of the model estimated processing time and simulation time, and determine if the mapped design satisfies the required execution time. Simulations run in a matter of a few seconds. Estimated execution times of a selected set of platforms for the binaural guidance system are reported in Figure 3.4. In the figure, the BeagleBoard is identified by the string “bb”. The planner implementations are labeled from P1 to P4. Planner alternatives are listed in order of increasing accuracy. P1 corresponds to a lower degree of spatial accuracy, but it has the fastest computational time. On the contrary, P4 is slower, but performs the geometric calculation at a higher resolution. Because the planner is computationally intensive, it is always mapped onto the notebook, since the BeagleBoard would not be able to sustain it. Design names starting with 3D refer to the

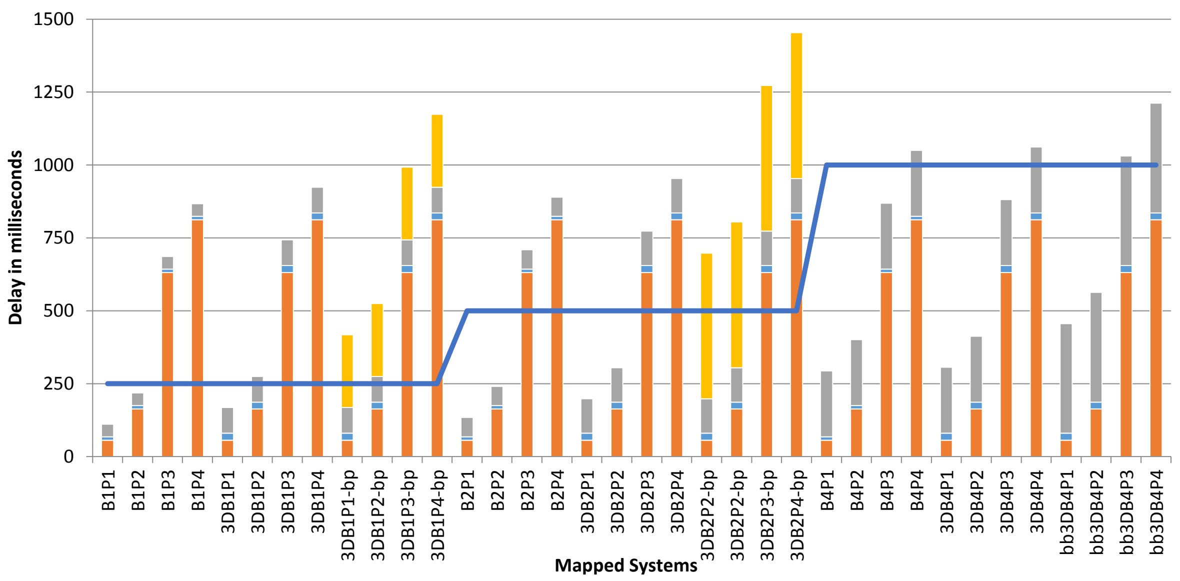


Figure 3.4: The binaural guidance estimated execution time. The blue line represents the maximum execution time required to perform continuous playback. On the bars, orange is the planner execution time, blue is the time required by the IMU, grey is the audio processing time, while yellow is the blocking playback time.

tridimensional sound rendering, as opposed to using the horizontal plane only. The sound buffer size is configured using three different values: 250 ms (labeled as B1), 500 ms (B2) or 1 second long (B4). The "-bp" suffix denotes a blocking playback call.

Figure 3.4 shows the playback time corresponding to the different buffer sizes using the continuous blue line, while the overall execution time of the different solutions is shown as colored bars, showing the planner execution time in orange, the time required by the IMU in blue, the audio processing time in grey, and the blocking playback time in yellow. Designs in which the execution time is lower than the playback time allow the system to reproduce the guidance sound continuously. If the design required execution time is greater than the playback time, the mapped system can reproduce the guidance signal only with pauses in between two consecutive signaling events longer than the delay time. This introduces glitches in the playback, which should be avoided. Among the alternatives, those with an audio processing execution time lower than 50 milliseconds should preferably be taken into considerations for implementation, since the latency affects the human recognition task the least, as discussed previously. Among these, those with a total execution time shorter than the playback time also allows the system to reproduce the guidance signal without interruption. Among the alternatives, for the system testing with participants, we chose the implementation with the shortest buffer time (B1) associated with the Planner configuration P1. Our final choice for experimentation was to adopt the notebook, because the implementations on the BeagleBoard do not respect the 50 ms deadline for the correct interpretation of the guidance sounds in any of the implementations. A more powerful portable platform should therefore be investigated for a wearable implementation.

3.10 Processing of moving sounds

The binaural sound synthesis algorithm has been implemented adopting the techniques described in the previous sections. To improve the user feeling, and to ensure a proper operation of the guidance system even while the user moves the head, we included the possibility of rendering sound movements. However, rendering moving sounds using HRTF poses some problems either they are measured or synthetically generated. The generation of binaural sounds is obtained by means of digital filters. When the virtual sound source moves from one position to another, the filter coefficients must change accordingly. Thus, moving sounds are rendered using time–varying filters. Moreover, traditional 3D sound generation techniques does not include the computation of the time delay of the incoming sounds which would make the sensation of distance more authentic. In our software, the time delay depends on the distance between the sound source and the listener's ears, and may change in time according the movements. As a result, processing fast changes of the time delay, or updating the filters coefficients, will results in clicks, pops, or clipping of the output sound signals.

When synthesizing digital sound, the time is discretized at the sampling frequency. This means that there would be a data of sound amplitude (sample) for each time period, and generally the data is stored in arrays (playback buffer). But in general, when modeling physic of sound waves, time delays may correspond to a non-integer multiple of the sampling period. In this case, the non-integer delay must be rounded to the nearest integer in order to put the sound sample in the corresponding position of the playback buffer. As a consequence of this approximation some annoying phenomena originate. To remove these noises one can introduce a low–pass filter in cascade to the rendering filters to remove high–frequency impurities. However, this introduces phase distortions that could mitigate the feeling of spaciousness. Moreover, if the resulting sound is made of many contributions, the result is prone to quantization errors.

To overcome this issue without using additional filters, it would be possible to use a processing algorithm that uses non–discretized time values, and digitalizes the resulting sequence in order to fill the playback buffer. In this case, time discretization is implemented by sampling the generated sequence each sample–period by interpolating samples that are adjacent in time. In our implementation, every sound signal is represented as a sorted sequence of two floating point values: the sample amplitude, and the sample delay. With a given time period, corresponding to the playback buffer size, the pair sequence is sampled with the playback sampling frequency, corresponding to 44.1 kHz, by linear interpolation of adjacent samples.

Considering the the delays only, the interpolated ITD or the overall time delay computed for the specific spatial location may correspond to a non–integer sample delay, which is not realizable using conventional, integral–tapped delay line, or buffer implemented as arrays. With traditional implementation, the non–integer delay must be rounded to the nearest integer value, and abrupt changes in delay times for closely neighboring spatial location could result from this type of ITD quantization error. The problem is exacerbated if the trajectory of the sound sources change quickly. Moreover, changes in the left and right HRTF filters themselves could produce a disruption in the output.

Our implementation treats each sample as a pair of amplitude and delay values, carrying the advantages of the interpolated delay lines, a well-known structure used in the implementation of some physical models which allows for a non-integer sample delay [63]. It is important to note that interpolated delay lines attenuate the high-frequency components of sounds, and therefore the systems overalls intended spatialization effects may be altered due to the interpolated delay lines spectral coloration. Nonetheless, informal listening seems to suggest that use of interpolated delay lines does not appreciably change the spatialization of most sounds [13]. Thanks to this implementation, we not only have reduced the number of filtering components, removed the quantization noise, and reduced the system complexity, but we also made possible the reproduction of the Doppler effect without overhead.

3.11 Analysis of moving sounds

To demonstrate the advantages introduced by the proposed implementation, we present the comparison between the standard implementation (employing arrays with constant time period) and the algorithm which utilizes a multimap data structure to avoid time discretization. Presented data is processed using ITD and ILD only, in other words it does not use elevation cues. To perform algorithm evaluation we launched the binaural synthesis software and stored on file the sample it generates. We then compared the results off–line using *Matlab*. We compare the sound stream obtained described algorithm (labeled as Map in the plots), and the sound obtained with quantized time delay against a reference signal that has been obtained using the very same algorithm but with a very high sampling frequency.

3.11.1 Case without the Doppler Effect

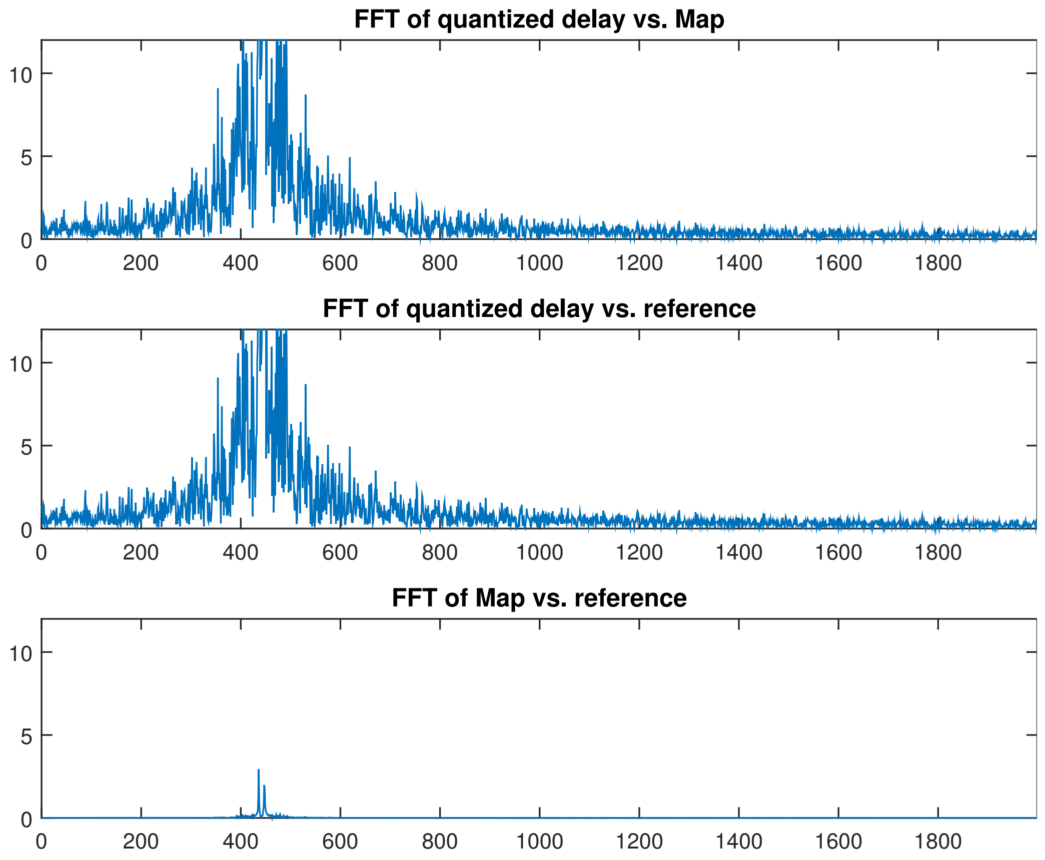


Figure 3.5: Fourier transform of the difference between the standard and the proposed algorithm (top), the standard and the baseline (center), and the proposed vs. the baseline (bottom) for a circular trajectory.

In this study, we considered sounds moving along a circular (fixed radius) trajectory around the listener's head. Since the distance between the center of the head and the sound source is constant during the experiment, there is no Doppler effect. In particular, we compare:

1. sounds obtained using quantized delay, as usually happens when storing digital samples on buffers,
2. sounds obtained using the described approach where digital sound processing is done simulating analog time resolution and discretized at the end of the processing chain,
3. the baseline or reference case, that should represent the real–world case, obtained using the traditional approach, but using a very high sampling frequency.

In particular, for the last case, we use a sampling frequency $F_s' = 1000 \times F_s = 44.1$ MHz. The input monaural signal is an A4 sine–wave which has a frequency of 440 Hz corresponding the the musical note A and is used as a general tuning standard for musical pitch.

In Figure 3.5, from top to bottom, we show the Fourier transform of the signals resulting from the sample–by–sample difference of 1. the standard quantized delay and the proposed algorithm (labeled as Map), 2. the standard approach and the reference obtained at F_s', 3. the proposed algorithm and the reference. The baseline signal has been down–sampled after synthesis with linear interpolation to match the sampling frequency of the other two cases. The difference of the FFT of the signal obtained with the quantized delay and the reference one is large and corrugated. The difference between the signal obtained with the proposed approach and the case with quantized delay is harsh as well. The comparison between the Map and the baseline instead is smooth and has a much lower maximum.

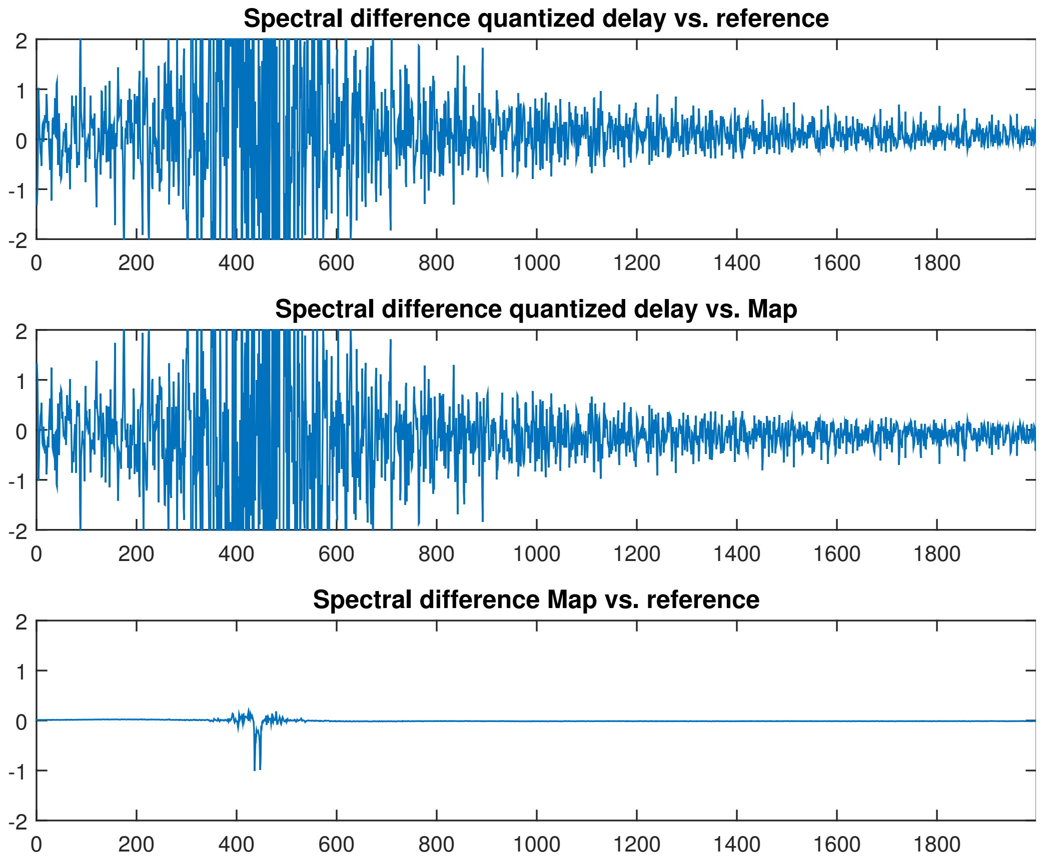


Figure 3.6: Difference between the spectra of the quantized delay algorithm vs. the reference (top). Difference between the spectra the proposed algorithm and the quantized delay algorithm (center). Comparison of the proposed algorithm and the baseline (bottom).

In Figure 3.6 we illustrate the result of the difference of the transforms for the standard vs. the reference (top), the proposed algorithm vs. the standard (center), and the difference of the spectra for the proposed and the reference. The difference between the

Buffer and the Map spectra has a lower peak (≤ 15) and it is slightly rough for higher frequencies. The center and top pictures present almost the same peak value (≥ 800 not shown in picture) centered around the pitch frequency of the input signal. Moreover, it is possible to notice that the utilization of the Buffer introduces noise with an amplitude of about 1 for frequencies distant from the input pitch. On the contrary, the difference between the Map and the reference is very smooth, and noise is always near 0 (specifically 0.25) except for a peak near the pitch frequency smaller with respect to the peak present in the quantized delay case.

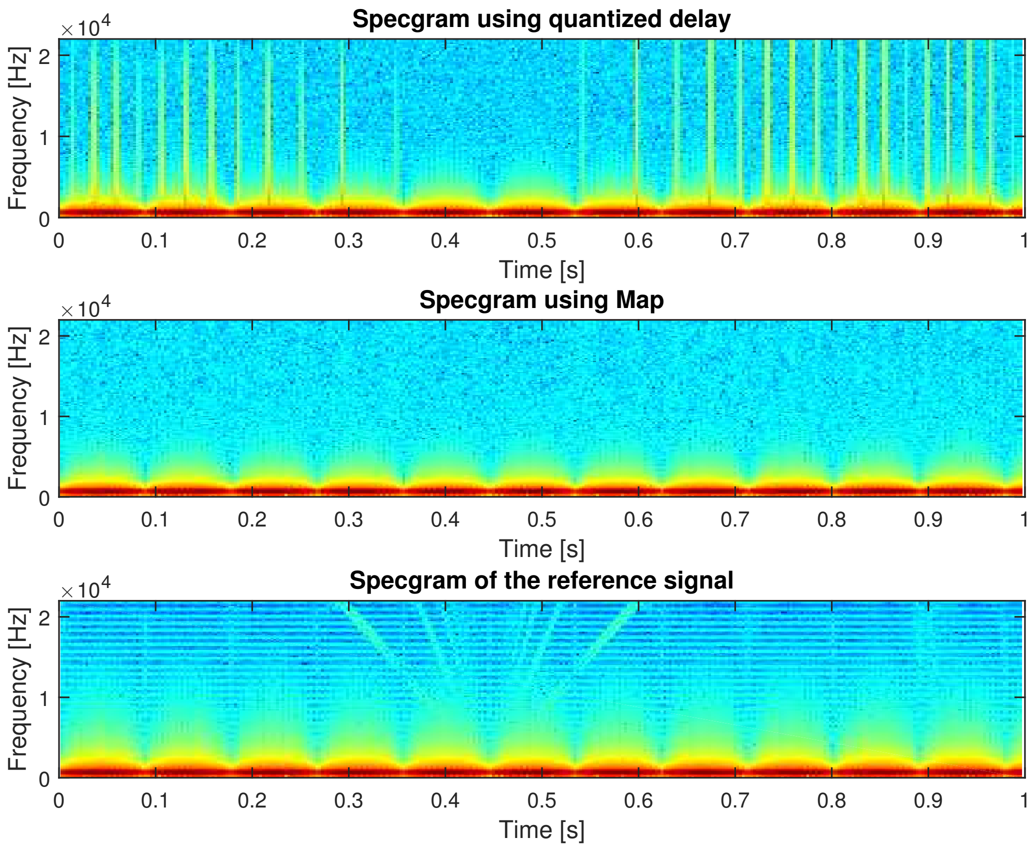


Figure 3.7: Spectrograms of one second of signal for the quantized delay algorithm (top), the proposed algorithm (center), and the baseline (bottom) when the virtual sound source moves along a circular trajectory.

Figure 3.7 shows the spectrogram for the three signals. The standard quantized delay called buffer on top, the algorithm that exploits maps in the middle, and the baseline at the bottom. The Buffer contains high frequency components that results in a "robotic" sound output and some clicks corresponding to the vertical lines in the plot. The Map approach does not suffer from the phenomena and sounds good. The baseline high–frequency signal has got some higher frequency components that has a behavior similar to what we have found with the buffer but with much lower intensity and are due to aliasing. Comparing the sound obtained with map and the one obtained under-sampling the analytical it is hard to perceive any difference. The sound obtained with the buffer is noisy, sounds robotized, and is unpleasant to hear.

3.11.2 Case with Doppler Effect

The analysis described in section 3.11.2 has been conducted for a virtual sound running across a linear trajectory. In this case, the sound source moves along a linear path centered 1 meter in front of the listener. The path extends for 2 meters on the left and right sides.

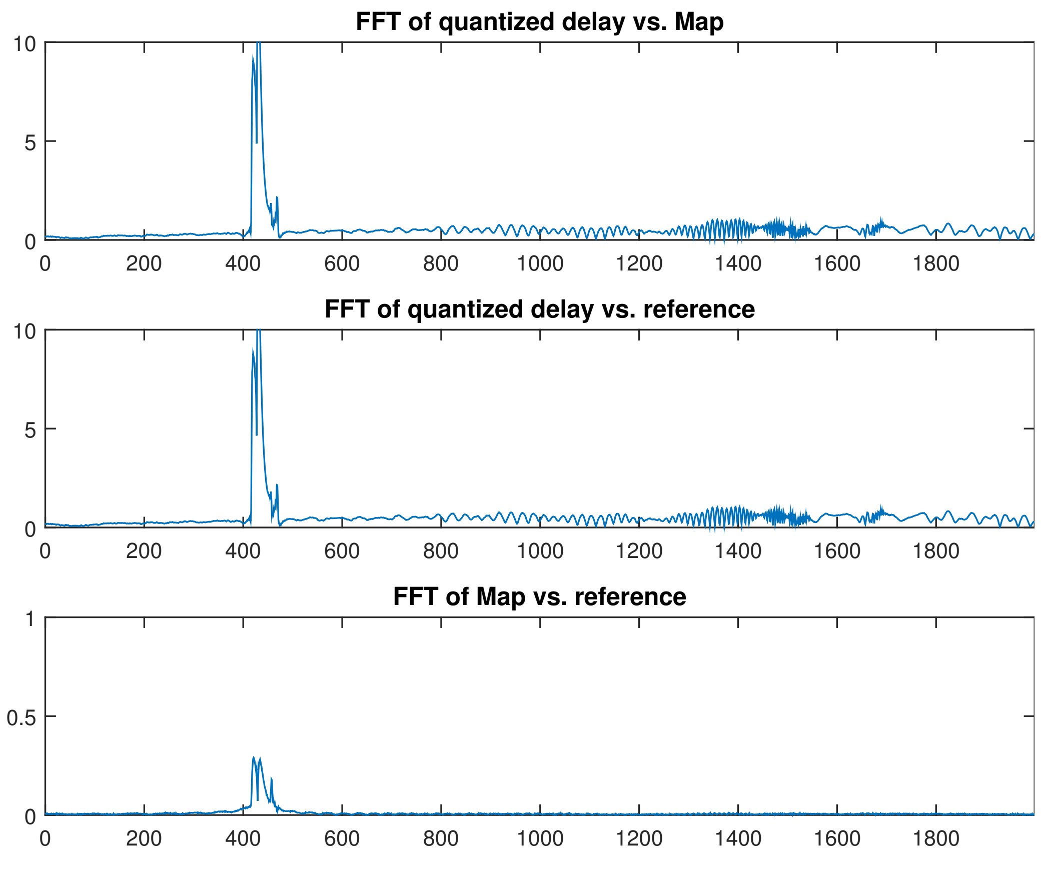


Figure 3.8: Fourier transform of the difference between the standard (quantized delay) and the proposed algorithm using Maps (top), the standard and the reference (center), and the proposed vs. the reference (bottom) for a linear trajectory. Notice the change of scale of the vertical axis for the image at the bottom.

In Figure 3.8, from top to bottom, we show the Fourier transform of the signals resulting from the sample–by–sample difference of 1. the standard quantized delay and the proposed algorithm (labeled ad Map), 2. the standard approach and the baseline obtained at F_s', 3. the proposed algorithm and the baseline. The reference signal has been down–sampled after synthesis with linear interpolation to match the sampling frequency of the other two signals. The difference of the FFTs of signal obtained with the buffer and the baseline one is very large for the pitch frequency and corrugated for other frequencies (middle), the same happens for the quantized case compared to the map (top). The signal is especially noisy compared to the case in which there is no the Doppler Effect. The difference among the FFT of the signal obtained with Map and the FFT of the baseline is almost equal to zero (around 0.3 compared to about 12 for the other two cases).

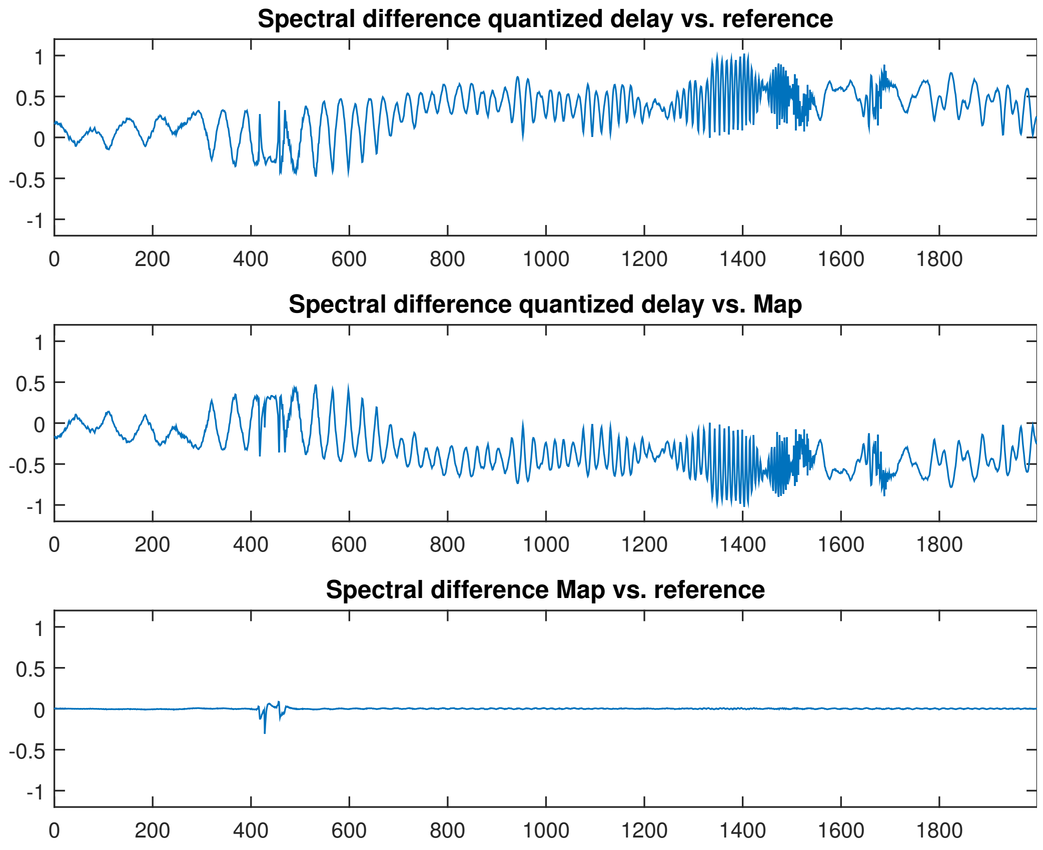


Figure 3.9: Difference between the spectra of the quantized delay algorithm vs. the baseline (top). Difference of the FFTs of the proposed algorithm and the standard case (center), and the spectral difference of the proposed non–quantized algorithm and the reference (bottom).

Fig. 3.9 shows the differences between the spectra of the signals obtained for the different cases. In this case, the signal obtained using the standard quantized approach is again noisy if compared to the reference or to the signal obtained using the `map` data structure. All the spectra of the signals have a a peak with a larger base with respect to the previous case, because the Doppler effect induce a pitch shift of the output signal (picture not shown). The difference between the signal obtained with the proposed approach and the reference case is almost negligible. In fact, it presents just a little spike in correspondence of the pitch frequency, but it is inaudible.

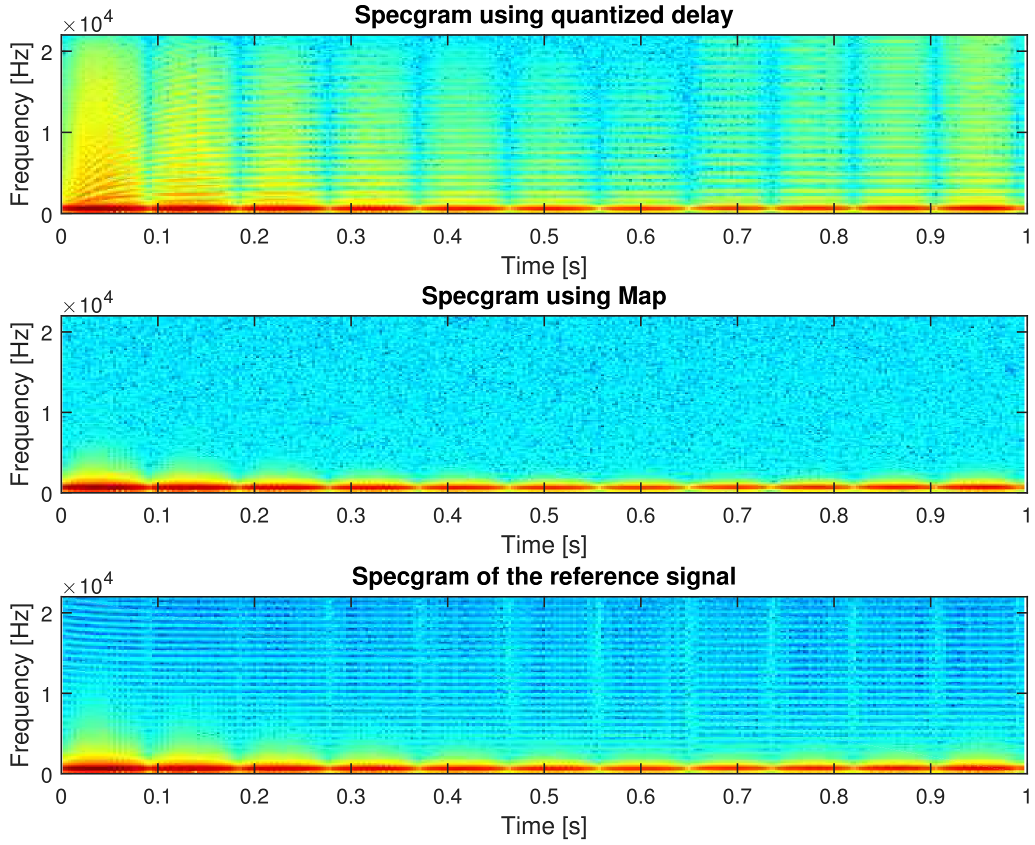


Figure 3.10: Spectrogram of one second of signal for the quantized delay algorithm (top), the proposed algorithm (center), and the baseline (bottom) when the virtual sound source moves along a linear trajectory.

Finally, Figure 3.10 shows the spectrogram for the signals obtained with the quantized buffer (top), the non-quantized buffer (middle), and the high–frequency reference signal. The signal in the buffer presents high frequency components, that are not present in case of the processing using non–discretized delay time. The baseline spectrum is very similar to the spectrum obtained by processing the sound signal with the proposed method, except for very narrow disturbances at higher frequency that originate from the down–sampling (aliasing) Listening to the sound obtained with map and the one obtained under-sampling the analytical it is hard to perceive any difference. Again, the sound obtained with the buffer is noisy, sounds robotized, and is unpleasant to hear especially for the pitch shift that originates from the Doppler effect.

Table 3.2: Signal to Noise Ratio of the standard and proposed implementations compared to the ideal one. SNR of the proposed implementation against the standard one.

Circular Trajectory (no Doppler Effect)			
	S = buffer, N = buffer - synth	S = map, N = map - synth	S = map, N = map - buffer
SNR	28.568 dB	60.358 dB	28.565 dB
Linear Trajectory			
	S = buffer, N = buffer - synth	S = map, N = map - synth	S = map, N = map - buffer
SNR	29.015 dB	59.305 dB	29.013 dB

3.11.3 Summary

To conclude the discussion, we have evaluated the Signal to Noise Ratio (SNR) for the different cases illustrated in the previous two sections. Results are summarized in Table 3.2. The SNR is expressed in decibel (dB) and computed as the logarithm of the ratio between the power of the signal and the power of the noise:

$$SNR = 10 \log_{10} \frac{P_{signal}}{P_{noise}}$$

In particular, we considered as the signal the sound signal obtained with one of the algorithms, and as the noise the result of the sample–by–sample difference between said signal and the reference signal since the reference signal is considered to be noiseless. The result of the subtraction of the *ideal* signal from the noisy one should be just noise. The left column of the table reports the SNR of the signal obtained with standard quantized time delay algorithm versus the baseline. In this case we get a SNR near 30 dB for both paths. The middle column compare the proposed approach to the baseline, and the SNR almost doubled and reaches 60 dB. For the sake of completeness, we compared also the standard and the proposed implementations (right column). Even in this case, the approach using buffer compared to the proposed algorithm exhibits a SNR almost equal to the value obtained when comparing the buffer against the baseline, 30 dB.

3.12 Experimental Evaluation

Besides the technical study of the binaural guidance system illustrated in the previous sections, we also conducted an experimental study to determine the guidance capability of all the interfaces implemented on the *c-Walker*. To this end, we conducted a series of tests illustrated in this section. Besides the binaural guidance mechanism, we tested the haptic interface, the mechanical guidance, and an acoustic guidance based on stereo sounds rather than 3D sound. Mechanical guidance is the only active mechanism, while all the others are passive. The main focus of the evaluation was on system performance, rather than on the user experience. The study objective is to develop a controlled experimental methodology to support system comparisons that can be applied in the future to potential *c-Walker* users. In this experimental study, we involved a sample of young participants.

Participants

Thirteen participants (6 females, mean age 30 years old, ranging from 26 to 39) took part in the evaluation. They were all students or employees of the University of Trento and gave informed consent prior to inclusion in the study.

Design

The study applied a within-subjects design with Guidance (4) and Path (3) as experimental factors. All participants used the four guidance systems (acoustic, haptic, mechanical, binaural) in three different paths. The order of the system conditions was counterbalanced across participants.

Apparatus

An exhaustive description of the device and of its different functionalities can be found in [50]. In the specific case of this experimental evaluation, the *c-Walker* is configured as in Figure 3.3 when it provides binaural guidance. When the system is configured to perform acoustic guidance, it does not make use of the IMU to control the user head orientation since stereo guidance only gives information on direction of the turn and does not convey information regarding the amount of the suggested steering amount. The haptic guidance system utilizes vibrating bracelets to communicate to the user the direction to take. The mechanical guidance instead takes control of the *c-Walker* to accompany the user within the correct direction.

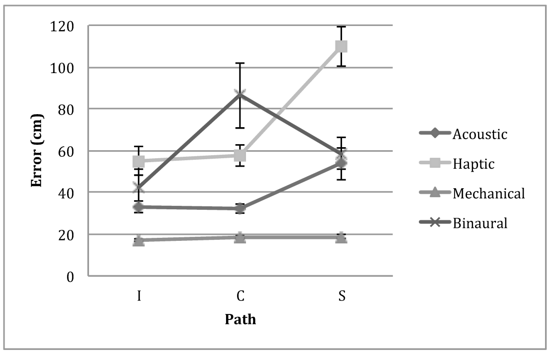
The short term planner of the *c-Walker* collects collects real–time information of the environment and uses it to plan safe courses that avoid collisions with other people or dangerous areas [14]. In the context of this experimental evaluation, the planner is disabled since the experiments took place in free space, without any dynamic obstacles along the way. We considered 10 meters long paths with a virtual corridor 60 centimeters wide. The virtual corridor defines a region of tolerance within which the *c-Walker* is considered to be inside the correct trajectory. Within the virtual corridor the guidance interfaces do not intervene. We implemented three kinds of path: straight (I), C–shaped (C), and S–shaped (S). The I path is a 10 meters line. The C path is a quarter of the circumference of a circle with a radius of 6.37 meters. The S path is composed of three arches of a circumference with a radius of 4.78 meters. The first and the third arches are $1/12$, while the second is $1/6$ of a circumference. The second arch is bent in the opposite side with respect to the others. There are six paths in total, including the mirrored version.

Procedure

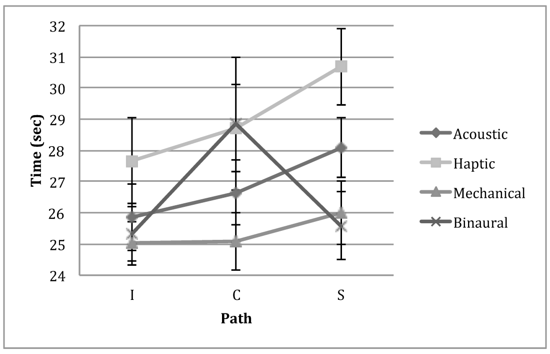
The evaluation was run in a large empty room of the University building by two experimenters: a psychologist who interacted with the participants and a computer scientist who controlled the equipment. At the beginning of the study, participants were provided with the instructions in relation to each guidance system. It was explained that they had to follow the instruction of the *c-Walker*. While the participant is on the correct trajectory there would be no system intervention. Otherwise, each system would have acted in different ways. The mechanical system would have turned the front wheels modifying its direction onto the correct path. In this case, participants could not force the walker and might only follow the suggested trajectory. When the *c-Walker*is along the correct path, the participants were given back the control of the walker. For the haptic guidance, a vibration would have indicated the side of the correction necessary to regain the right trajectory. The stereo guidance does not take advantages of sound localization. The stimuli to convey to the user the instruction to move in the left direction is made by a audio signal only on the left speaker. Using haptic or stereo guidance, the interfaces do not convey information indicating the turn intensity. The amount of correction is expressed by the frequency of repetition of the signals. The stereo guidance does not make use of the inertial monitoring unit. The binaural sound guidance is made using the described spatialization algorithm and the user head monitoring. The binaural guidance provides a sound indicating both the direction and the amount of the correction.

Before each trial, the appropriate device was put on the participant (i.e., headphones or haptic bracelets). Only in the case of the binaural system, participants were given a brief training to make them experience the spatial information of the sounds. The starting position of each trial varied among the four corners of a rectangular virtual area (about 12×4 meters). The *c-Walker*was positioned by the experimenter with a variable orientation according to the shape of the path to be followed. Specifically, at the beginning of each I trial, the walker was turned $10°$ either to the left or to the right of the expected trajectory. At the beginning of each C and S trials, the walker was located in the right direction to be followed. Participant started walking after a signal of the experimenter and repeated 10 randomised paths for each guidance system for a total of 30 trials.

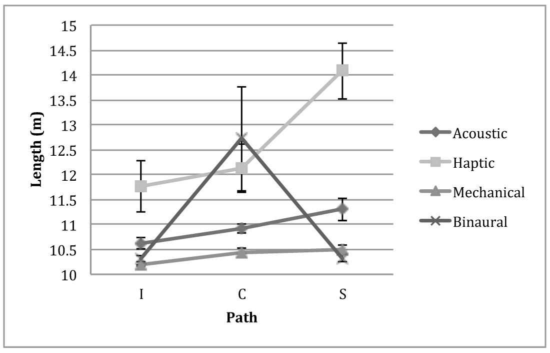
At the end of each system evaluation, participants were invited to answer 4 questions, addressing ease of use, self–confidence in route keeping, acceptability of the interface



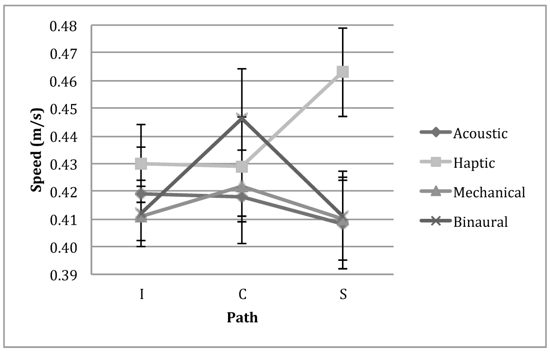
(a)



(b)



(c)



(d)

Figure 3.11: Different metrics as function of Guidance and Path: (a) average error (cm), (b) average time (s), (c) average length (m), (d) average speed (m/s).

in public spaces and an overall evaluation on a 10 points scale (10 meaning positive). Moreover, participants were invited to provide comments or suggestions. The evaluation lasted around 90 minutes per participant.

Data analysis

Performance was analysed considering four dependent variables. A measure of *error* was operationalized as deviation from the optimal trajectory and calculated using the distance of the orthogonal projection between the actual and the optimal trajectory. We collected a sample of 100 measurement (about one value every 10 centimetres along the curvilinear abscissa of the path) that were then averaged. *Time* was measured between the start of participant's movement and the moment the participant reached the intended end of the path. *Length* measured the distance walked by the participant, whereas *speed* corresponded to the ratio between the length and the time. For each participant and guidance system, we averaged an index scores for the four S, the four C and the two I paths.

Guidance	**I**	**C**	**S**	**Average**
Acoustic	33.0	32.2	53.7	**39.6**
Haptic	55.0	57.4	109.7	**74.0**
Mechanical	17.1	18.3	18.8	**18.1**
Binaural	42.1	86.3	58.7	**62.4**
Average	**36.8**	**48.6**	**60.2**	

Table 3.3: Average error (cm) for each experimental condition.

Guidance	**I**	**C**	**S**	**Average**
Acoustic	25.8	26.7	28.1	**26.9**
Haptic	27.7	28.7	30.7	**29.0**
Mechanical	25.0	25.1	26.0	**25.4**
Binaural	25.3	28.9	25.6	**26.6**
Average	**26.0**	**27.3**	**27.6**	

Table 3.4: Average time (in seconds) for each experimental condition.

Error

Descriptive statistics of error are reported in Fig. 3.11 (a) as a function of Guidance and Path. The mechanical guidance differed significantly from all the others being the most precise interface. Moreover, the acoustic guidance was significantly different from the haptic. Results indicated that the straight path (I) was significantly easier from the other two. In the mechanical guidance condition, the error was not affected by the path and showed very low variability among participants. On the contrary, for all other conditions there was an effect of Path on the magnitude of the error. Mostly for the haptic, but also for the acoustic guidance, the S path had the highest error. Interestingly, for the binaural guidance, the highest error emerged with the C path. Fig. 3.12 shows some qualitative results of the experiments.

Time

Figure 3.11 (b) shows the average time in relation to both guidance and path. The mechanical guidance allows the user to go to through the path significantly faster than the haptic. Walking along the I path is faster than the S path. Walking time was independent of Path for the mechanical and the binaural guidance. Conversely, the S path was performed significantly slower than the I path for both the acoustic and the haptic guidance. The average time are summarized in Tab. 3.4.

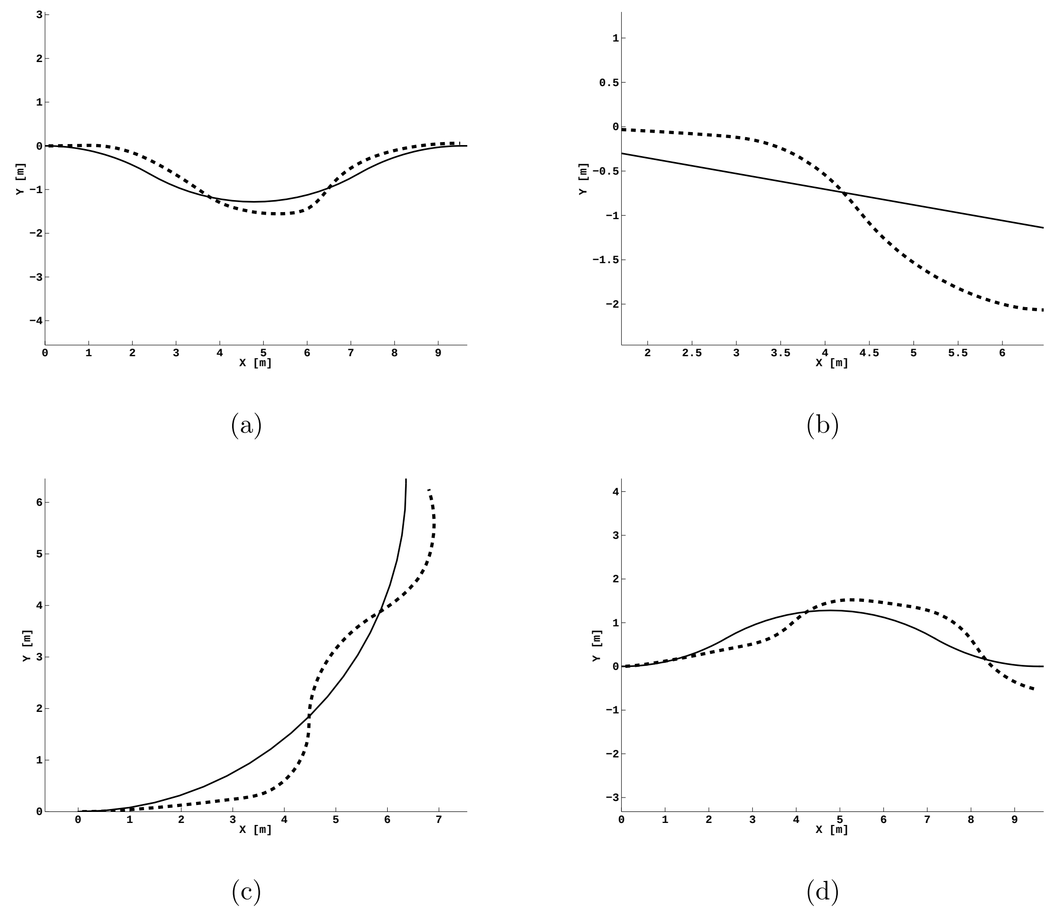


Figure 3.12: Some examples of qualitative results for different guidance algorithms: (a) mechanical, (b) haptic, (c) audio, (d) binaural. Dash line represents the user trajectory, the continuous line represents the suggested path.

Length

Only two participants (both of them in the S path and the binaural guidance condition) walked less than the optimal path length. The haptic guidance differed significantly from all the others. Moreover, the mechanical guidance differed significantly from the acoustic. The I path differed significantly from the C and S paths (Tab. 3.5). The haptic guidance showed the worst result in the S path. For the mechanical condition, the performance was different between the I and S path. For the binaural condition there was no effect of path. Fig. 3.11 (c) shows the average length in relation to both Guidance and Path.

Guidance	I	C	S	Average
Acoustic	10.62	10.92	11.3	**10.9**
Haptic	11.76	12.12	14.08	**12.7**
Mechanical	10.18	10.44	10.5	**10.4**
Binaural	10.31	12.72	10.32	**11.1**
Average	**10.7**	**11.6**	**11.6**	

Table 3.5: Average travelled length (meters) for each experimental condition.

Guidance	I	C	S	Average
Acoustic	0.42	0.42	0.41	**0.42**
Haptic	0.43	0.43	0.46	**0.44**
Mechanical	0.41	0.42	0.41	**0.41**
Binaural	0.41	0.45	0.41	**0.42**
Average	**0.42**	**0.43**	**0.42**	

Table 3.6: Average speed (m/sec) for each experimental condition.

Speed

Fig. 3.11 (d) reports the average time as a function of experimental conditions. The average speed and the results are summarized in Tab. 3.6. Participants were particularly fast walking the S path.

Questionnaire

After testing the various guidance interfaces, participants were given a questionnaire to understand their level of satisfaction. Results indicates that the mechanical guidance is perceived as easier to use. Moreover, participants using the mechanical interface feel confident to maintain the correct trajectory. Concerning the acceptability to use the guidance systems in public spaces, the mechanical guidance was the preferred one. Participants spontaneously commented that the mechanical system was easy to follow and required little attention. However, some of them complained that it might be perceived as coercive and risky due to possible errors in route planning or in actuation.

Participants reported a general dislike about wearing headphones mostly because they might miss important environmental sounds and because of the look. Most of the participants agreed that the binaural condition required more attention than all the other systems. However, participants appreciated that it was something new, interesting and provided a constant feedback on the position. Most of them preferred the binaural system to the acoustic one because it provided more information, yet some reported a difficulty in discriminating the spatial location of stimuli.

Most of the participants reported to prefer the haptic guidance system to the acoustic, as easier and less intrusive. In relation to the both guidance condition, participants complained about the poverty of the left and right instructions and the lack of a modulation. Some participants suggested possible ways to increase communication richness, such as, for the acoustic system, different volume indicating the magnitude; verbal feedback; different tones in relation to the angle. For the haptic system comments included modulating the frequency of the vibration in relation to the magnitude of the correction. Some participants reported a kind of annoyance for the haptic stimulation but only for the first minutes of use.

3.12.1 Discussion

The aim of this study was to gather quantitative and qualitative information in relation to the evaluation of four different guidance systems. To this aim participants had the opportunity to navigate non-visible paths (i.e., virtual corridors) using four the different guidance systems. To maintain the correct trajectory participants could only rely on the instructions provided by the *c-Walker* and, after using each system, they were asked to provide feedback.

As expected, in terms of performance, the mechanical guidance was the most precise. The results show the consistency of the deviation along the different paths, a low variability among the participants and a slight difference in relation to the shape of the paths. The results of the questionnaire further support quantitative data showing that, on average, participants liked the mechanical guidance the most in relation to easiness, confidence in maintaining the trajectory, acceptability and overall judgment. The only concern for some users was that it might be perceived as coercive and risky due to possible errors in route planning. In the acoustic guidance, there were only left and right sounds while in the binaural guidance, the sound was modulated by modifying the binaural difference between the two ears. Although more informative, in terms of quantifying the angle of the suggested trajectory, the binaural guidance system emerged to be worse than the acoustic system in the C path. However, it is likely that with adequate training, the performance with the binaural system could improve a lot. The results of the questionnaire suggest that both the systems using headphones were not very acceptable because of the possibility to miss environmental sounds and because of the look. Moreover, the binaural system was reported to require more attention than the acoustic one, although no difference emerged in terms of confidence in maintaining the correct trajectory. Overall, the binaural guidance was appreciated because it was "something new" and it provided a more detailed information. Indeed, most of the participants' suggestions related to the acoustic and haptic guidance systems were addressed at codifying the instructions in terms of the angle of the correction.

Significant performance differences emerged between the haptic and the acoustic guidance, which could in part be explained by the natural tendency to respond faster to auditory stimuli rather than to tactile stimuli, and in part by the different algorithm employed in the evaluation. It is evident that, independent of the communication channel, the dichotomous nature of the stimulation (left/right) tended to stimulate long left and right corrections leading to zigzagging. In terms of user experience, the haptic guidance was perceived as more acceptable than the acoustic and the binaural systems, and no different from the mechanical one.

Indeed, most of the participants commented that the haptic bracelets could be hidden and did not interfere with the environmental acoustic information. On the contrary, an issue emerged with regards to the acceptability of the practical requirement of wearing headphones. The binaural system was perceived as a promising solution which captured the user attention. But it has been questioned the inability to hear surrounding sounds (or to hear them muffled). To overcome this problem, we designed an analog circuit board capable of mixing environmental sounds (captured by a couple of microphones mounted on the headset itself) with the guidance signal coming from the computing board. The circuit board size make it suitable to be fitted on the headphone arches on top of the already present IMU board.

Chapter 4

Thermoelectric Energy Recovery System

In this chapter, we illustrate how the proposed design methodology is applied to a very different design context. Unlike the case previously illustrated, in this activity we model two heterogeneous systems that are functionally separated. On one hand, a high-performance system representing a CPU of a data center, on the other hand a low–power device that provides environmental monitoring or active CPU cooling to the high performance device. The low–power system is powered by a circuit that harvests the energy from the heat generated by the larger system. Even though they are functionally separated, the two systems have a physical relationship that makes the operation of the low-power system subject to the activity carried out by the high performance device. The methodology allows the model to capture this relationship in the design environment concurrently with the function architectural codesign. After having designed and characterized the energy recovery system and having implemented the low–power device, the design tool has allowed us to explore several design solution to identify the optimal one.

4.1 Context of the Project

A common problem for all the computing systems is related to the thermal management. In fact, electronic circuits dissipate heat during their operation according to the clock frequency, the voltage, and the micro–fabrication technology. Active or passive heat sinks, and fans are used to prevent the occurrence of faults, but if mechanical systems are not sufficient to decrease the temperature, the electronic circuit itself can scale down its clock frequency, or its supply voltage to decrease the temperature. However, this is not always convenient since a lower clock frequency often means increasing the task execution time. The problem scales with the system's size, so it is even more evident in data centers.

To improve data centers management and to reduce the risk of faults, it is of fundamental importance to gain a greater insight of energy utilization inside the facility. To this end detailed measurement of environmental quantities is required. We address this problem by designing a system that provides monitoring using the dissipated heat as a source of energy, and that can be also useful to alleviate the issue of heating. Our goal is to design and to realize a system that satisfies the following requirements:

1. it is intended to be used in the data center of the future, made of low–power ARM core processors,
2. the system acts as a passive and/or active CPU cooler,
3. it works as a wireless sensor node, i.e. it monitors environmental quantities of interest, and sends them over the wireless channel to a data collection unit that takes care of managing the data center and the monitoring system,
4. it collects data with a sampling frequency equal or higher with respect to devices already present in the market,
5. it works with free energy.

We focus our attention on ARM CPUs, because nowadays they are considered the most promising technology for the realization of the future server farms [56], as those used by large IT companies for Web 2.0 and cloud computing services. Recent ARM CPUs perform a significant number of operations, and can execute complex applications. ARM–based devices offer lower energy requirements, reduced costs compared to multicore architectures, higher thermal dissipation and consequently lower management costs. Besides this, the low computational complexity required by web services routines makes ARM CPUs highly interesting for this novel application [72].

A monitoring system for data centers is used to measure several environmental parameters whose value, when out of the ordinary, could denote that a hazard may be incurred. Different types of sensors, such as light or proximity detectors, can reveal the presence of unauthorized personnel. The presence of coolant leaks, excessive moisture, or unwanted liquids, can be revealed through moisture sensors. Moreover, by controlling the temperature value it is possible to infer whether there is an overheating problem, due for example to a fault in the cooling system. Monitoring a data center not only means to detect faults, but also to have available information useful to improve the utilization of the data center. For example, using collected data and advanced prediction techniques combined with the ability of dynamically allocate the tasks, it is possible to optimize the scheduling of processes to improve the energy utilization.

Our goal is to design and to realize a system based on wireless sensor network to monitor the data center infrastructure. The system does not requires power wires or batteries to operate. Since cloud farms are plenty of wasted heat, we considered to supply our system with Thermoelectric Generators (TEGs). A TEG is a device that converts a local temperature gradient (usually due to a heat flow from thermal source) into electrical energy, by exploiting the Seebeck effect [68]. A TEG is made by a junction of two dissimilar metal bars. When bars face different temperatures, electrons move between the hot side and the cool side. The electromotive force is proportional to the thermal difference that exists between the two metal layers. The efficiency of the conversion also depends on the composition of the two conductive bars. The device can monitor the environmental parameters with a shorter sampling period compared to commercially available devices. Moreover, thanks to its reduced size, a large number of devices can be placed inside a server room, therefore it is possible to perform distributed monitoring of the environment, and to provide a wider cumulative data rate.

Data Center Genome Project, a monitoring system developed by Microsoft Research, uses Wireless Sensor Network to sample the environment every 30 seconds [41]. Other monitoring system already available in the market usually are powered from the grid, burdening the data center Power Usage Effectiveness (PUE). The sampling period of solution proposed by Dell, called OpenManage Power Center Temperature Monitoring, can be set to 1, 3, or 6 minutes [17].

The system we developed works also as an energy–neutral hybrid cooler. While operating as a wireless monitoring node, the system works as a passive heat–sink allowing data center CPU to work within the safe temperature range. When requested, the system stops performing monitoring to harvest more electrical energy that can be used to supply a fan to provide CPU active cooling. When active cooling is possible (meaning, when the system has scavenged enough energy to power a cooling fan), the host CPU runs at overclock speed thus it can perform more computation (or can finish the job early). While the CPU is overclocked, if the CPU temperature reaches a warning threshold, the system activate the fan to provide active cooling. In case the scavenged energy is no more sufficient to power the fan, when the CPU temperature reaches again a warning value, the CPU returns to a lower clock frequency to prevent overheat.

4.1.1 Author Contribution

Within the context of the project, several papers have been published. An initial study devoted to the characterization of commercial thermo electric generators ca be found in [60]. This characterization has been performed to find the relation between the task running on the host processor and the generated power. Harvested power has been used to supply a first version of the wireless sensor node based on the *Wispes WT24H* wireless node. Subsequently, in [64] we described the development of the system with a deepen analysis of the features of the thermo electric generator accompanied by the circuit for conditioning and storage of the harnessed energy. We also shown the mathematical model of the input–output relation of the device and the simulation of the complete system. Subsequently, we demonstrate how the harvested energy can be used to build a energy–neutral wireless sensor node for monitoring of data centers and the utilization of beacon packet frequency to infer the temperature of the host CPU [62]. Moreover, we illustrate the utilization of the system as an active and as a passive CPU cooler to sustain overclocking [61]. A complete description of the energy harvesting module and the wireless sensor node based on the MSP430 architecture can be found in [11] together with the mathematical model of the input–output relation of the harvester and the model of the CPU temperature for the host system. The personal contribution of the author of this thesis is related to the design of the experimental setup for gathering data, the development of the simulator for system evaluation, and the analysis of data to extract models.

4.1.2 Chapter Structure

The rest of this chapter is organized as follows. First we describe the components of the system in Section 4.2. In addition to describing the functionality of the device under consideration, we also show how the various components of the system have been characterized to extract models useful to perform the overall system analysis. The information obtained from the characterization phase were subsequently used to describe components within a model in METROII. Section 4.3 illustrates how the TERS system has been model adopting the methodology described in this thesis. The tool allows us to analyze the performance of the system and to compare how different solutions behave, in particular it is possible to evaluate the effect on the overall system is subject to the variation of certain parameters of components that constitute the system.

4.2 Prototype Development

4.2.1 Target host platforms

Market forecasts pointed out that the current trend in data centers is to migrate from traditional architecture toward low–power devices similar to those used for embedded system solutions. To this end, in our experiment we use as host devices two embedded system boards based on ARM processors:

- a Pandaboard with a 1 GHz ARM Cortex A9 CPU and 1 GB of low power DDR2

RAM [51];

- a Samsung Arndale equipped with an Exynos 1.7 GHz dual-core ARM Cortex A15 processor with 2 GB DDR3 RAM 800 MHz [2].

Both systems run Linux kernel version 3.10. Nowadays, ARM–based devices are considered a promising solution for building computing server for cloud service providers and Web 2.0 applications. For example, OBS uses servers constituted by clusters of Arndale-Board. Codethink and Calxeda build ARM–based SoC for cloud services based on A9 or A15 architectures, with up to 2432 cores.

4.2.2 Characterizing harvesting module

To estimate the order of magnitude of the power generated by thermoelectric generators in the context of computing systems, we have compared four different devices provided by different manufacturers. These devices differ in terms of aspect ratio, number of cells and cells arrangement. The devices we chose are:

- *Nextreme* eTEG HV56 Thermoelectric Power Generator with 32×32 mm squared footprint, and a 31×33 mm power generator without the output power regulator provided by the vendor [45];
- *Micropelt* TE-CORE7 TGP-751 ThermoHarvesting Power Module with a 43×33 mm heat sink, and a circular footprint with $\phi = 10$ mm [40].
- *Peltier-Cell* PE1-12706AC 40×40 mm squared cells.

The first two devices of this list are specifically designed as thermoelectric generators, in fact they are made by microfabricated thermocouples optimized for the Seeback effect. The third consists of a generic Peltier cell generally used for cooling applications that we used in two different configuration: as a single cell (one–stage), and two cells connected in series (two–stage).

We built the two–stage structure to optimize the thermal exchange between the ARM CPUs and the Peltier cells. The two–stage configuration is shown in Figure 4.1. The single stage the Peltier/thin spreader/Peltier is substituted by a single element. The lower spreader (thickness 2 mm) allows the heat to distribute on the whole surface, while the one in between (thickness 0.1 mm) speeds up the exchange between the two Peltier cells, and spreads the heat once again. We put thermal grease in between each layer of different materials to guarantee maximum thermal contact. The spreaders are made of aluminum because of its thermal efficiency, low cost and market availability. The last layer of the stack is a commercial heat sink that optimizes the heat dissipation with the environment.

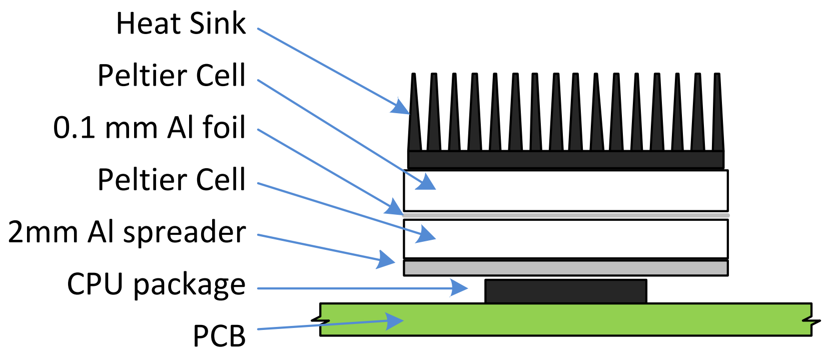


Figure 4.1: Thermoharvester schematic made of 2 Peltier cells, spreaders and sink in stacked fashion. We put thermal grease in between each layer of different materials to guarantee maximum thermal contact.

The characterization of harvesters performance are obtained using open circuit voltage and impedance matched power. We conducted the characterization of the TEGs listed above by executing different tasks in sequence in a benchmark fashion. In particular the tasks were selected to achieve a range of time length and CPU loads (with fixed clock frequency set as the maximum of each ARM CPU board):

- **video encoding** using `ffmpeg` with 4 threads, which converts a 2 hour–long movie;
- **multithread application** that performs millions of algebraic and trigonometric computations of floating point numbers using a user defined number of threads (called `busywork` in pictures);
- **kernel operations** in this task a Linux kernel is uncompressed, compiled and then cleaned, then the folder is removed.

Output voltage and current have been measured over a matched load with a 1 s sample period.

The benchmark applications allowed us to obtain an output power profile for several configurations (ARM CPUs plus TEGs). Detailed results of the experiments are reported in Figure 4.2 to Figure 4.9. These pictures present the output impedance matched power of each TEG under the ΔT produced by the heat dissipated by the CPUs, computed as the product of the measured voltage and current. ΔT in turn is strictly related with the CPU load. The air conditioning system of the lab kept the room temperature almost constant in the range 22° C to 25° C. Generally speaking, the longer the task the higher the CPU temperature as well as the number of memory accesses and switches (algebraic computations). Pandaboard's CPU builds up lower temperature with respect to the Arndale's (37° C vs. 80° C), resulting in a 10× difference in the TEG measured output power in our experiments (μW range for Pandaboard while mW for Arndale's). All but Nextreme exhibit a good reactivity to fast temperature spikes, this one produces smooth

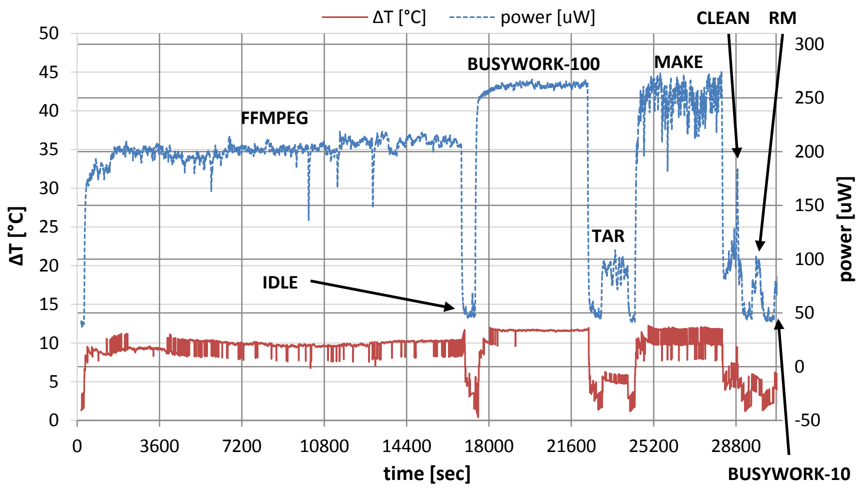


Figure 4.2: Output power of the Nextreme TEG placed on top of the Pandaboard while running the benchmark application. The continuous line represents the thermal gradient as difference between the hot side and the heat-sink.

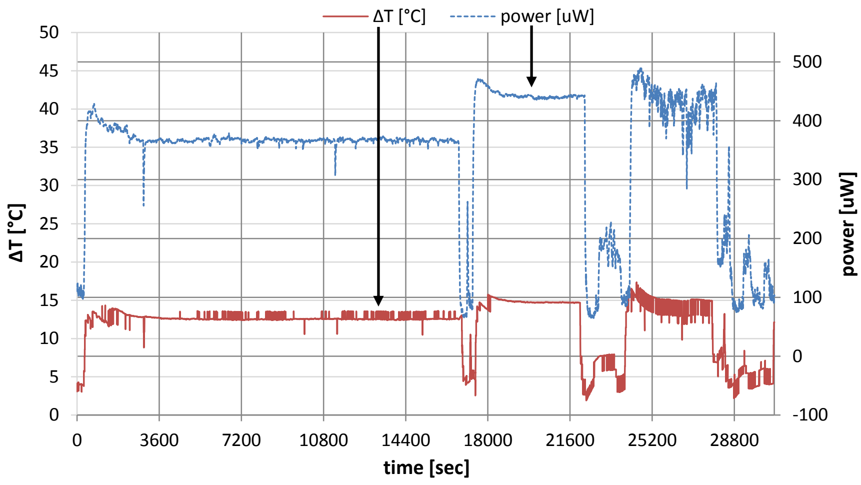


Figure 4.3: Output power of the Micropelt on Pandaboard.

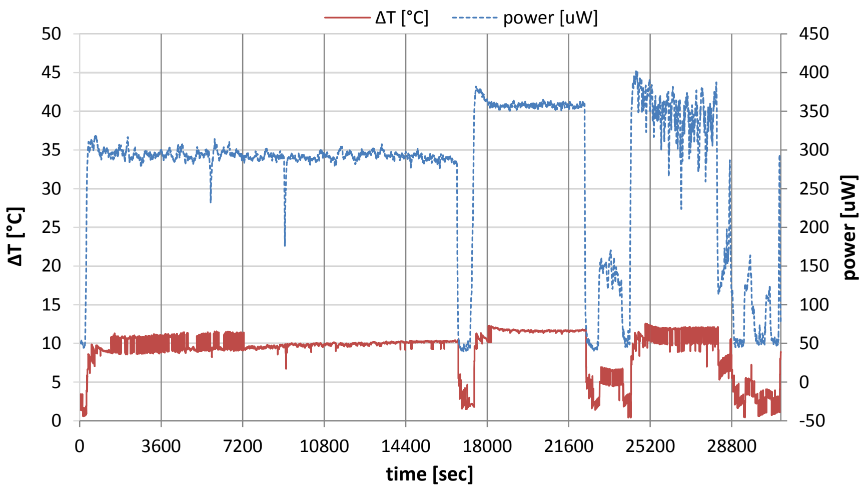


Figure 4.4: Output power of a single Peltier cell on Pandaboard.

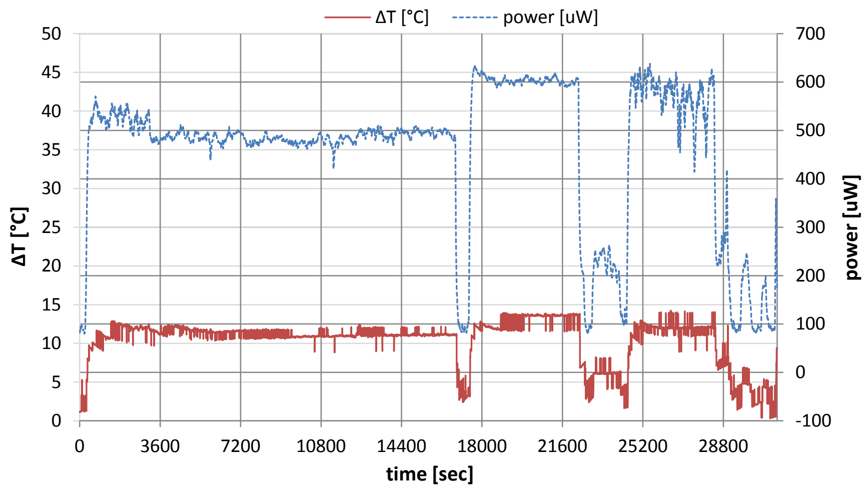


Figure 4.5: Output power of the two–stage Peltier cells on Pandaboard.

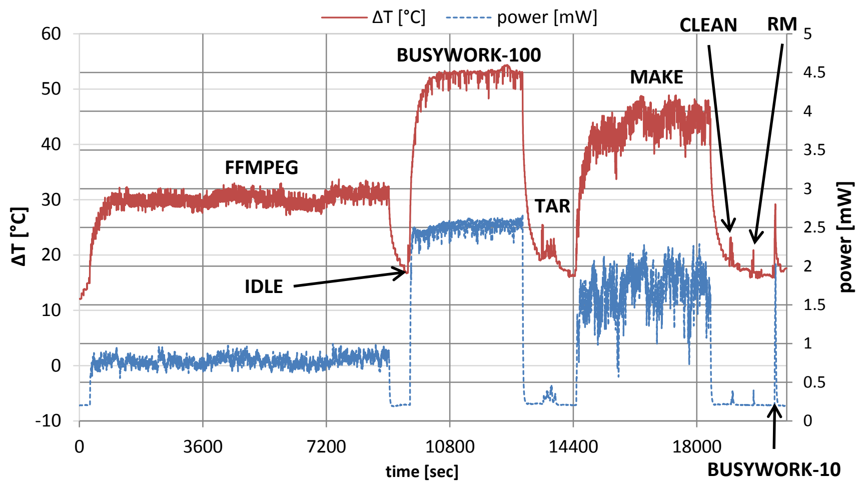


Figure 4.6: Output power of the Nextreme on Arndale.

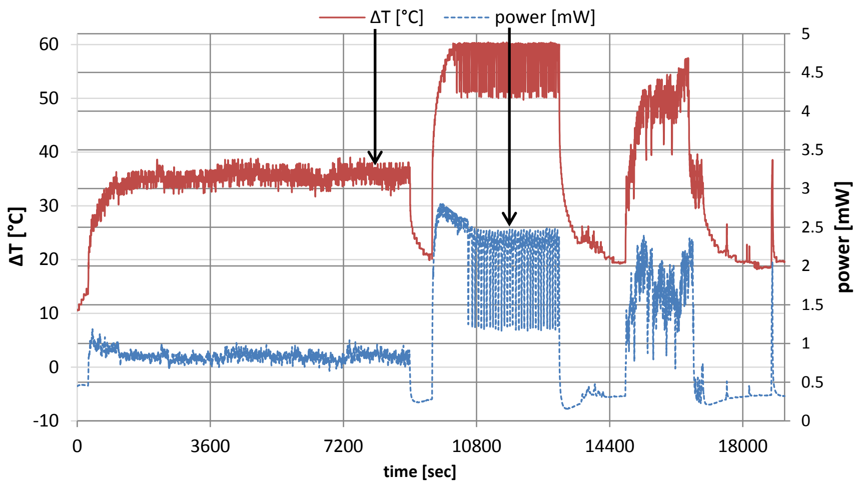


Figure 4.7: Output power of the Micropelt on Arndale.

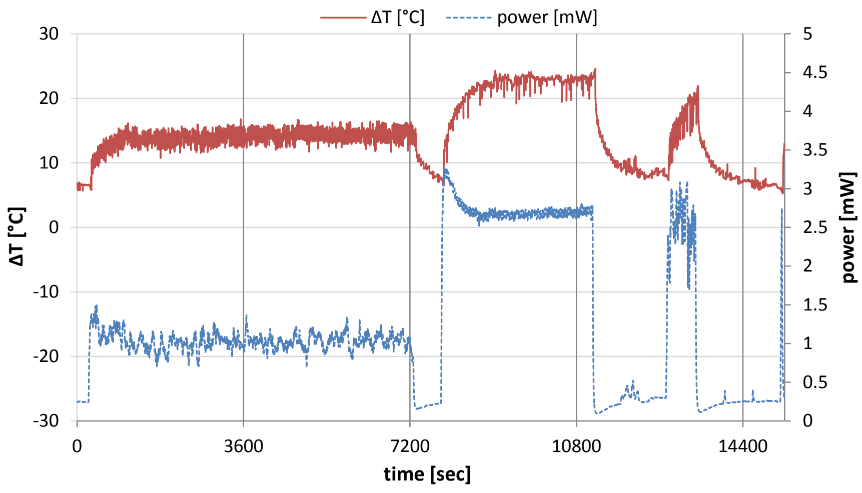


Figure 4.8: Output power of a single Peltier cell on Arndale.

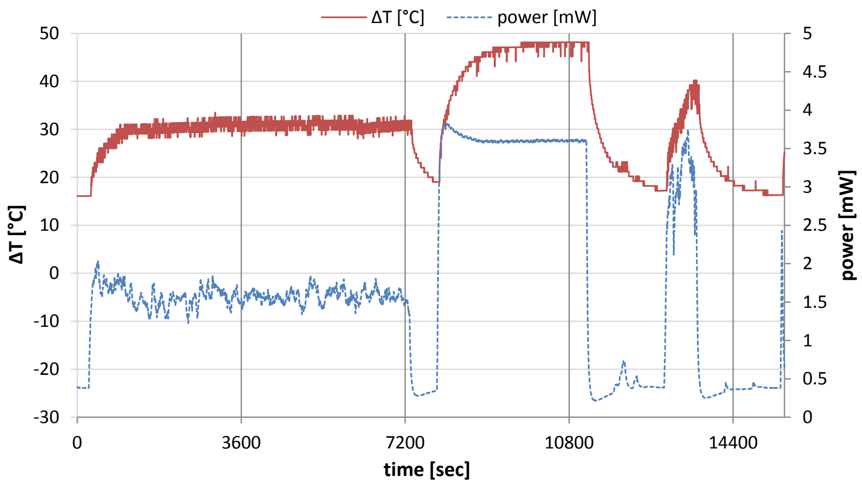


Figure 4.9: Output power of the two–stage Peltier cells on Arndale.

power profiles while others result in sharp peaks at the beginning of each task. Micropelt has the worst thermal efficiency with respect to the others: we can notice the effect of the voltage scaling in the middle of the benchmark –around 10800 seconds– clearly depicted in Figure 4.7 and less evidently in Figure 4.6, where the output power "flickering" reflects the temperature behavior, which is regulated automatically by the Linux kernel governor. The two–stage Peltier cell almost double the output power of the single–stage, the spikes move from 350 μW to 600 μW in Pandaboard's case and from 2.6 mW to 3.6 mW with Arndale.

The two–stage system exhibits higher performance with respect to other configurations. The comparative results, pictured in Figure 4.10, demonstrate that the performance of this TEG setup are better for tasks that last more than 30 seconds: `idle`, `ffmpeg`, `busywork-100`, `tar`, `make` and `clean`. Here we can notice the maximum mean impedance matched power of almost 3.6 mW on Arndale and 600 μW in case of `busywork-100`. Only in case of very short tasks, `rm` and `busywork-10` last few seconds each, the Micropelt TEG performs better and only on the Arndale board.

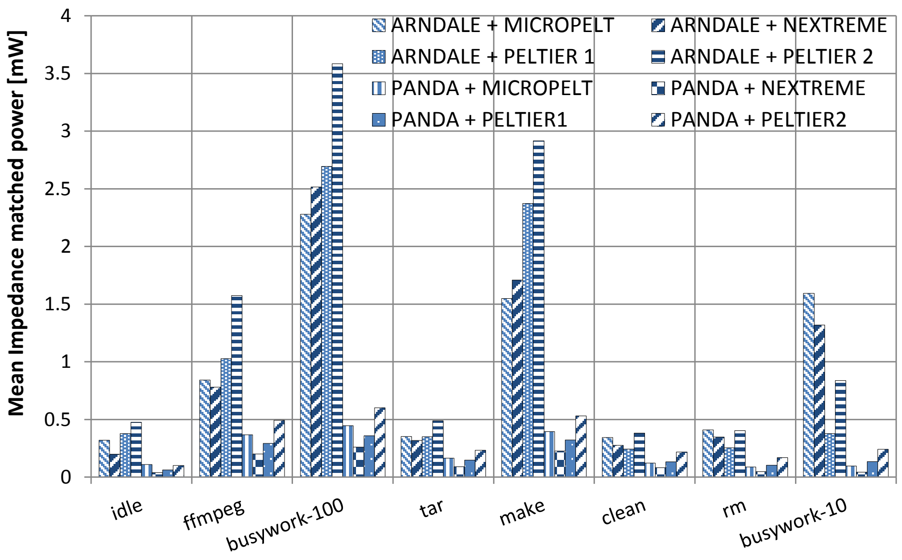


Figure 4.10: Average output power per benchmark's stage.

The two–stage Peltier cell system outperforms the other solutions also in terms of total energy. Considering the results of both Figure 4.12 and 4.13 together, the conclusion for Arndale board is that, in lower time, the stacked architecture is able to harvest the highest amount of energy (26 J in 4^h 30^m with double Peltier cell vs. 24 J in 5^h 45^m with Nextreme and 20 J in 5^h 20^m); while for Pandaboard the execution time is almost constant (26 J with double Peltier, 6 J with Nextreme and 11 J with Micropelt in 8^h 20^m). The difference in the Arndale's results is due to the effect of the voltage scaling performed by the kernel governor, which reduces the execution speed to reduce the CPU temperature. As stated

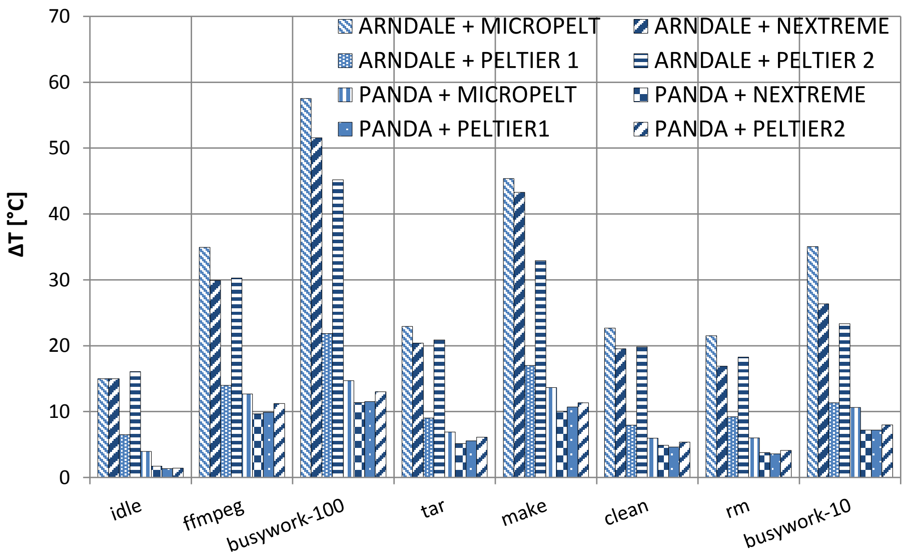


Figure 4.11: Average ΔT per benchmark's stage.

above, Nextreme and Micropelt TEGs are not able to dissipate the heat generated by the Arndale's CPU and also to efficiently exploit the high variation in the resulting ΔT.

Having a device on top of the CPU package limits the natural heat exchange of the microprocessor. If the system overheats, high temperature may cause structural damages to the ARM CPU, and may reduce the device lifetime. Also the TEG may suffer structural damages if exposed to excessive thermal gradients [35]. Therefore, we evaluated the impact of the harvesting system on the CPU temperature. The results are presented in Section 4.2.5. We can anticipate that the two–stage solution is the most suited to build an energy neutral embedded system to monitor the state of the server and its environment. We refer to this configuration for the rest of this thesis.

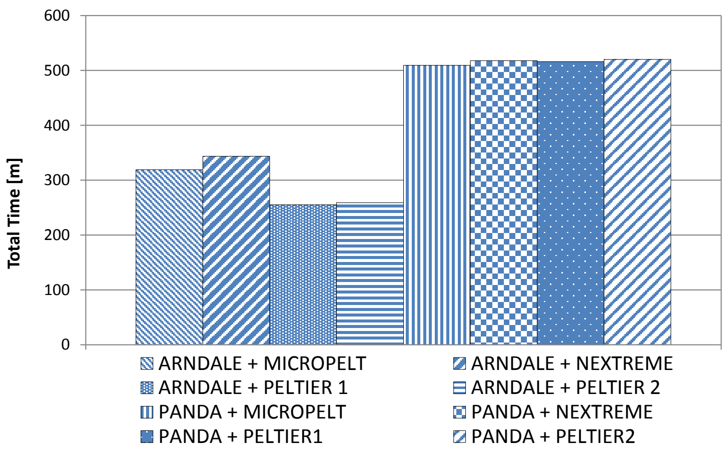


Figure 4.12: Total benchmark execution time per configuration.

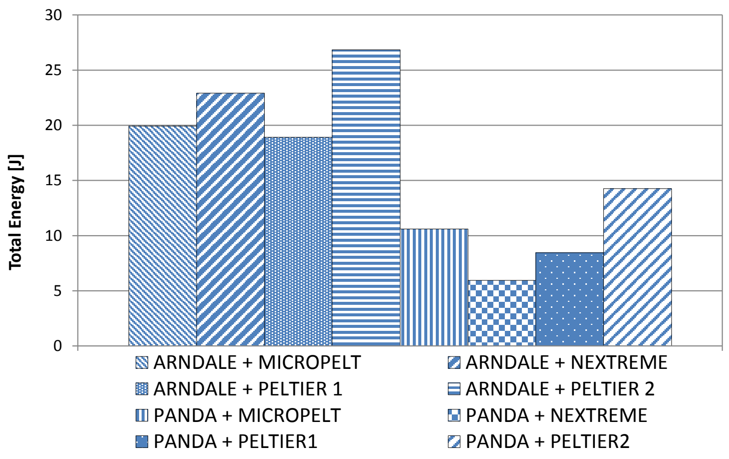


Figure 4.13: Total impedance matched energy per configuration.

4.2.3 Description of the Conditioning Circuit

The electrical power generated by TEGs is generally lower than the supply required by the wireless sensor nodes or the fan. Typically, it stands in the order of hundreds of microwatts with voltages in the hundred of millivolts. Therefore, an efficient conditioning circuit is required to raise the generated voltage up to an acceptable level to feed the system, and to accumulate energy. The sensing node operates sporadically in short bursts with a given duty cycle, therefore, the conditioning circuit must contain a storage element to accumulate the energy, and to provide energy on demand. The sizing of the storage elements depends on the kind of application implemented on the wireless node, and on the expected system lifetime. In case the system has to supply the fan, the storage element must store a voltage higher with respect to the amount required by the wireless sensor system.

Operating under a low duty cycle, the monitoring system consumes, on average, a low amount of power, and the application can be supported using a small energy buffer (a capacitance of few millifarads). On the other side, a larger storage unit guarantees the node operations for longer time, even in case there is no power at its input. In any case, the size of the storage elements is driven by the application requirements, because it depends on the amount of time necessary to accumulate the required energy.

The complete conditioning circuit, shown in Figure 4.14, is composed of a cascade of several components, namely: a step-up (boost) converter, a rectifier, a storage unit, a voltage comparator, and a LDO voltage regulator. The initial stage consists of a resonant step-up oscillator that converts a very low-power input voltage (hundreds of millivolts) into a higher voltage output. The resonator circuit multiplicative factor (σ) depends on the voltage at its input. The resonator requires at least 220 mV input voltage to let the power through the output with the amount of energy required to charge the storage unit.

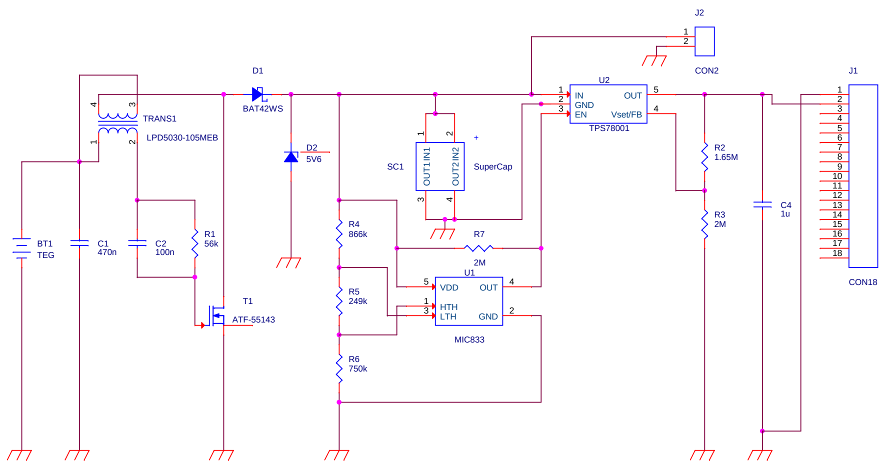


Figure 4.14: Electronic diagram of the conditioning hardware circuit that converts and stores the energy generated by the TEG, and supplies the WSN.

With an input larger than 290 mV the step-up converter provides a 5 V output voltage, with $\sigma \geq 15$.

Then, the signal is filtered, decoupled, and rectified to be used by the subsequent stage: the storage unit *SC1* which consists of a capacitive element. The storage unit may be made with supercapacitors or solid state batteries depending on the robustness required by the application. For example, with less-frequent, but energy-demanding tasks (e.g., wireless communication) a supercapacitor of a few Farads represents a good solution, since it takes a few hours to charge, with high energy density to supply powerful embedded platforms [39]. We selected a single capacitor aiming to boost recharge time rather than energy density.

The Low DropOut voltage regulator (LDO) (*U2*) maintains a steady output voltage which is lower than the input voltage source. It is needed to supply the node with a constant, proper voltage even when the voltage stored in the supercapacitor is higher, so to limit the current drop. The LDO is controlled by a comparator (*U1*) that drives its enable signal. When the charge accumulated in the storage unit is above a fixed threshold (TH_h, expressed in Volts), the enable signal rises, and the LDO starts feeding the load. The enable signal is kept high until the charge drops below the lower fixed threshold (TH_l). This configuration allows the output conditioning unit to be decoupled during the recharge phase. The two thresholds are selected by means of a voltage divider corresponding to three resistor elements in the schematics: *R4*, *R5*, and *R6*. Those three resistance value can be adjusted according to the target sensing device.

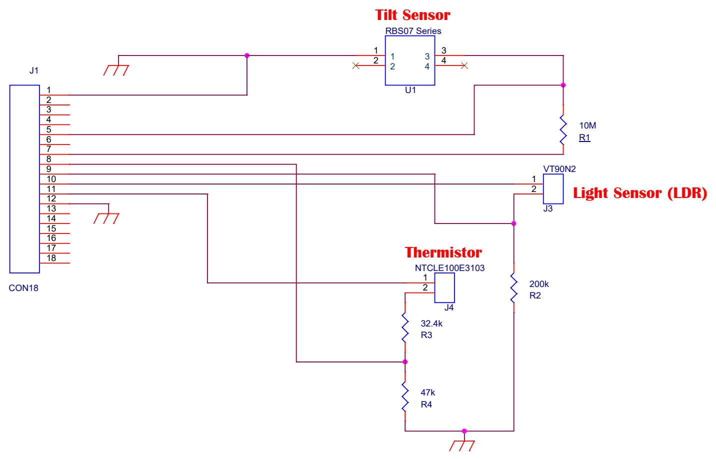


Figure 4.15: Schematic representation of the interconnection between the wireless node and the sensors.

4.2.4 TERS as Wireless Monitoring Node

We developed a wireless sensor node based on the *TI eZ430-RF2500*. The TI board is equipped with a *MSP430* microcontroller and also has radio capability. The *TI eZ430* board implements the proprietary *Simplicity* communication protocol stack, which provides advanced networking capabilities. We implemented a communication protocol that does not require setting up the radio link, the packet acknowledge, and token forwarding, since for the monitoring purposes a star-topology is most suitable, and allows the node to consume less energy when transmitting data. The *MSP430* shows an energy budget in line with the availability provided from the TEG. Several sensors have been connected to the microcontroller/ transceiver board according to the diagram reported in Figure 4.15.

A Light-dependent Resistor (LDR), the *Excelitas Tech VT90N2* [21], has been used to measure light variations, and to spot the presence of personnel inside the server room. The LDR is connected to a voltage divider connected to the ADC mounted on the *MSP430*. To minimize the power consumption, the LDR circuit is powered only during measurement. The sensor stabilizes its value after only 76 ms. A tilt-sensor, the *DIP - RBS07 Series* [48], has been attached to the node, and it is used to detect if the server rack door has been opened. When the sensor stands on its correct position, the connection is closed. If it has been tilted, for example because the door has been opened, the sensor switches the electrical connection. A thermoresistor ([67]) is placed within the heat sink central fins to measure the temperature. The wireless node collects the environmental parameters, and sends the aggregated data to a central node. Moreover, the node can exploit the temperature information to infer the thermal gradient, and estimate the expected generated power of the TEG. Exploiting this prediction, the sensor node can determine autonomously at which rate to operate in order to stay alive. To reduce the power consumption, the monitoring node powers the sensors only when it is sampling their value.

The monitoring node is fed as soon as the voltage stored in the supercapacitor reaches a voltage threshold TH_h= 3.08 V. The node boots with its registers settled for network configuration, timer usage and ADCs, and set to Low-Power Mode 3 (LPM3) with interrupts enabled. In LPM3 the node is considered to be in Idle state. After a given amount of time, I_s, the microcontroller wakes up, and broadcasts a packet to a remote node (or to a PC) with the measured value for the heat sink temperature, light, orientation, and its voltage supply (that is the voltage of the conditioning circuit storage unit). Then in returns to Idle, and reiterates after I_s seconds, assuming that the amount of energy available is enough to power it. Otherwise it will stay in Idle, or switches off in case the voltage of the supercapacitor decreases under the threshold TH_l= 2.4 V. In fact, when the lower voltage threshold is crossed, the comparator opens the path between the storage unit and the microcontroller.

We implemented two different policies to choose the value of I_s:

Fixed time: in this implementation I_s is fixed, and set equal to one second. Therefore, if the supercapacitor contains enough power to supply the node, the node continuously transmits with a constant data rate;

Temperature dependent: in the second implementation, the time between two successive transmissions is chosen by the sensor node according to the temperature measured on the heat sink of the harvester, which provides an indication of the amount of scavenged energy, used to estimate an optimal communication interval as described below.

In the first implementation, the node goes to LPM3 in the time interval between two successive transmitting events, limiting the total power consumption. However, this implementation makes it possible for the node to drain enough power to bring the voltage on the storage unit below the TH_l threshold, in case the harnessed energy is not sufficient. If this occurs, the node cannot operate until the available power returns larger than TH_h.

In the second implementation, the value of I_s is a function of the heat sink temperature. Since the temperature on the heat sink is a function of the TEG system temperature, we can infer from it the amount of power the harvester is generating, and then predict the expected amount of charge that can supply the node. The node estimates the amount of power available through a look-up-table containing the data that express the relationship between temperature on the heat sink, and generated voltage, which was derived from characterization data extracted from the prototype.

In this discussion, we have abandoned the analysis of the relation between task and generated temperature (or generated power) to adopt a more generic approach. Tasks running on the host processor have been described according to three parameters: the clock frequency f_{CLK}, the percentage CPU_{load} of CPU time dedicated to the task, and the duration I_{task} of the process. This allows us to model every software as a combination of these three values, and to associate the corresponding CPU temperature. With $f_{CLK} =$ 1.2 GHz the mean heat sink temperature stabilizes around 31.8° C, while the average CPU temperature is near 45° C. The node transmits every 30 s even if it is possible to send every 13 s with a $\Delta T = 14°$ C. As a consequence the system consumes less power than available, and the storage voltage increases. With the adjustable transmission rate implementation, after a certain period of time, the voltage in the storage capacitors tends to stabilize around 4.6 V, therefore the system is self-sustainable.

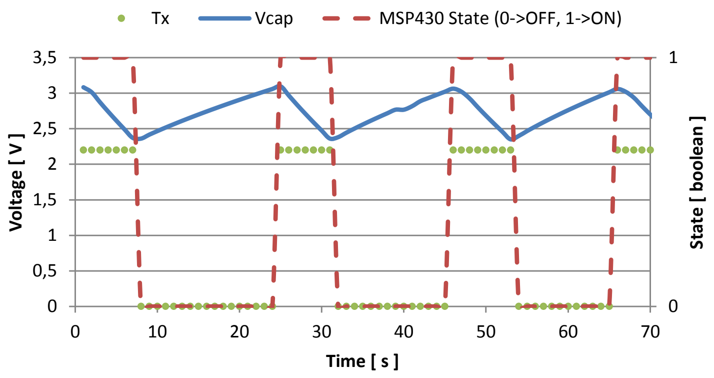


Figure 4.16: Voltage in the storage capacitance and transmission events with $f_{CLK} = 1.2$ GHz, and $CPU_{load} = 100\%$.

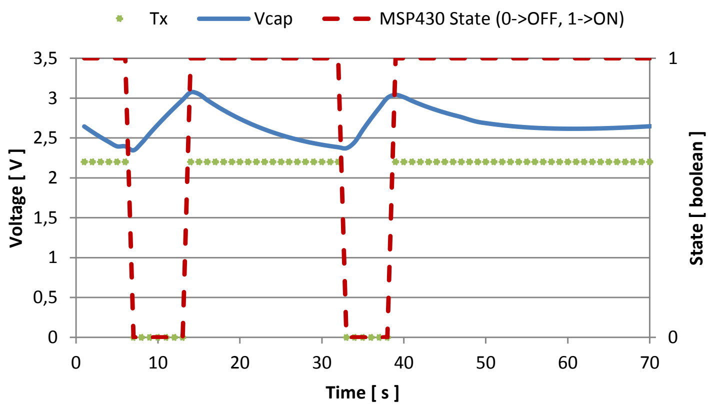


Figure 4.17: Voltage in the storage capacitance and transmission events with $f_{CLK} = 1.7$ GHz, and $CPU_{load} = 100\%$.

Every time the supply voltage crosses the high threshold, the node activates, and starts sampling and sending. In Figure 4.16, and Figure 4.17 we represented the amount of time the node is active, and the number of transmissions it is able to perform with two different values of f_{CLK}. When the host processor runs with $f_{CLK} = 1.2$ GHz and $CPU_{load} = 100\%$, the recharge time required to reach the voltage threshold is about 25 s, and after being activated, the monitoring unit is able to send data for 7 or 8 times before discharging the supercapacitor, and becoming inactive. This condition is represented in Figure 4.16, while Figure 4.17 shows measured data when the host processor clocks at its maximum speed: $f_{CLK} = 1.7$ GHz and $CPU_{load} = 100\%$. In the latter case, it is possible to perform continuous transmissions (every second) without discharging the supercapacitor once the temperature reaches its steady state. Moreover, just 6 seconds are required to recharge the storage unit.

In Figure 4.18 and Figure 4.19 we show the activity of the node when the $CPU_{load} = 50\%$ for $f_{CLK} = 1.2$ GHz, and for $f_{CLK} = 1.7$ GHz respectively, together with the voltage across the storage supercapacitor. Even with half the CPU load, the energy harvesting system is able to recharge the storage supercapacitor. For $f_{CLK} = 1.2$ GHz, the monitoring system can sample the sensors and send the data six times before going under threshold. After that, 33 seconds are required to restore the charge. For $f_{CLK} = 1.7$ GHz, the monitoring unit senses and sends eleven times and switches off for 8 or 9 seconds before resuming.

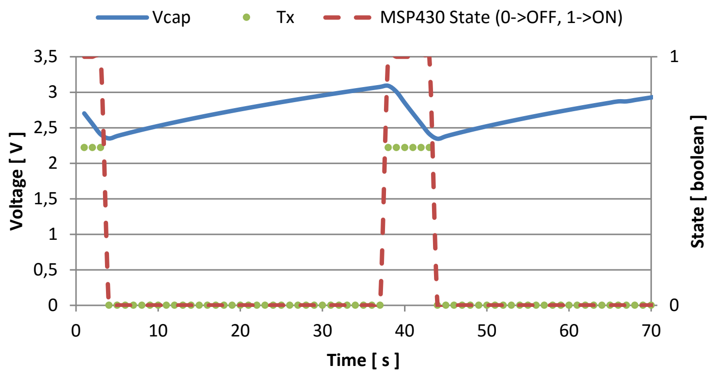


Figure 4.18: Voltage in the storage capacitance and transmission events with $f_{CLK} = 1.2$ GHz, and $CPU_{load} = 50\%$.

The power consumption of the wireless sensing node has been characterized by measuring the voltage across a resistor R_{meas} placed in series between the sensing node and the power supply circuitry. Through this approach, and by knowing the resistance $R_{meas} = 10.2\ \Omega$, we computed the effective power consumption. By analyzing how the values of the power consumption evolve with time, we also associated the specific task the node is performing to the corresponding amount of energy spent. Table 4.1 summarizes the task specific power consumption value as extracted from Figure 4.20. From

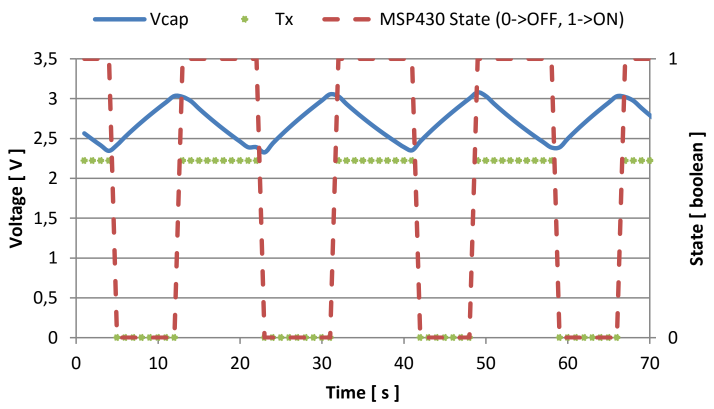


Figure 4.19: Voltage in the storage capacitance and transmission events with f_{CLK} = 1.7 GHz, and $CPU_{load} = 50\%$.

Table 4.1 it is possible to compute the amount of energy required by the system to perform a single sense-and-send operation, that is $E_{TX} = 84.69$ μJ. Moreover, knowing that for each send operation the radio is used for $I_{TX} = 4.86$ ms, the total current consumption of the *MSP430* board in Idle mode together with its radio module *CC2500* is [26, 25]: $MSP430_{IDLE} + CC2500_{IDLE} = 1.3$ μA. The power consumption is: $P_{IDLE} = 2.2$ V $\times 1.3$ μA $= 2.86$ μW. Therefore, to supply the system in Idle mode for one second 2.84 μJ are needed. Summing all together, to run the application on the board for one second, $E_{tot} = 87.53$ μJ is the required total energy.

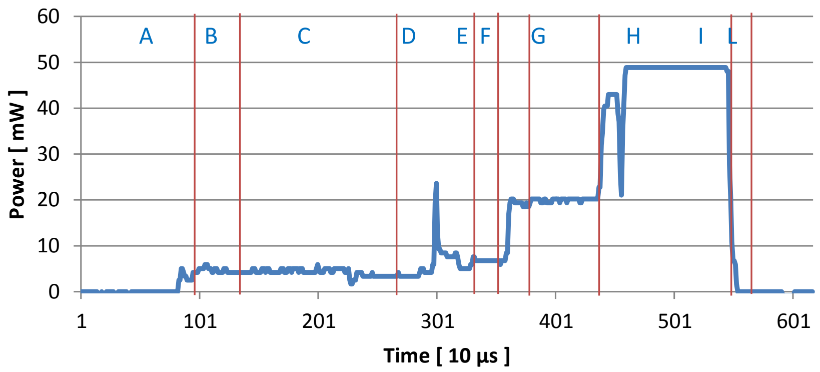


Figure 4.20: Measured power consumption of the prototype node during a sense and send event.

4.2.5 TERS as Passive CPU Cooler

We evaluated the cooling capability of the harvesting system placed on top of the CPU package. In our experiment we started with the host device in Idle and we launched a process with (f_{CLK}, CPU_{load}, I_{task}) = (1.7 GHz, 100%, 120 s). We measured the CPU temperature in three different cases: the CPU without passive dissipation (as it is sold

Table 4.1: Power consumption of the *eZ430-RF2500* board during a sense-and-send event.

Symbol	Event	$\bar{I}$ mA	Time ms	P mW	E μJ
A-L	Idle	1.3×10^{-3}	-	-	-
B	Temp & Vcc	1.94	0.38	4.26	1.61
C	NTC Temp	1.94	1.37	4.26	5.83
D	Calc & XOSC Startup	2.56	0.70	5.63	3.94
E	Ripple Counter	3.02	0.21	6.64	1.39
F-G	Msg & PLL calib.	8.29	0.87	18.23	15.86
H	Tx Mode	20.95	1.19	46.09	54.84
I	Switch LPM	3.95	0.14	8.69	1.21

by the vendor), with the heat sink on top (the same heat sink mounted on top of the harvesting module), and with the complete proposed prototype (harvester and heat sink). The experiment is used to understand whether the system causes the CPU to overheat. One the contrary, it turns out that our prototype allows the CPU to work at lower, safer temperatures. Results are depicted in Figure 4.21. As the reader can easily notice, the temperature of the CPU without heat sink rapidly reaches the safety threshold (about 90° C). If we had not turned off the kernel DVFS, this would intervene to reduce f_{CLK} and prevent overheating. The presence of the heat sink allows the temperature to not exceed 62° C. Even better, when the complete harvester is mounted the temperature reaches only 55° C. Therefore, the proposed system is an excellent passive heat sink which ensures a reduction of temperature of about 5° C in Idle mode, and up to 7° C when the target host runs at its maximum performance. Compared to the host deprived of the heat sink, the harvester guarantees to cool the CPU by about 35 degrees. This is contrary to other similar applications where the presence of a TEG in between a desktop CPU package and the heat sink raises the CPU temperature by about 10–20° C [28, 38].

4.2.6 TERS as Active CPU Cooler

Generally, data centers machines work at $f_{CLK} < f_{CLK}^{MAX}$. However, it is sometimes necessary to run at maximum performance, even overclocking the machine to speed up computations and/or serve concurrent processes within a time limit. In these cases, the thermal management is crucial to guarantee the safety of the computing units. To address this scenario, the energy neutral hybrid cooling system works in symbiosis with the host CPU. To provide active cooling, we equipped the system with a $40\phi \times 5$ mm fan and an additional electronic circuit to control the fan. The fan is rated at 5 V and 250 mA.

By experimental evaluation, we measured the minimum voltage (4.4 V) required to switch it on and reach a speed sufficient to cool down the heat-sink within the safety

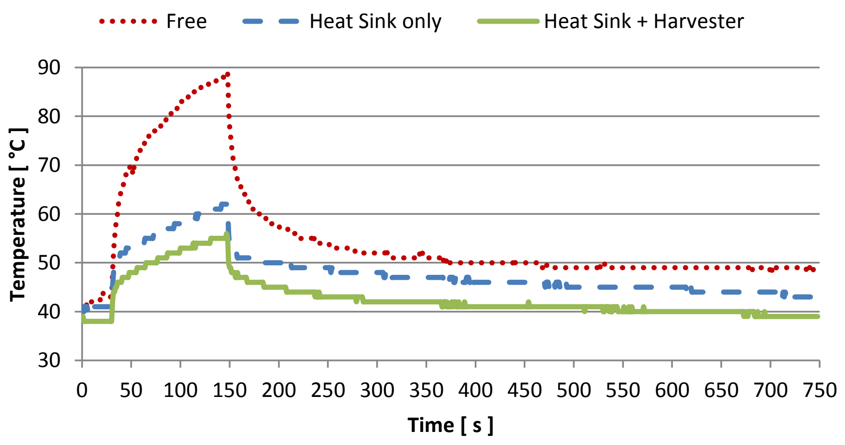


Figure 4.21: CPU temperature trend in three configuration: with and without the heat sink, and with the proposed prototype on top.

threshold. In this case the storage unit is made of two supercapacitors arranged in a reconfigurable bank block (CBB, or flying capacitors). The CBB are charged in parallel to speed up the process, then switched in series to reach the 4.4 V required by the fan. The serial configuration takes 5× more time to collect the same amount of energy with respect to the parallel one (≈ 600 s versus ≈ 130 s). Considering the average energy collected by the TEG, this is still an improvement in speed of recharge even if it has been measured in the order of ten hours. Supercapacitors are charged up until the voltage across them reaches the threshold TH_{cap}. The value of TH_{cap} has been set equal to 2.2 V because capacitors are charged in parallel and the series equivalent 25 F provides enough current to switch on the fan (thanks also to the low internal resistance of the supercapacitors). In this case, the comparator is used to check the storage charge status, its thresholds are V_{MAX} equal to 2.2 V and V_{MIN} equal to 2.0 V. Moreover, the circuit includes –even if those are not shown– the switches to commutate the storage elements and to turn on the fan by direct connection; both are realized with relays in our experimental workbench.

To illustrate how the active cooling system works in coordination with the host CPU overclock, we describe the procedure for the Arndale Board. In normal conditions, the host CPU carries out its tasks at clock frequency f_{CLK} = 1.5 GHz while the harvesting module scavenges energy. When the stored energy is above the threshold TH_{cap}, the system can afford an overclocking phase. Immediately after the charge in the supercapacitor reaches the required threshold, the host CPU switches to its maximum clock frequency f_{CLK} = 1.7 GHz. The fan is not switched on simultaneously with the overclock start. This choice allows us to extend the overclocking total time and the energy collected by the TEG because of two reasons. First, during overclock the harvested power increases as the thermal gradient ΔT increases. Moreover, the CPU temperature rises up slowly thanks

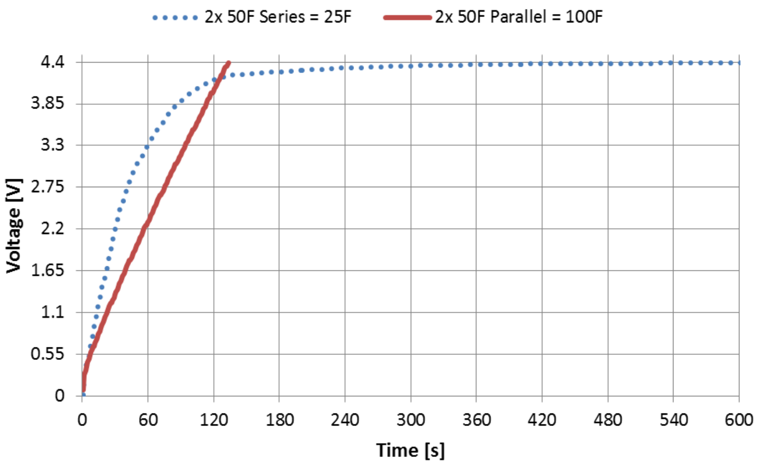


Figure 4.22: Voltage stored in two 50 F supercap in time, parallel versus series configurations.

to the passive cooling capabilities offered by the presence of the two–stage and heat–sink. While in overclock mode, the system checks the CPU temperature periodically until it matches the TH_{CPU} threshold. The TH_{CPU} threshold has been set a little lower with respect to the maximum design temperature of the CPU to prevent circuit overheating. At that time, the fan is switched on for a programmable time amount (active cooling phase). During active cooling, the CPU temperature decreases by an amount that depends on how long the fan is spinning. After the active cooling phase, the CPU can continue to work overclocked since its temperature is decreased. The second time the TH_{CPU} is exceeded the host CPU clock frequency slows down to the normal operating mode (f_{CLK} = 1.5 GHz), since not enough energy remains in the storage unit to supply the active cooling.

To evaluate the global performance of the system used for cooling, we conducted a test with the ArndaleBoard, and the two–stage harvester configured for cooling as introduced in the prototype description. The workload of the host CPU is always set at CPU_{load} = 100%, and f_{CLK} that changes according to the description presented in the previous paragraph. Even if a continuous fixed workload is not realistic to model the usage of data-centers and high performance computing machines in general, this choice is justified by the fact that we wanted to evaluate both the performance in terms of energy harvested from the heat and the passive heat dissipation in very harsh conditions. It would not be possible to test all the possible workload scenarios and there is no benchmark to evaluate TEGs performance with computing units, so we designed the above to be simple and easily reproducible on any other platform. These settings represent the best conditions for energy harvesting during normal mode, on the other hand they represent the most unfavorable conditions as regards the amount of time the CPU spends overclocked. The temperature threshold is set as $TH_{CPU} = 68°$ C, that is 2° C lower than the maximum acceptable temperature of the ARM CPU.

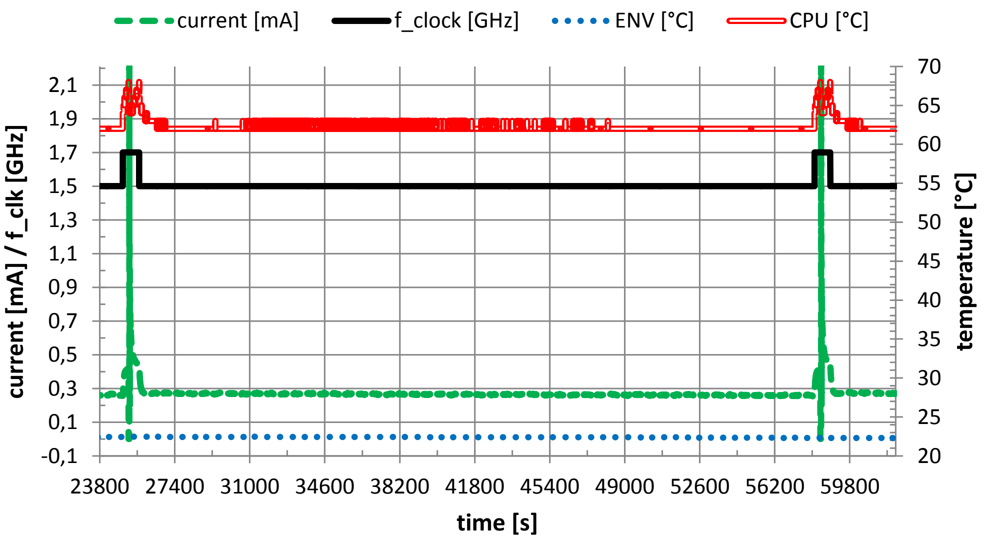


Figure 4.23: Long term stability of the passive cooling with 100% CPU load @ 1.5 GHz. Time required to recharge the supercap.

Figure 4.23 depicts the time interval required by the prototype circuit to recharge the storage supercapacitors after a cooling phase. It is meaningful to underline the ≈ 10 hours required in the reported case that refers to a supercapacitor $C_{SC} = 100$ F. This picture also presents the long term stability of the thermal performance of the system, since in between the two spikes (two active cooling instants) the CPU temperature is basically constant (the top curve that oscillates between 62° C and 63° C). In this picture the current is the TEGs output that charges the storage, f_{CLK} is the CPU clock frequency, and the lower stable dotted–line represents the environmental temperature.

We evaluated different activation times for the cooling fan ranging from 2 up to 30 s, but the most interesting are with 10 s and 30 s cases. The 30 seconds case represents the limit we imposed because of the amount of energy at our disposal. An example result in this case is depicted in Figure 4.24. Here the CPU temperature increases two times above to the threshold $TH_{CPU} = 68°$ C, the first occurrence triggers the drop in the charge current (which means the stop of the recharge to power the fan); the second time it triggers the end of the overclock phase. The 10 seconds active time instead is interesting because it underlines the very impacting effect of the environmental temperature on the performance. In this case we obtained the same overclocking length in time we get for the 30 s case (of 13 minutes) with only 2.5° C difference in the environmental temperature. With temperature same as above, the 10 s activity resulted in few minutes of overclocking

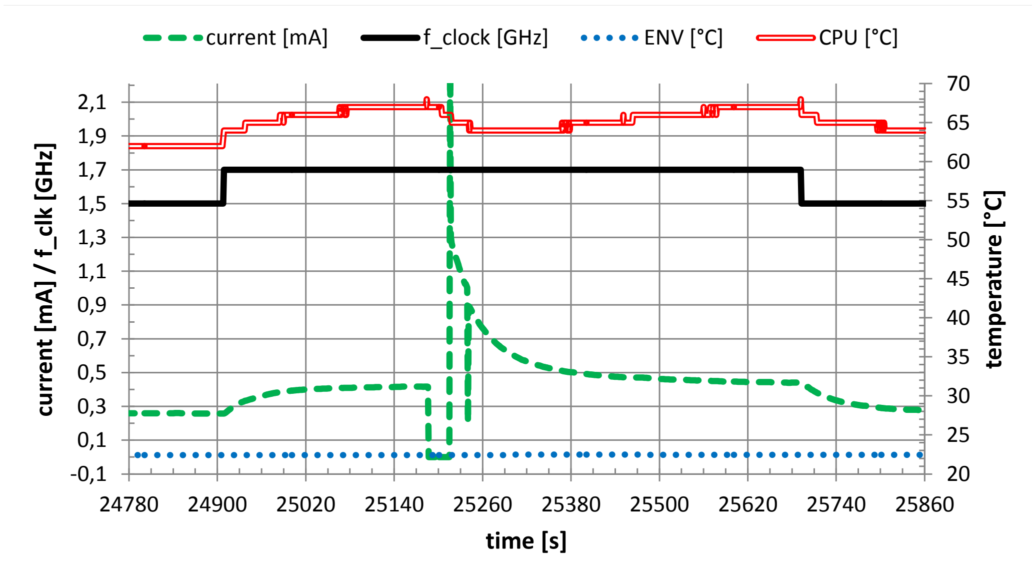


Figure 4.24: Active cooling, fan activity of 30 s allows 13 min overclocking @ 22.5° C.

(see Figure 4.25).

4.2.7 Characterization of the harvesting module and conditioning circuit

To obtain a formula to compute the energy generated by the TERS, we characterize the system composed of the two–stage harvester with the conditioning circuit (without storage). We did this analysis in two steps. First, we studied the correlation between the task that the CPU is performing (expressed in terms of f_{CLK}, CPU_{load}, and I_{task}), and the thermal gradient (ΔT) that builds up between the CPU package and the heat sink. Then, we deduced the function that links the thermal gradient with the TERS output power.

We obtained the thermal model of the CPU cores by letting the board run a given task for the amount of time required for the CPU temperature to stabilize. To generalize our analysis, rather than referring to specific software applications, we identify a task the CPU is running as a combination of three values: the CPU clock frequency (f_{CLK}), the percentage of activity in the CPU devoted to the task execution (CPU_{load}), and the amount of time required to complete the task (I_{task}).

For each combination of the three CPU values, we obtain a value of the generated thermal gradient $\Delta T = T_{hot} - T_{cold}$. The input-output relation of the harvester, which links the thermal gradient to the generated power, has been obtained by measuring the internal temperature of the host CPU (T_{hot}), the temperature of the heat-sink (T_{cold}), and the corresponding output power P on our workbench. The results are influenced by transients, however we determined a reasonable mathematical relation using the same approach that

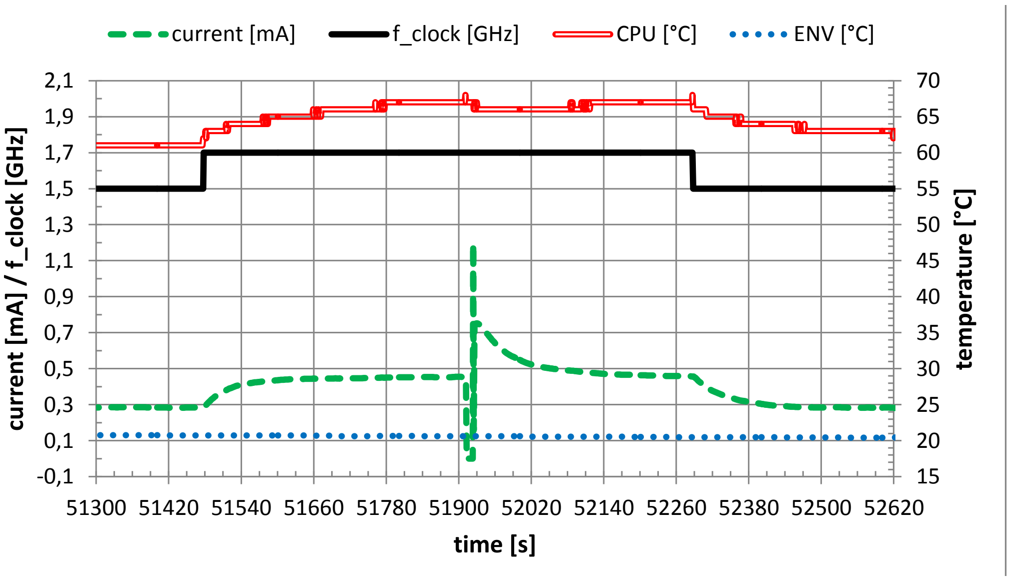


Figure 4.25: Active cooling, fan activity of 10 s allows 13 min overclocking @ 20° C.

was used to characterize a previous version of the same harvesting circuit [39]. The resulting fifth-order polynomial is:

$$P(\Delta T) = a_0 + \sum_{i=1}^{5} a_i \cdot \Delta T^i, \tag{4.1}$$

where the coefficients describing our demonstrator are:

- $a_0 = -0.6345$
- $a_1 = 0.2852$
- $a_2 = -0.0465$
- $a_3 = 0.0033$
- $a_4 = -9.5715 \times 10^{-5}$
- $a_5 = 9.9485 \times 10^{-7}$

Figure 4.26 shows the cloud of dots representing measured values in green, and the fitting curve that is expressed by the polynomial in orange. The amount of generated power that can actually be saved in the storage unit of the harvesting device depends on the conditioning circuit efficiency η. To determine the circuit efficiency, we evaluated the difference between the ideal energy for the storage capacitance C_{sc} when charged with a voltage V equal to the voltage available at the TERS output, and the energy actually stored in the capacitor on our workbench. The efficiency is inversely proportional to the voltage, and, in the specific case, it ranges from 20% up to 58% for an input voltage in between 240 mV and 500 mV. Thus, the value of harvested energy is: $E = \eta \cdot P(\Delta T)$.

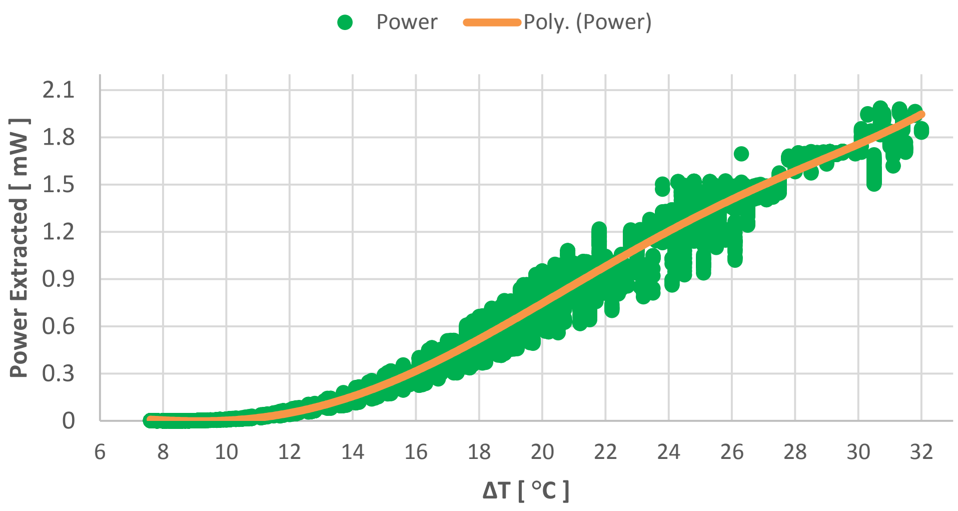


Figure 4.26: Relation between thermal gradient and extracted power. Green dots represent measured values, the orange line represents the fitting polynomial used by the simulator

4.3 Modeling TERS

Implementing the proposed methodology, we first consider the physical models. The mathematical models have been obtained from the analysis of data collected in a dedicated experimental setup and then implemented in C/C++. We implemented the CPU heating function, the formula for the conversion of thermal gradient into electrical energy and the charge/discharge of the supercapacitor. Likewise, we characterized the energy consumption of the MSP. After that, we defined the relation between the tasks that run on the CPU with the generated temperature, and the status of the MSP with its power consumption to define the architecture models. Then, the physical–architectural components were combined with the cyber–architectural components to define the complete design.

Unlike the previous case, where events were logically ordered in time, in this system some state changes of a component are triggered by events that take place in other components, whose duration is not known in advance. This requires a more careful account of the synchronization. In fact, the data center *host* and the monitoring system functionalities are separated, but their activation is related by physical processes. The activity of the monitoring system configured to perform CPU cooling is triggered by the temperature of the ARM CPU, but the ARM CPU and the MSP are not connected from the point of view of the architecture, so there is no direct interface to trigger the event. The situation is similar for the monitoring configuration, which works as long as the recovered energy is sufficient to power the microcontroller and the sensor, independently of the state of the *host*. Events of the physical quantities are not handled by the scheduler, so the corresponding state transitions are triggered through direct interfaces which, in this case, relate the two CPUs. A representation of the METROII model is depicted in Figure 4.27. In METROII, schedulers are used to specify the logical order of begin and end events of functions, and events are used by the mapper to synchronize elements. But for the reasons listed above, not all the events are handled by the scheduler, because the start event of a functionality may be fired when a physical quantity of another component satisfies a given criteria. Moreover, the time annotator is used to specify the timing resolution of the simulation.

4.4 Physical Model

The physical model of the TERS is composed of the *ARM CPU*, the *TEG*, the *MSP*, and the interfaces between the aforementioned components. From the physical model point of view, the *ARM CPU* is a source of heat; and the *MSP* is a component that consumes the power stored in the *TEG* component. Unlike the cyber model, in the physical model, *ARM CPU* and *MSP* are connected together through the *TEG* component. The physical models of the two CPUs handle the physical processes and interactions (heat exchange and power consumption). The *ARM CPU* component includes the model of thermal dissipation of the processor implemented as theoretical heating and cooling curves. The theoretical heating curve defines the temperature gradient trend over time considering a constant ambient temperature as:

$$\Delta T(t) = \Delta T_{max} \cdot (1 - e^{-t/\tau})$$

where ΔT is the thermal gradient between the CPU package and the ambient temperature, and ΔT_{max} is the target temperature difference. τ is the time constant of the system, that is the time required for the system to reach the target temperature ΔT_{max} in ideal conditions (i.e., if there were no heat exchange with the environment). Conventionally, τ corresponds to the time required for the system to reach the value $0.632 \cdot \Delta T_{max}$, and the system is considered to be in steady state after about $4 \div 5$ times τ.

When the CPU starts performing a more demanding task, the simulator computes the current value of the thermal gradient as:

$$\Delta T(t) = \Delta T_{begin} + \Delta T_{target} \cdot (1 - e^{-t/\tau})$$

with ΔT_{begin} the value of temperature at the beginning of the new task, ΔT_{target} the steady state temperature of the new task. On the contrary, when the CPU switches from a more demanding task to a less demanding activity, or to the idle state, or when the cooling fan intervenes, the thermal gradient value is updated according to:

$$\Delta T(t) = \Delta T_{target} - (\Delta T_{target} - \Delta T_{begin}) \cdot e^{-t/\tau}$$

The values of the target temperatures were measured on our workbench during the characterization phase of the prototype device; and, observing the transitions between different tasks, we also derived values for τ.

The relation between the thermal gradient and the generated output power of the TEG is implemented inside the *TEG* physical/architectural component. This component contains the thermoelectric generator and the supercapacitor physical models. The thermoelectric generator model computes the actual generated power of the TEG as a function of the thermal gradient. The super-capacitor charge and discharge model is included in the *TEG* component. The generated power makes the voltage inside the super-capacitor increase as:

$$V_{sc}(t) = \sqrt{V_{sc}(t-1) + \frac{2V_{generated}(t)}{C_{sc}}}$$

When the *MSP* is powered on, it consumes power according to its state and the task it is executing. Therefore, the *TEG* component diminishes the storage voltage accordingly, with:

$$V_{sc}(t) = \sqrt{V_{sc}(t-1) - \frac{2V_{consumed}(t)}{C_{sc}}}$$

The interface between the *MSP* and the *TEG* components models the energy consumption of the *MSP*. The supplyPower interface is used to trigger state transitions of the *MSP* which works according to the energy at its disposal. For example, the interface is used to switch on the *MSP* when the voltage stored in the *TEG* supercapacitor increases over the threshold of operation of the *MSP*. This kind of state transitions are modeled directly, and not using the METROII scheduler because physical quantities are not visible to the event scheduler.

4.5 Cyber Model

The cyber model includes software and hardware components of the system. The model of the TERS is composed of the *host server*, the *FIFO* component, the *Host_Task*, the *ARM CPU*, the *Cooling_Task* or the *Sense_Task*, and the *MSP*. In particular, the *host server* component models the functional activity of the server (i.e., jobs/tasks the server executes). It interfaces with a *FIFO* component used to capture the job start and finish events. The *host server* is mapped into the *Host_Task* that associates the current job to a a set of parameters, such as the clock frequency, the load percentage and the job duration. The two CPUs cyber components (ARM and MSP) provide the execution to their respective tasks. The *ARM CPU* provides the execution of the jobs requested by the *Host_Task* with specific parameters. The *MSP* cyber component models the execution of the firmware implemented in the wireless sensor node. The *Cooling_Task* and the *Sense_Task* model two firmware which correspond to two possible activities that the *MSP* can provide. The first contains instructions to control the *ARM CPU* clock frequency, and to supply the cooling fan to implement the active cooling task that supports *ARM CPU* overclocking. The *Sense_Task*, instead, contains the instructions to sample the sensors that is used to implement environmental monitoring with WSN. From the cyber standpoint, the *ARM CPU* and the *MSP* are disconnected, because their tasks are unrelated.

4.6 Functional Model

The TERS functional model is made of two components:

host device : this component represents the ARM CPU of the data center;

blocking channel consisting of a symbolic blocking *FIFO*. This component is used only to capture the functionality of the *host* CPU, and for METROII to represent the *begin* and *end* event of the computations made by the *host* device.

Each *host* device is connected to its corresponding blocking channel. The functional model only captures the execution of the *host* device to calculate the amount of processing done as a function of the time required for processing, the clock frequency and its load percentage devoted to the current application. The functional model of the system is so simple because we it models just the activity of the data center CPU.

4.7 Architectural Model

The architectural model of the TERS is more populated, and the model for a sensing device requires different components with respect to the model for a cooling device. The Architectural Model comprises:

arndaleProcessor : a component which models the ARM CPU of the host device and provides computation functionality; it also works as the source of thermal energy which is provided at the output as data to be forwarded to other components;

TEG : which models the behavior of the thermoelectric generator, and contains the energy storage unit (super-capacitor);

MSPProcessor : that models the architecture of the wireless sensor network node.

The architectural model includes the Tasks that can be mapped into the computing platform:

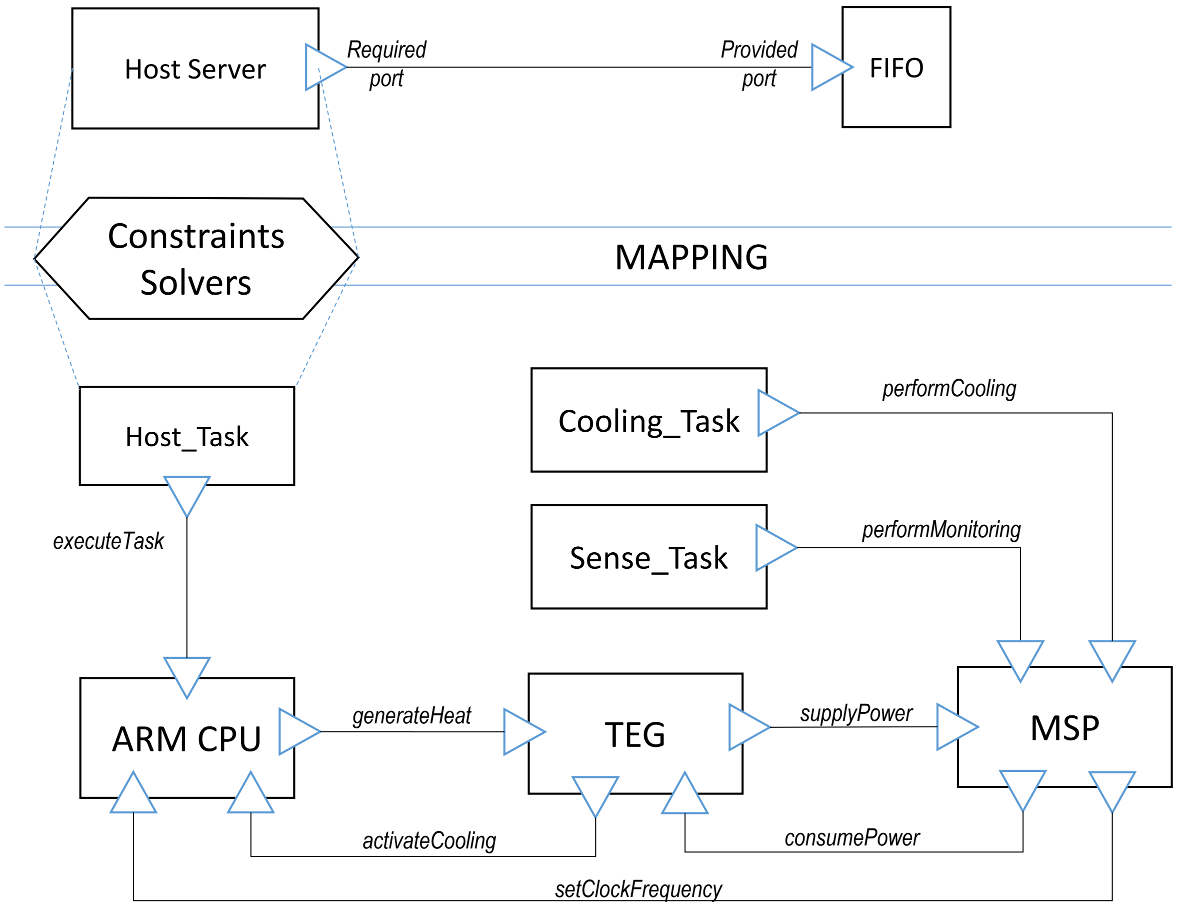


Figure 4.27: The METROII model for the thermoelectric energy recovery system applied to the ARM CPU. The figure includes both the *Sense_Task* and the *Cooling_Task* that can be implemented alternatively.

1. a *Host_Task* represents the activity running on the host processor,
2. a *Sense_Task* represents the behavior of the wireless sensor node when it is configured to perform monitoring,
3. a *Cooling_Task*, that can be used as an alternative to the *Sense_Task* to model the behavior of the TERS device if configured to perform active CPU cooling to the *host* device.

The model describing a system configured to perform monitoring includes a *Sense_Task*, while the model describing the active cooling system for overclocking must include a *Cooling_Task*. The physical quantities and the control of the state evolution of all the components that build the system are part of the architectural description, because state evolution depends on physical quantities (i.e., CPU temperature, and supercapacitor state of charge).

The *Host_Task* component contains parameters such as the load percentage devoted to the required functional activity, the clock frequency, and the CPU package temperature. The temperature is computed at each iteration as a function of the CPU clock frequency and load according to the values derived from measurement made on the prototype device [64]. We implemented a thermal model for the Pandaboard and a thermal model for the ArndaleBoard. Both the models consider the two–stage peltier harvester on top of the CPU package. To simulate a different harvester on the same boards, a different thermal model must be included om the *Host_Task* component.

The *TEG* components parameters specify the efficiency of the *TEG* (η), the storage capacitance (C_{sc}), the supercapacitor maximum voltage (V_{sc}^{max}), and the present stored energy. The *TEG* component interfaces with both the host *CPU* and the *MSP* components. The *TEG* component receives from the *ARM CPU* component the CPU temperature which is used to determine the present harvested energy, according to the formula that has been derived from measurement made on the prototype device. The *TEG* interface with the *MSP* receives request of energy, that can be served only in case the stored energy is greater than the voltage threshold that allows the *MSP* to activate. If the *MSP* is active, the *TEG* component receives request of energy, and subtracts from the present storage voltage the amount of energy required according to the task that the *MSP* component is performing. In particular, the *MSP* can be in *Idle mode* when it is active but performing neither the *Sense_Task* nor the *Cooling_Task*. If the *MSP* is performing the *Sense_Task*, it samples the sensors and transmits the data over the wireless channel, and at the same time the *TEG* component decreases the amount of stored energy accordingly.

When a *MSP* is configured to perform the *Cooling_Task*, the iteration among components becomes more articulated because of how the blocks interact with each other in order to implement the over clocking and cooling procedure discussed in [61]. First, given that the fan power consumption to perform cooling is higher than the power required to perform monitoring, when we want to simulate the cooling procedure, the value for the activation threshold of the *TEG* component is set to the corresponding, and greater, value. When the harvested energy is enough to sustain the cooling activity, the *MSP* component sends the command that makes the CPU run with the maximum available load, and at the maximum clock frequency to the corresponding *ARM CPU* component. As a consequence, the remaining time needed to perform the running job decreases, while the CPU temperature increases. If the temperature were to reach the threshold of 82° C, the dynamic voltage and frequency scaling (DVFS) of the *host* device would intervene to scale down the clock frequency, thus reducing the generated temperature. But in this case, DVFS is not activated because the monitoring device is ready to cool down the CPU package with the following procedure. When the CPU temperature reaches 80° C the *MSP* feeds power to the cooling fan. At that time, the CPU temperature is decreased according to the fan specifications and the energy stored in the super capacitor is subtracted accordingly; at the same time, the *host* CPU continues to run at its maximum clock speed continuing to generate a lot of heat. After a while, the CPU temperature reaches again the 80° C threshold. At that time, if the stored energy is sufficient, the *MSP* can again activate the fan; otherwise, it will set the CPU to a lower clock frequency to prevent overheat. In the latter case, the host time left to complete the job is calculated again in compliance with the clock speed and load.

4.8 Mapped Systems

We mapped the design onto two different host devices: a 1 GHz ARM Cortex–A9 CPU Pandaboard with 1 GB DDR2 RAM, and a Samsung Arndale equipped with a 1.7 GHz dual core Cortex–A15 with 2 GB DDR3 RAM. We chose these two embedded boards because they are representatives of a class of systems with reasonable computing performance and a low heat dissipation. The simulator considers the thermal model of the corresponding *host* device. The model is written in C within the *host* component of the architectural model. We considered different values for the storage capacitance (C_{sc}), and two different cooling fans. The fan parameters in the simulator include the required voltage of operation, the power consumption, and also their cooling capacity. Objective of the design exploration is to determine which set of parameters representing possible implementations, guarantee the best overall performance in terms of monitoring sampling rate and host performance.

4.9 Results

We analyze the results of the presented model estimating processing time, amount of samples transmitted, number of cooling events, amount of residual energy on the storage supercapacitor, as well as simulation time for different configurations. Figure 4.28 illustrates the number of cooling events and the number of packets sent during one week of operation of the system using the ArndaleBoard as a *host*, while Figure 4.29 shows the result obtained conducting the same analysis using the thermal model of the PandaBoard as *host*. We mapped the system using models we gathered from measurements of the devices. For each design, we evaluated the performance of the monitoring system configured to perform either cooling or sensing. We pick the design tm1C1 for deployment because it ensures the MSP executes a large number of cooling and sensing events maintaining a small size for the TEG compared to the solution with more Peltier cells (namely the

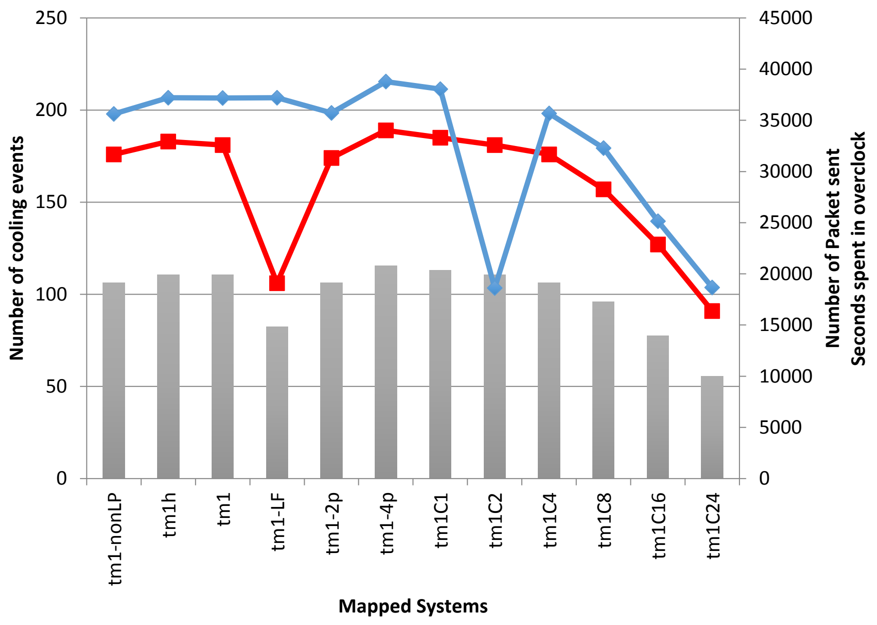


Figure 4.28: Average number of cooling (in red) and sensing events (in blue) in a week for different configurations of the monitoring device applied on top of the Arndale CPU. Seconds the CPU spent in overclock (represented by bars).

tm1-4p). The very same TERS design exhibits good performance also applied to the Pandaboard (tm2C1). However, the advantage of having more Peltier cells is more evident on the Pandaboard than in the Arndaleboard (tm2-4p vs. tm1-4p). Having larger storage supercapacitor is more detrimental for the monitoring activity on the Pandaboard than on the ArndaleBoard (see blue markers for tm2C16 compared to tm1C16). In fact, the design tm1C16 makes 71.4% the number of activities of the design with a storage capacitor four time larger (tm1C4). On the contrary, the design tm2C16 sends only 37.5% the number of packets sent by the design on the Pandaboard tm2C4.

The simulation of one week of operation of 100 host processors, and their corresponding TEGs, with 90 monitoring devices configured to perform cooling and 10 configured for monitoring, requires about 2 hours of computation time. To simulate 2 monitoring system, one configured to perform the *Sense_Task* and the other configured to perform the *Cooling_Task*, for one week of operation, the simulator takes 105 seconds on average.

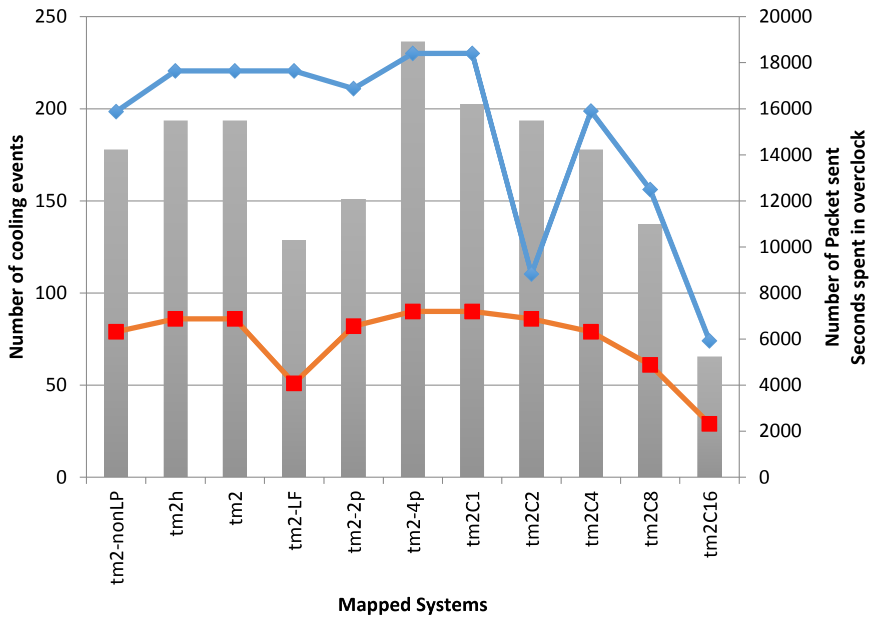


Figure 4.29: Average number of cooling and sensing events (in red and blue respectively) in a week for different configurations of the monitoring device applied on top of the Pandaboard CPU. Seconds the CPU spent in overclock (represented by bars).

Chapter 5

Conclusion

5.1 Summary

We have proposed and discussed a methodology for the design and evaluation of Cyber-Physical System. Our approach consists in extending the traditional function/architecture co-design principle to clearly distinguish the cyber from the physical world. The two are then integrated again through a double mapping step, which brings separate functional and architectural models together into an overall system implementation. We demonstrated the proposed methodology applying it to two design examples of CPSs. As case studies we have chosen two systems for which the author has actively participated in the design. The case study of the audio guidance system is subject to real–time constraints. The design of the data center monitoring and cooling system is studied because its contribution to the data center energy efficiency is subject to the power availability, and because of the complex interactions that exists between components. In both cases, the METROII framework allows engineers to express the connections between the functional world as formalized by control algorithms as well as the events of the physical world to capture both aspects of the design. The results obtained from the evaluation agree with the implementations, and allowed us to choose the appropriate platform for further experimentation. The same approach can be adopted to express other types of interaction that can happen in a CPS, and to explore various design solution independently from the level of abstraction used for the system description. One advantages of the implemented methodology is the separation of concerns inherited from METROII. This aspect makes easier the design exploration because the states of the components are separated and their interactions results from mapping process automatically.

5.2 Future work

Presented work can be extended to model other systems by including a variety of components and to support more parameters for architectural design exploration. Moreover, the feature that comes by default from METROII are still supported. Future work is mainly devoted to modeling and mapping other CPSs to further extend the methodology to support design elements that have not been considered in the presented case studies. In fact, by modeling different systems makes it possible to spot trends that enable the methodology to be formally defined for entire groups of systems.

Even for the presented cases, it would be possible to increase the level of detail of the models to analyze aspects of the design that have not been considered in current models. In the case of the binaural guidance system, the presented model contains a feature that have been not used yet. The person architectural component, in fact, can be used to perform simulations with real data we collected during the experimental evaluation. The spatial coordinates of the planned path, and the actual coordinates the user have traveled can be read from the planner and the person architectural components. In this way it is possible to analyze the relation between the application load and the type of trajectory (straight or curved path) according to the behavior of the user that of course depends on the degree at which he/she interprets the interfaces.

Considering the design of the TERS, we mainly focused our attention to energy considerations. The presented TERS model does not account for the issue related to the network. This reflects the current implementation of the smart sink since whenever there is enough energy to operate, the node transmits its data. Thus, our analysis allows us to determine the maximum number of transmission, but does not consider how many network packets are actually received. In a realistic deployment, a network protocol that handles packet collision using clear channel assessment and a retransmission policy must be included. It can be possible to model the network by inserting an architectural component that represents the channel into the design. The channel architectural component contains a provided port for each MSP component that support the execution of the Sense_Task. The provided port allows a component to occupy the channel for the amount of time required to perform the data transmission. If a component requires to transmit data while the channel is already occupied it stops and waits for the channel to be free in order to transmit. Obviously, in this case the power consumed from the node must be modeled and will be higher with respect to the transmission without collisions. This will allows us to obtain a more realistic simulation of the smart heat sink operations.

Bibliography

[1] V.R. Algazi, R.O. Duda, D.M. Thompson, and C. Avendano. The CIPIC HRTF database. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 99–102, 2001.

[2] Arndale Board [Technical Reference]. http://www.arndaleboard.org/wiki/-index.php/WiKi. 2013.

[3] Giuseppe Ascia, Vincenzo Catania, and Maurizio Palesi. A GA-based design space exploration framework for parameterized system-on-a-chip platforms. *Evolutionary Computation, IEEE Transactions on*, 8(4):329–346, 2004.

[4] Assistants for safe mobility (assam). `http://assam.nmshost.de/`.

[5] Durand R Begault. Perceptual effects of synthetic reverberation on three-dimensional audio systems. *Journal of the Audio Engineering Society*, 40(11):895–904, 1992.

[6] Durand R Begault, Elizabeth M Wenzel, and Mark R Anderson. Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. *Journal of the Audio Engineering Society*, 49(10):904–916, 2001.

[7] Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, Christoph Clauß, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, Manuel Monteiro, Thomas Neidhold, et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference*, pages 105–114. Linköping University Press, 2011.

[8] Borzoo Bonakdarpour, Marius Bozga, Mohamad Jaber, Jean Quilbeuf, and Joseph Sifakis. A framework for automated distributed implementation of component-based models. *Distributed Computing*, 25(5):383–409, 2012.

[9] Jeffrey Borish. Extension of the image model to arbitrary polyhedra. In *Journal of Acoustic Society of America*, volume 75, 1984.

[10] C.P. Brown and R.O. Duda. A structural model for binaural sound synthesis. *Speech and Audio Processing, IEEE Transactions on*, 6(5):476–488, Sep 1998.

[11] Davide Brunelli, Roberto Passerone, Luca Rizzon, Maurizio Rossi, and Davide Sartori. Self-powered WSN for distributed data center monitoring. *Sensors*, 16(1):57, 2016.

[12] Joseph T Buck, Soonhoi Ha, Edward A Lee, and David G Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *International Journal of Computer Simulation*, 4:155–182, August 1994.

[13] Corey I Cheng and Gregory H Wakefield. Moving sound source synthesis for binaural electroacoustic music using interpolated head-related transfer functions (hrtfs). *Computer Music Journal*, 25(4):57–80, 2001.

[14] Alessio Colombo, Daniele Fontanelli, Axel Legay, Luigi Palopoli, and Sean Sedwards. Motion Planning in Crowds using Statistical Model Checking to Enhance the Social Force Model. In *Proc. IEEE Int. Conf. on Decision and Control*, pages 3602–3608, Florence, Italy, 10-13 Dec. 2013. IEEE.

[15] Abhijit Davare, Douglas Densmore, Liangpeng Guo, Roberto Passerone, Alberto L. Sangiovanni-Vincentelli, Alena Simalatsar, and Qi Zhu. METROII: A design environment for cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 12(1s):49:1–49:31, March 2013.

[16] Institut de Recherche et Coordination Acoustique/Musique. Listen hrtf database. URL: http://recherche.ircam.fr/equipes/salles/listen/index.html.

[17] Dell Inc. Openmanage power center temperature monitoring. [Online] `http://www.dell.com/support/article/us/en/19/SLN283321/EN`.

[18] Douglas Densmore, Roberto Passerone, and Alberto L. Sangiovanni-Vincentelli. A platform-based taxonomy for ESL design. *IEEE Design and Test of Computers*, 23(5):359–374, May 2006.

[19] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber–physical systems. *Proceedings of the IEEE*, 100(1):13–28, 2012.

[20] The e-no-falls project. `http://www.e-nofalls.eu`.

[21] Excelitas Tech. Photoconductive cell, vt900 series. [Online] `http://www.farnell.com/datasheets/612931.pdf`.

[22] Fabio P Freeland, Luiz WP Biscainho, and Paulo SR Diniz. Efficient hrtf interpolation in 3d moving sound. In *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio.* Audio Engineering Society, 2002.

[23] Liangpeng Guo, Qi Zhu, Pierluigi Nuzzo, Roberto Passerone, Alberto L. Sangiovanni-Vincentelli, and Edward A. Lee. Metronomy: a function-architecture co-simulation framework for timing verification of cyber-physical systems. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, CODES14, pages 24:1–24:10, New Delhi, India, October 12–17, 2014.

[24] Dan Henriksson and Hilding Elmqvist. Cyber-physical systems modeling and simulation with modelica. In *International Modelica Conference, Modelica Association*, volume 9, 2011.

[25] Texas Instruments. Cc2500 low-cost low-power 2.4 ghz rf transceiver. [Online] `http://www.ti.com.cn/cn/lit/ds/swrs040c/swrs040c.pdf`.

[26] Texas Instruments. Mixed signal microcontroller. [Online] `http://www.ti.com.cn/cn/lit/ds/slas504g/slas504g.pdf`.

[27] The iwalkactive project. `http://www.iwalkactive.eu/`.

[28] Sriram Jayakumar and Sherief Reda. Making sense of thermoelectrics for processor thermal management and energy harvesting. In *International Symposium on Low Power Electronics and Design (ISLPED)*, 2015.

[29] Jeff C Jensen, Danica H Chang, and Edward A Lee. A model-based design methodology for cyber-physical systems. In 7^{th} *International Wireless Communications and Mobile Computing Conference*, IWCMC, pages 1666–1671, Istanbul, Turkey, July 4–8, 2011. IEEE.

[30] Prachi Joshi, Sandeep K. Shukla, Jean Pierre Talpin Inria, and Huafeng Yu. Mapping functional behavior onto architectural model in a model driven embedded system design. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC15, pages 1624–1630, Salamanca, Spain, April 13–17, 2015. ACM.

[31] Jean-Marc Jot. Efficient models for reverberation and distance rendering in computer music and virtual audio reality. In *Proc. 1997 International Computer Music Conference.* Citeseer, 1997.

[32] Hokeun Kim, Liangpeng Guo, E.A. Lee, and A. Sangiovanni Vincentelli. A tool integration approach for architectural exploration of aircraft electric power systems. In

IEEE 1st International Conference on Cyber-physical systems, Networks, and Applications, CPSNA13, Taipei, Taiwan, August 19–20, 2013.

[33] Anja Kollmuss and Julian Agyeman. Mind the gap: why do people act environmentally and what are the barriers to pro-environmental behavior? *Environmental education research*, 8(3):239–260, 2002.

[34] R.Hari Krishnan and S. Pugazhenthi. Mobility assistive devices and self-transfer robotic systems for elderly, a review. *Intelligent Service Robotics*, 7(1):37–49, 2014.

[35] Laird Technologies, Inc. Thermoelectric handbook. www.lairdtech.com/temhandbook/.

[36] Edward A Lee. Cyber-physical systems-are computing foundations adequate. In *Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, volume 2, 2006.

[37] Geunho Lee, Takanori Ohnuma, and NakYoung Chong. Design and control of JAIST active robotic walker. *Intelligent Service Robotics*, 3(3):125–135, 2010.

[38] Soochan Lee, Patrick E Phelan, and Carole-Jean Wu. Hot spot cooling and harvesting CPU waste heat using thermoelectric modules. In *Proc. of ASME 2014 Int. Mechanical Engineering Congress and Exposition*, Montreal, Canada, November 14–20, 2014. American Society of Mechanical Engineers.

[39] Maurizio Rossi, Luca Rizzon, Matteo Fait, Roberto Passerone, Davide Brunelli. Energy neutral wireless sensing for server farms monitoring. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(3):324–335, September 2014.

[40] Micropelt. *TE-CORE6 — TE-CORE7 ThermoHarvesting Power Module.* [Datasheet], October 2012.

[41] Microsoft Corporation. Data center genome project. [Online] `http://research.microsoft.com/en-us/projects/dcgenome/`.

[42] James A Moorer. About this reverberation business. *Computer music journal*, pages 13–28, 1979.

[43] Federico Moro, Daniele Fontanelli, Roberto Passerone, Domenico Prattichizzo, Luca Rizzon, Stefano Scheggi, Stefano Targher, Antonella De Angeli, and Luigi Palopoli. Follow, listen, feel and go: alternative guidance systems for a walking assistance device. *arXiv preprint arXiv:1601.03915*, 2016.

[44] Sirajum Munir, John A. Stankovic, Chieh-Jan Mike Liang, and Shan Lin. Cyber physical system challenges for human-in-the-loop control. In *8th International workshop on feedback computing. San Jose, CA, June 25*, 2013.

[45] Nextreme Thermal Solutions. *Thermomobility Wireless Power Generator (WPG-1).* [Datasheet].

[46] A. Noyer, P. Iyenghar, E. Pulvermueller, J. Engelhardt, F. Pramme, and G. Bikker. A model-based workflow from specification until validation of timing requirements in embedded software systems. In *Proceedings of the* 10^{th} *IEEE International Symposium on Industrial Embedded Systems*, SIES15, pages 1–4, Siegen, Germany, June 8–10, 2015.

[47] Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Davide Bresolin, Luca Geretti, and Tiziano Villa. A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proceedings of the IEEE*, 103(11):2104–2132, 2015.

[48] Oncque. Dip - rbs07 series. [Online] `http://www.oncque.com/05-tilt_switch.html?CID=1`.

[49] Luigi Palopoli, Antonis Argyros, Josef Birchbauer, Alessio Colombo, Daniele Fontanelli, Axel Legay, Andrea Garulli, Antonello Giannitrapani, David Macii, Federico Moro, Payam Nazemzadeh, Pashalis Padeleris, Roberto Passerone, Georg Poier, Domenico Prattichizzo, Tizar Rizano, Luca Rizzon, Stefano Scheggi, and Sean Sedwards. Navigation assistance and guidance of older adults across complex public spaces: the dali approach. *Intelligent Service Robotics*, 8(2):77–92, 2015.

[50] Luigi Palopoli et al. Navigation assistance and guidance of older adults across complex public spaces: the DALi approach. *Intelligent Service Robotics*, pages 1–16, 2015.

[51] pandaboard.org. *OMAP4 PandaBoard System Reference Manual.* [Datasheet], Rev. 0.4, September 2010.

[52] Alessandro Pinto, Alvise Bonivento, Alberto L. Sangiovanni-Vincentelli, Roberto Passerone, and Marco Sgroi. System level design paradigms: Platform-based design and communication synthesis. *ACM Transactions on Design Automation of Electronic Systems*, 11(3):537–563, July 2006.

[53] Claudius Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II.* Ptolemy.org, 2014.

[54] Qi Zhu. *Optimizing mapping in system level design.* PhD thesis, University of California, Berkeley, 2008.

[55] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the* 47th *Design Automation Conference*, pages 731–736. ACM, 2010.

[56] Rick Merritt. Dell, IBM give thumbs up to ARM Servers. In *EE Times*, 2010.

[57] Luca Rizzon, Federico Moro, Roberto Passerone, David Macii, Daniele Fontanelli, Payam Nazemzadeh, Michele Corrà, Luigi Palopoli, and Domenico Prattichizzo. c-walker: a cyber-physical system for ambient assisted living. In Alessandro De Gloria, editor, *Applications in Electronics Pervading Industry, Environment and Society*, volume 351, page 75. Springer International Publishing, 2015.

[58] Luca Rizzon and Roberto Passerone. Embedded soundscape rendering for the visually impaired. In *Proceedings of the* 8th *IEEE International Symposium on Industrial Embedded Systems*, SIES13, pages 101–104, Porto, Portugal, June 19–21, 2013.

[59] Luca Rizzon and Roberto Passerone. Spatial sound rendering for assisted living on an embedded platform. In *Applications in Electronics Pervading Industry, Environment and Society*, pages 61–73. Springer International Publishing, 2014.

[60] Luca Rizzon, Maurizio Rossi, Roberto Passerone, and Davide Brunelli. Wireless sensor networks for environmental monitoring powered by microprocessors heat dissipation. In *Proceedings of the* 1st *International Workshop on Energy Neutral Sensing Systems*, ENSSys13, pages 8:1–8:6, Rome, Italy, November 14, 2013.

[61] Luca Rizzon, Maurizio Rossi, Roberto Passerone, and Davide Brunelli. Energy neutral hybrid cooling system for high performance processors. In *Proceedings of the* 5th *International Green Computing Conference*, IGCC14, pages 1–6, Dallas, TX, November 3–5, 2014.

[62] Luca Rizzon, Maurizio Rossi, Roberto Passerone, and Davide Brunelli. Self-powered heat-sink SoC as temperature sensors with wireless interface: Design and validation. In *Proceedings of IEEE SENSORS 2014*, pages 1575–1578, Valencia, Spain, November 2–5, 2014.

[63] Davide Rocchesso. Fractionally addressed delay lines. *Speech and Audio Processing, IEEE Transactions on*, 8(6):717–727, 2000.

[64] Maurizio Rossi, Luca Rizzon, Matteo Fait, Roberto Passerone, and Davide Brunelli. Energy neutral wireless sensing for server farms monitoring. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(3):324–334, September 2014.

[65] Vassiliki Sfyrla, Georgios Tsiligiannis, Iris Safaka, Marius Bozga, and Joseph Sifakis. Compositional translation of Simulink models into synchronous BIP. In *Proceedings of the* 5th *IEEE International Symposium on Industrial Embedded Systems*, SIES10, pages 217–220, Trento, Italy, July 9–11 2010. IEEE.

[66] Dmitry Sinyukov, Ross Desmond, Matthew Dickerman, James Fleming, Jerome Schaufeld, and Taskin Padir. Multi-modal control framework for a semi-autonomous wheelchair using modular sensor designs. *Intelligent Service Robotics*, 7(3):145–155, 2014.

[67] Vishay. Ntcle100e3 datasheet. [Online] `http://www.vishay.com/docs/29049/ntcle100.pdf`.

[68] R.J.M. Vullers, R. van Schaijk, I. Doms, C. Van Hoof, and R. Mertens. Micropower energy harvesting. *Solid-State Electronics*, 53(7):684 – 693, 2009.

[69] Darren ER Warburton, Crystal Whitney Nicol, and Shannon SD Bredin. Health benefits of physical activity: the evidence. *Canadian medical association journal*, 174(6):801–809, 2006.

[70] H. Wierstorf, M. Geier, A. Raake, and S. Spors. A free database of head-related impulse response measurements in the horizontal plane with multiple distances. *Audio Engineer Society 130th Convention*, 2011.

[71] Yang Yang, Alessandro Pinto, Alberto Sangiovanni-Vincentelli, and Qi Zhu. A design flow for building automation and control systems. In *Proceedings of the* 31st *IEEE Real-Time Systems Symposium*, RTSS10, pages 105–116, San Diego, CA, US, Nov. 30 – Dec. 3, 2010.

[72] Zewde Yeraswork. AMD Launches First ARM-based Server CPU. In *EE Times*, 2014.

[73] Jia Zou, Slobodan Matic, Edward A Lee, Thomas Huining Feng, and Patricia Derler. Execution strategies for PTIDES, a programming model for distributed embedded systems. In 15th *IEEE Real-Time and Embedded Technology and Applications Symposium*, RTAS09, pages 77–86, San Francisco, CA, United States, April 13–16, 2009.