Nuno Miguel Cerqueira da Costa

Licenciatura em Ciências de Engenharia Biomédica

User Friendly Knowledge Acquisition System For Medical Devices Actuation

Dissertação para obtenção do Grau de Mestre em
Engenharia Biomédica

Orientador: Prof. Doutor Hugo Gamboa

Júri:

Presidente: Prof. Doutor Mário António Basto Forjaz Secca

Arguentes: Prof. Doutor Nuno Manuel Garcia dos Santos

Vogais: Prof. Doutor Hugo Filipe Silveira Gamboa

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Outubro, 2012

User Friendly Knowledge Acquisition System For Medical Devices Actuation

To the ones that love me,
the ones that tried to help.
To life, the greatest adventure.

Acknowledgements

Great was the effort to breed a solution, because the objective was not always well defined, and to narrow the possibilities and create a good path many questions needed answers. As a small token of my appreciation, below, I acknowledge the helpers, the doers and the goods that entered in my life living their mark.

First, I need to thank professor Hugo Gamboa for providing this opportunity, and when he was challenged by me to create a non specific software with the most potential possible he helped me with ideas that bare fruits in the end, for that I'm grateful. I'm also grateful for the chance to work in Plux - Wireless Biosignals, S.A., which gave me different tools regarding the business environment.

Without maturation through the Biomedical Engineering course held in the *Faculdade de Ciencias e Tecnologias* from *Universidade Nova de Lisboa* it would be impossible to achieve the objectives proposed in this dissertation. Therefore, I must thank the institution and all the professors, friends and persons that aid me through this journey.

Furthermore, a special thanks to Neuza Nunes for the fast response to all my questions and the undoubtedly help to perform a better work. A great thanks to Tiago Araujo due to its availability in helping me in this project, Joana Sousa for caring and the other staff from Plux for conveying a pleasant environment. Additionally, I must make a personal thanks to Daniel Rodrigues for being a good friend and providing me a hand whenever I needed, and Suse Vasconselos for giving me motivation and sunny days. To my parents, it's my duty to thank them for given me the pleasure of enjoy life, the greatest adventure.

Finally, I state that in order to achieve a better understanding of this beautiful world, full with countless of amazing and mysterious things, the course of Biomedical Engineering was perfect to construct some fundamental foundations that will open profitable paths to attain my goal. This being said, I'm pleased to present this final work, that gather knowledge acquired in these six months of work and throughout the course. On top of that, I deeply hope that this work will have a wider contribution in Biomedical and other research areas, because like Carl Sagan referred *the history of science teach us that the maximum that we hope for are consecutive improvements in our understanding, thus the maximum*

that each generation hope to achieve is to slightly reduce the error-bars and enlarge the number of data related to the fields where this error-bars are applied.

Abstract

Internet provides a new environment to develop a variety of applications. Hence, large amounts of data, increasing every day, are stored and transferred through the internet. These data are normally weakly structured making information disperse, uncorrelated, non-transparent and difficult to access and share. Semantic Web, proposed by the World Wide Web Consortium (W3C), addresses this problem by promoting semantic structured data, like ontologies, enabling machines to perform more work involved in finding, combining, and acting upon information on the Web.

Pursuing this vision, a Knowledge Acquisition System (KAS) was created, written in JavaScript using JavaScript Object Notation (JSON) as the data structure and JSON Schema to define that structure. It grants new ways to acquire and store knowledge semantically structured and human readable. Plus, structuring data with a Schema generates a software robust and error – free.

A novel Human Computer Interaction (HCI) framework was constructed employing this KAS, allowing the end user to configure and control medical devices. To demonstrate the potential of this tool, we present the configuration and control of an electrostimulator.

Nowadays, most of the software for Electrostimulation is made with specific purposes, and in some cases they have complicated user interfaces and large, bulky designs that deter usability and acceptability. The HCI concedes the opportunity to configure and control an electrostimulator that surpasses the specific use of several electrostimulator software. In the configuration the user is able to compile different types of electrical impulses (modes) in a temporal session, automating the control, making it simple and user-friendly.

Keywords: Human Computer Interaction, Knowledge Acquisition System, Ontology, Schema Language, JSON, Electrostimulation.

Resumo

A Internet faculta um novo ambiente para a concepção de aplicações com várias finalidades. Desta forma, grandes quantidades de dados são armazenados e transmitidos através da internet, aumentando dia após dia. Usualmente, estes dados são mal estruturados tornando a informação dispersa, sem correlação, pouco transparente, de difícil acesso e difícil de partilhar. A Semantic Web, proposta pelo World Wide Web Consortium (W3C), resolve este problema promovendo dados semanticamente estruturados, tal como as ontologias, possibilitando que as máquinas realizem um maior volume de trabalho no contexto da pesquisa, combinação, e atuação sobre informação na Web.

Prosseguindo esta visão, um Sistema de Aquisição de Conhecimento (KAS) foi criado, desenvolvido em JavaScript usando JavaScript Object Notation (JSON) como a estrutura de dados e JSON Schema para definir essa estrutura. Este, proporciona novas formas de adquirir e armazenar conhecimento semanticamente estruturado, legível aos seres humanos. Adicionalmente, com os dados estruturados por um Schema o software torna-se robusto e menos susceptivel a erros.

Uma nova Interação Humano Computador (HCI) foi desenvolvida baseada neste KAS, permitindo ao utilizador final configurar dispositivos médicos e controlá-los, como por exemplo um electroestimulador.

Atualmente, a maior parte dos softwares para Electroestimulação são produzidos com fins específicos. Em alguns casos, estes têm interfaces complicadas, designs largos e com muita informação afetando a usabilidade e aceitabilidade. A HCI desenvolvida permite a configuração e controlo de um electroestimulador, superando o uso específico de vários softwares de Electroestimulação. Na configuração, o usuário tem a possibilidade de compilar diferentes tipos de impulsos elétricos (modos) numa sessão temporal, automatizando o controlo e tornando-o simples e intuitivo.

Palavras-chave: Interacção Humano Computador, Sistema de Aquisição de Conhecimetno, Ontologia, Schema Language, JSON, Elecroestimulação.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

EEG Electroencephalography

EMG Electromyography

ECG Electrocardiography

ES Electrical Stimulation or Electrostimulation

EMS Electrical Muscle Stimulation or Electromyostimulation

NMES Neuromuscular Electrical Stimulation

FES Functional Electrical Stimulation

TES Transcranial Electrical Stimulation

$\mathbf{V_m}$ Transmenbrane Voltage

WWW World Wide Web

HCI Human-Computer Interaction

IR Information Retrieval

KDD Knowledge Discovery in Databases and Data Mining

XML Xtensible Markup Language

JSON JavaScript Object Notation

API Application Programming Interface

GUI Graphical User Interfaces

KAS Knowledge Acquisition System

SR Symbolic Regression

GP Genetic Programming

GE Grammatical Evolution

1

Introduction

1.1 Motivation

Internet advent has strongly influenced modern electric devices. Web based conventional, expert [1] and intelligent systems [2] with knowledge and database connectivity provide novel architectures and solutions for on-line and off-line applications. Also, it allows multi-access communications between different users and mobile or nomadic computing (the use of portable computing devices in conjunction with mobile communications technologies to enable users to access Internet and data on their home or work computers from anywhere in the world) [3, 4]. Another favourable possibility is the implement of feedback control systems, also named closed loop systems [5], very important for biofeedback control [6]. For these reasons, Biomedical Web-based solutions, in particular biosignals solutions, start to make their début in health management striving to achieve preventive medicine [7, 8, 9].

The easiness of access, storage, transmission of data and the exponential proliferation of Internet users enclose new complexities: authentication; scalable configuration management; security; huge masses of high dimensional and often weakly structured data. Structuring data pursued by Semantic Web opens new opportunities, because there is always room for improving and to develop more adequate languages. Methods and approaches to solve the problem emerged from research in Human-Computer Interaction (HCI) [10], Information Retrieval (IR), Knowledge Discovery in Databases and Data Mining (KDD) [11]. These methods assist end users to identify, extract, visualize and understand useful information from data. One of the methods to solve the problem is the establishment of ontologies. It is possible thanks to Semantic Web and holds great promise in manipulating information in ways that are useful and meaningful to humans.

Mechanisms to specify ontologies expanded in the last years, the Schema Language is one of them [12, 13].

Electric flow systems have a vital role in the organism. In order to preserve the integrity of this system, and the organism, most of the health diagnosis have progressed in this path: Electromyography (EMG), Electroencephalography (EEG), Electrocardiography (ECG), between others. In favour of infuse and receive information from the bioelectric system for treatment, control, tests/analysis, it was developed the notion of Electrical Stimulation or Electrostimulation (ES) in parallel with acquisition and signal processing. Therefore, ES is affiliated to Biosignals and it's theoretically supported by Bioelectromagnetism [14].

In 1791, scientist Luigi Galvani showed that electricity, when applied to a frog's leg, could cause muscle twitches [15]. In the intervening centuries, scientists have learned much about how ES affects muscle tissue and tried to apply it to muscles paralysed by neuromuscular disease to create both therapeutic and functional effect. This field of medicine is known variously as Electrical Muscle Stimulation or Electromyostimulation (EMS), Neuromuscular Electrical Stimulation (NMES) and Functional Electrical Stimulation (FES) electromyostimulation, consisting in nerve manipulation through electrical pulses aiming muscular contraction or sensory response on distinct applications [16, 17]. Nowadays, the applications can be divided in two areas:

- **Electrotherapy:** Rehabilitation [18]; Spinal cord injury, stroke, sensory deficits, and neurological disorders (with Neural prostheses) [19]; Urinary incontinence [20]; Transcranial Electrical Stimulation (TES) as a method to elicit electronarcosis, electrosleep and electroanalgesia (for pain relief) [21, 22]; Treatment of lower limbs venous insufficiency related symptoms in pregnant women [23]; electrostimulation of the acoustic nerve for profoundly deaf patients can, in the best cases, reach almost complete speech understanding without lip reading [24].

- **Physical conditioning:** fitness; active recovery; optimizing physical performance by improvement of maximum strength of a muscle (muscular tonus) in less time [25, 17, 26] .

Although ES may hold much promise there are technical challenges still unsurpassed. Commercial software solutions for electrostimulators grow every day, but these are often limited by a variety of factors including cost, source code inaccessibility, hardware compatibility, and more. Consequently, a strong tradition in scientific research is to write custom software routines. While superb for the specific tasks at hand, these custom solutions rarely offer the flexibility and extensibility needed for them to be transferable across platforms, hardware configurations, and experimental paradigms without significant modifications. Therefore, in present times software/hardware solutions are required to provide a device with a multi-purpose platform (sport, therapy or investigation) and a dynamic WEB based software enabling the user to create their own protocols [27, 28, 29].

In order to achieve this goal a Knowledge Acquisition System (KAS) and HCI were created, supplying the tools to configure and control biomedical and other electrical devices, for example an electrostimulator.

The Human Computer Interaction framework was constructed using the KAS. The last was written in JavaScript using JavaScript Object Notation (JSON) as the data structure and JSON Schema to define that structure, implementing a mechanism to define ontologies and use them to acquire knowledge from the user then store it semantically structured.

It was with great motivation that this overarching tool was developed at PLUX - Wireless Biosignals, S.A.. The dissertation development at Plux also permitted the first contact with a business environment and the access to the needed material and knowledge resources, making possible to conclude the objectives purposed by myself and the members of Plux.

1.2 Objectives

The prime purpose of this work was to develop a Web-based Human Computer Interaction framework for precise, dynamic and user-friendly control of medical devices manipulated trough electric impulses, like an electrostimulator. With this intention, through the development of this work was clear that a higher level of abstraction would be required to construct a multi-purpose software, assuring a wider contribution on Biomedical research field. From this level was envisioned a Knowledge Acquisition System based on JSON and JSON Schema enabling the creation of any type of forms with specialized Graphical User Interfaces (GUI) to acquire information from the end user and store it semantically structured. With the help of this mechanism was feasible the configuration of electrical impulses and sequentially compile them, supporting the construction of a Human Computer Interaction framework that automates the control of medical devices. It can also support the control of other electrical devices, like for example LEDs.

These central objectives aimed to collect the best in the fields of Human-Computer Interaction (HCI), Information Retrieval (IR), Knowledge Discovery in Databases and Data Mining (KDD), Knowledge Acquisition System (KAS), Electrical Stimulation or Electrostimulation (ES), and other areas, to transform it in something new. As a result, exceed the concept of specific software referred in Section 1.1 and create an application with a simple, human readable data structure supplying the user with easy tools to design all type of protocols, where the only limitation for creativity is the hardware.

For validation, different types of tests were performed with the help of an electrostimulator. As a consequence, was possible to detect bugs and optimize parts of the application. This tool is incorporated in a software for acquisition and processing of Biosignals by PLUX and it can be used for both stand-alone (off-line, Opensignals server from Plux) and Web-based (on-line) applications.

1.3 Dissertation Overview

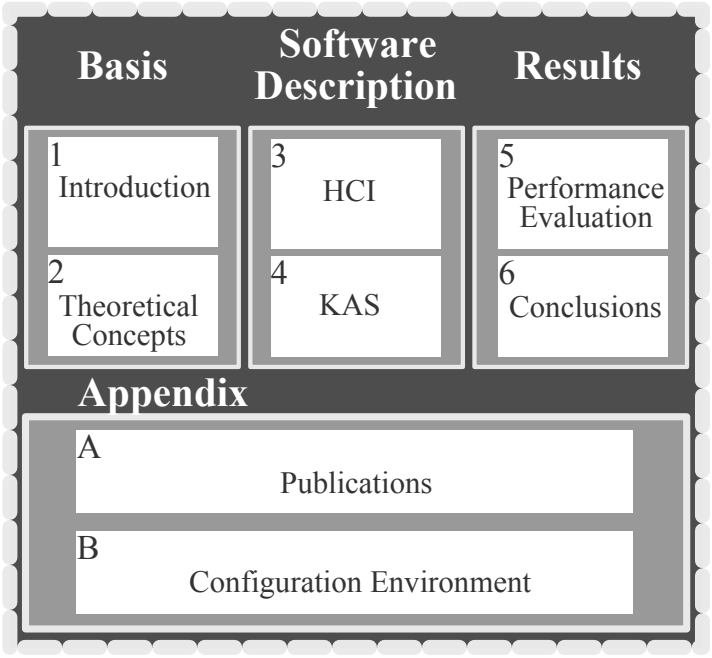

Figure 1.1: Dissertation schematic.

This section provides an overview of the aspects delineated in the context of this dissertation. The dissertation is divided into four parts composed with five chapters and two appendix, as the schematic in Figure 1.1.

In the present chapter, the thesis subjects are exposed, providing some insight on the state of the art, depicted within the section Motivation, and the Objectives which led to the development of this work.

The second chapter describes the concepts required throughout the thesis: the importance and issues related to Web based software; HCI and KAS definitions; Symbolic Regression Programming, a process where data are fitted by suitable mathematical formula; Ontologies, classes and subclasses of information with relations and inference rules between them; Schemas, JSON and JSON Schema to construct data semantically structured; resume of the languages used in the software, JavaScript and Python and finally the theoretical basis, bioelectricity, concerning ES and the link with biosignals. These first two chapters form the basis for the development of the thesis.

A novel Human Computer Interaction was developed with the intention of controlling the actuation of electrical devices, like Biosignals Acquisition Systems, LEDs, motors, between others. The components from this HCI are depicted in the third chapter.

The Knowledge Acquisition System is described in the fourth chapter. A versatile system to architect simple and intuitive interfaces with specialized GUI to acquire knowledge from the end user, and store it semantically structured.

To show the potential of this software and test its performance, the fifth chapter compiles a series of evaluations and examinations with the intention of control an electrostimulator.

Some final remarks and future work are presented in Chapter 6. Appendix A contains the papers published in the context of this research work, and Appendix B contains the image of the configuration environment.

2

Concepts

This chapter encloses the theoretical base for this dissertation, familiarizing the concepts required throughout the project to achieve the purposes introduced in Section 1.2.

Web-based control systems and its connection with Human-Computer Interaction (HCI) will be introduced to understand the motivation in design a Web-based framework. The terms HCI and Knowledge Acquisition System (KAS) are specified because they play an important role in this dissertation. Ontology, Schema language, JavaScript Object Notation (JSON) and Symbolic Regression Programming are also described and their interlock allowed the software development. Plus, the language platforms used to built the software, JavaScript and Python, are both resumed.

In addition, an overview in bioelectricity, especially in Electrical Stimulation or Electrostimulation (ES), introduces the concepts required for software validation. ES is affiliated with biosignals, in particular bioelectricity, as a method employed by humans to interfere with the bioelectrical system in profitable ways. Without biofeedback, one of the main branch of biosignals and the scientific method for analysis [30], this project wouldn't be whole, so an overview in biosignals acquisition and processing will be carried.

2.1 WEB Based Software

During a short period of time occurred an explosion of technology and information, providing effective means to achieve higher standards of medical education, patient treatment and biomedical research. As a consequence, in the present, these technologies are used for: collaborative medicine, nationally and globally oriented electronic health care

between patients and health professionals around the world, and collaborative biomedical research and education possibilities using distributed system capabilities over high speed networks and new multimedia video conferencing technologies. Through these means, create "Centers without Walls" enabling Biomedical researchers to do their work without being bound by geographic location.

As the most powerful multimedia form of communication, Internet has already proved significant economic and scientific potential with unmatched educational and social benefits. The packet-switched digital transmission services embodied in the Internet are independent of content and they evolved to support voice, conventional television, video imaging, formatted text, the World Wide Web (WWW) and other forms of communication and control [8]. In a definitive article on the subject, Conklin [31] suggests the following four broad applications for computer based hypertext systems, and potentially the WWW, which have great relevance to medical practice and medical informatics [7]:

- **macro literary systems**: systems and technology that support large on-line libraries with computer mediated inter-document links (for example, network oriented publishing, reading, criticism, and collaboration in document creation).
- **problem exploration systems**: tools to support early unstructured thinking on a given problem (for example, early authoring, outlining, problem solving, programming, and design).
- **structured browsing systems**: small-scale teaching, reference, and public information systems similar in design and function to macro literary systems. In these systems, ease of use is a critical design component.
- **general hypertext systems**: general-purpose systems for reading, writing, collaboration, etc., designed to allow experimentation with a range of hypertext applications.

In the next chapters the reader will notice that the software developed in this work makes part of these applications.

To solve biomedicine problems, generated by using the Internet, we need to focus in the central Internet problems. The easiness of access, storage, share of data and the exponential proliferation of Internet users enclose new complexities: authentication; scalable configuration management; security; huge masses of high dimensional and often weakly structured data between others. Weakly structured data is the main problem addressed in this project. Structuring data pursued by Semantic Web (promotes common formats for data) creates opportunities to develop new languages and adapt, improve old ones. Methods and approaches to solve the structuring problem emerged from research in HCI [10], IR, KDD [11]. These methods assist end users to identify, extract, visualize and understand useful information from data.

Other issues addressed are usability and acceptability of applications. Ease of use affects the users performance and their satisfaction, while acceptability affects whether

the product is used or not [32, 33]. Therefore every application should pursue these objectives.

Resuming, Internet applications, like WEB-Based control systems, have significant impact on biomedicine by dramatically improving the ease with which the distribution and access to information is provided. WWW is a unparalleled resource due to its ease of use, its platform independent client-server software, the wide availability of inexpensive WWW browser applications, and its support of distributed hypertext and multimedia. Another important part is security, use of WWW for financial transactions has resulted in the development of a number of technologies designed to make Internet-based communication secure. These encryption-based technologies facilitate the creation of secure wide-area access to clinical information systems and remote control of devices via the Internet [34].

2.2 HCI and KAS

Human-Computer Interaction (HCI) enclosures the study, planning, and design of interactions between people (users) and computers. It's often described as the intersection of computer science, behavioural sciences, design and several other fields of study [35].

Knowledge Acquisition System (KAS) attempts to acquire fundamental knowledge from humans (users) and store it in memory to reuse in different engineered applications, like artificial intelligence, between others [36].

2.3 Symbolic Regression Programming

The term Symbolic Regression (SR) represents a process where data are fitted by suitable mathematical formula (Symbols). This process is known amongst mathematician and used when some data of unknown processes are obtained. It's also used in programming.

Today there are two well known methods, which can be used for SR in programming. The first one is called Genetic Programming (GP) and the second one is Grammatical Evolution (GE). Genetic Programming is a symbolic regression performed with evolutionary algorithms instead of human beings. The ability to solve very difficult problems was demonstrated several times, now it can even be applied, e.g. to synthesize highly sophisticated electronic circuits. Grammatical Evolution can be described as an unfolding of GP because of some common principles, which are the same for both algorithms. One important characteristic of GE is that it can be implemented in any arbitrary computer language. In contrast to other evolutionary algorithms, GE was only used with a few search strategies and a binary representation of the populations [37].

One of the tools emerging in this area with great importance is "Eureqa", a highly praised symbolic regression program defined by Schmidt and Lipson that aid to describe the underlying mechanisms producing the data with simple functions [38].

2.4 Ontologies

Semantic Web, proposed by the World Wide Web Consortium (W3C), promotes semantic structured data, like ontologies, enabling machines to perform more work involved in finding, combining, and acting upon information on the web. Ontologies hold much promise in manipulating information in ways that are useful and meaningful to the human user.

Ontologies are collections of information with specific taxonomies and inference rules to define relations between terms. The taxonomy defines classes of objects and relations among them, and inference rules provide advanced ways of relate information by deduction. Classes, subclasses and relations amidst entities are a very powerful tool for Web use. We can express a large number of relations between entities by assigning properties to classes and allowing subclasses to inherit such properties.

The structure and semantics provided by ontologies make it easier for an entrepreneur to provide a service and can make its use completely transparent. Ontologies can enhance the functioning of the Web in many ways, like relate the information on a page to the associated knowledge structures and inference rules, thus creating robust and clean applications [12].

2.5 Schema, JSON and JSON Schema

Tools to create ontologies have spring in the late years, the Schema Language is one of them. Schema was first developed for Xtensible Markup Language (XML) as a notation for defining a set of XML trees. The cause was that XML only allows users to add arbitrary structure to their documents but says nothing about what the structures mean [39]. A useful schema notation must: allow efficient parsing; be readable to the user; allow limited transformations corresponding to the insertion of defaults; identify most of the syntactic requirements that the documents in the user domain follow; be modular and extensible to support evolving classes [40].

This language makes possible the creation of structured data and automated tools to present the data in a human readable form making easier the extraction and visualization of useful information. Therefore, in this field of structuring data, Schema largely supersede Document type Definitions (DTDs) for markup language.

One example of this procedure is Protégé, an open source Ontology Editor and Knowledge Acquisition System. Protégé is a framework written in Java and uses Swing (Java GUI widget toolkit) to develop complex user interfaces. These interfaces provide the user with tools to construct domain models and knowledge-based applications with ontologies. As a graphical tool for ontology editing and knowledge acquisition, it can adapt to enable conceptual modelling with new and evolving Semantic Web languages [13]. Protégé let us, like the Schema, create domains in a conceptual level without knowing the syntax of the language ultimately used on the Web, to construct interfaces, passing

and storing information, between others. In this way, we can concentrate on the concept types (integers, arrays,...), the relationships in the domain and the facts about them that we need to describe.

JSON is a simple, lightweight and human readable text-data structure for information exchange. The approach for information exchange is simpler than XML, by the less verbose structure of the notation. Interpreting JSON is native in some languages with the existence of several support libraries that make JSON a platform independent language [41]. JSON structure is composed of name/value pairs separated by comma, curly brackets holds objects and square brackets holds arrays. Values can be numbers, strings, booleans, arrays, objects and null. In the example below is an object containing information of an address and phone number:

Listing 2.1: Example of a JSON

```
{"address":{
     "streetAddress": "21␣2nd␣Street",
     "city":"New␣York"
 },
 "phoneNumber":
 [{
     "type":"home",
     "number":"212␣555-1234"
 }]
}
```

Considering these features, JSON was selected as the data structure of this work, and is defined by JSON Schema.

From the XML Schema concept, a set of specifications were established to create a Schema for JSON. A JSON Schema is a Media Type (standard draft of options) that specifies a JSON-based format to define the structure of JSON data, providing a contract (set of rules) required in a given application and how to interact with the contract. Accordingly, JSON Schema specifies requirements for JSON properties with the following intentions:

- Validation (data integrity);
- Documentation;
- Interaction (UI generation - forms and code);
- Hyperlink Navigation.

JSON Schema is also a JSON with a compact implementation and can be used on the client and server. Specifications are organized in two parts [42]:

- **Core Schema Specification:** primary concerned with describing a JSON structure and specifying valid elements in the structure.

- **Hyper Schema Specification:** define elements in a structure that can be interpreted as hyperlinks, in others JSON documents and elements of interaction (This allows user agents to be able to successfully navigate JSON documents based on their Schemas).

Below is an example of a JSON Schema defining the structure for the JSON example showed before:

Listing 2.2: Example of a JSON Schema

```
{"type":"object",
 "required":false,
 "properties":{
    "address": {
      "type":"object",
      "required":true,
      "properties":{
        "city": {
            "type":"string",
            "required":true
        },
        "streetAddress": {
            "type":"string",
            "required":true
        }
      }
    },
    "phoneNumber": {
        "type":"array",
        "required":false,
        "items":{
            "type":"object",
            "required":false,
            "properties":{
               "number": {
                  "type":"string",
                  "required":false
                },
                "type": {
                  "type":"string",
                  "required":false
                }
            }
}}}}
```

2.6 JavaScript and Python

JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It's most commonly used in web browsers, and, in that context, the general-purpose core is extended with objects that allow scripts to interact with the user, control the web browser, and alter the document content that appears within the web browser window. It is commonly called client-side JavaScript to emphasize that scripts are run by the client computer rather than the web server [43].

Python is an interpreted, interactive, object-oriented programming language. It provides high-level data structures such as lists and associative arrays (called dictionaries), dynamic typing and dynamic binding, modules, classes, exceptions, automatic memory management, etc.. It has a remarkably simple and elegant syntax and yet is a powerful and general purpose programming language. It was designed in 1990 by Guido van Rossum. Python is modular by nature. The kernel is very small and can be extended by importing extension modules. The Python distribution includes a diverse library of standard extensions (some written in Python, others in C or C++) for operations ranging from string manipulations and Perl-like regular expressions, to Graphical User Interfaces (GUI) generators and including web-related utilities, operating system services, debugging and profiling tools, etc [44].

JavaScript and Python can be linked trough websockets, full-duplex communications channels over a single TCP connection. This way, Python aids JavaScript in high-level programming, like storing and retrieving information from servers.

2.7 Biosignals

Signal is any action (physical, economical or social) that encodes a message or some kind of information [45]. Therefore the term biosignal encloses all kind of signals from biological beings that can be monitored and measured for extracting and understanding the underlying physiological mechanisms of a specific biological event or system [46, 47]. Biosignals can be classified based on their physiological origins, when there is an interest in distinguish the fundamental physical characteristics of a process that originated a specific signal, in order to model and analyse it correspondingly [48, 49]. In this thesis the required class of biosignals is connected to bioelectricity, presented next.

2.7.1 Bioelectricity

Bioelectricity is one of the essential human sciences, depicted as changes in electrical currents generated by the sum of electrical potential differences across a specialized tissue, organ or cell system. Since the electric field propagates through the biologic medium, most of the times this potential may be acquired at specific anatomic regions on the surface, eliminating the need to perform invasive measurements.

Neurons are cells specialized for the integration and propagation of electrical events. It is through such electrical activity that neurons communicate with each other as well as with muscles and other end organs. Therefore, an understanding of basic electrophysiology is fundamental to acknowledge the function and dysfunctions of neurons, neural systems and the brain [50]. This knowledge is wisely used to stimulate neurons and muscles with electrical impulses, producing favourable interferences in the organism.

The following section emphasizes the birth and contributions of Electrical Stimulation or Electrostimulation (ES).

2.7.1.1 Electrical Stimulation (ES)

And still we could never suppose that fortune were to be so friend to us, such as to allow us to be perhaps the first in handling, as it were, the electricity concealed in nerves, in extracting it from nerves, and, in some way, in putting it under everyone's eyes. With these statement in 1791 Luigi Galvani, Professor of the University of Bologna and member of the Accademia delle Scienze, divulged the importance of his scientific achievement. He was the first to provide evidence for the electrical nature of the enigmatic fluid involved in nerve conduction and muscle contraction [51].

With the evolving progress in the understanding of electrical phenomena in excitable membranes, particularly after the Hodgkin-Huxley fundamental studies on squid giant axon and, more recently, after the development of patch-clamp technology, scientists have tried to apply this knowledge to muscles paralysed by neuromuscular disease to create both therapeutic and functional effect.Now we have better understanding in the difficulties that this science has to offer. One of the difficulties is that electricity is only involved in the excitation of the membrane of muscle and nervous cells, while the energy used by the contraction machinery is the chemical energy accumulated in molecules containing high-energy phosphate bonds as final products of metabolic processes, making harder to perform a stimulus with a precise contraction. Other difficulty is that physiological effects of nerve stimulation essentially depend on the type of nerve stimulated and not on the type of stimulus used [15]. Furthermore, the effects from electrical forces can be beneficial, as with medical diagnostic devices or biomedical implants, or can be detrimental, as with chance exposures that we typically call electric shock (Tasers). Whether a biological effect is judged beneficial or detrimental often depends on the context. Independent in the interest, it is crucial to understand the range of probable biological reactions to electrical stimulation [52].

This subdivision of bioelectromagnetism applied to medicine is known variously as Electrical Muscle Stimulation or Electromyostimulation (EMS), Neuromuscular Electrical Stimulation (NMES), Functional Electrical Stimulation (FES), Electrical Stimulation or Electrostimulation (ES), consisting in nerve manipulation through electrical pulses aiming muscular contraction or sensory response in distinct applications. Electric energy is

generated with an electronic device outside biological tissues, designated electrostimulator. The use of electrical devices is pervasive in modern society. Nevertheless, they face some technical problems due to the fact commercial software solutions to control devices grow every day, but these are often limited by a variety of factors including cost, source code inaccessibility, hardware compatibility, and more.

Electrical stimulation can be administered to the affected muscles one of two ways: electricity can be sent across the skin via surface electrodes, or it can be applied directly to the muscle or the "motor nerve" that services that muscle via implanted electrodes. Surface electrodes offer the advantage of not requiring surgery to use but they are only able to target muscles close to the surface of the skin, and the electrodes must often be placed very precisely in order to achieve maximum effect. Implanted electrodes can be placed directly next to the affected muscle or motor nerve, allowing for much more precise and repeatable stimulation to the muscle, although they are invasive and need to be placed surgically. Since the electric stimulation of biological tissues requires the use of electrodes, any practical study should include consideration of electrodes and electrode-tissue interaction. The mechanical properties of electrodes are important, particularly with respect to implants whose lifetime is measured in years. Additionally, since the flow of electricity from the electrode (where electrons carry the charges) into the tissue (where ions carry the charges) may involve an electrochemical reaction, this area must be carefully studied as well [19].

Electric energy is applied to excitable tissue in order to activate it or even interfere with the electrical signal, and the threshold necessary for eliciting a nerve fiber action potential is 100 to 1,000 times less than the threshold for muscle fiber stimulation [53]. The term "threshold" defines the lowest level of electrical charge that generates an action potential. The nonlinear membrane properties of excitable tissue interfere with the stimulus and it can be defined as transthreshold or subthreshold stimulus. The subthreshold stimulus is insufficient to cause the transmembrane potential to reach the threshold, while the transthreshold stimuli reaches the threshold and activate the excitable tissue. However, subthreshold electric energy may also be applied for other therapeutic purposes [14].

If a muscle fiber is electrically stimulated it responds with a twitch (spasm, sudden contraction of the muscle) generated by the fiber tension (twitch force), as shown in Figure 2.1. On the other hand, if a train of stimuli is supplied whose time interval is shorter than the twitch duration, then temporal summation will occur and a larger tensile force will be developed. For a high enough stimulus frequency a smooth (rather than bumpy) tension response is observed (this is the fusion frequency), leading to a maximum (tetanus) contraction. Thus, ideal stimulation frequencies range from 12-16 Hz for upper-limb applications and 18-25 Hz for lower-limb applications. Greater muscle force generation is accomplished by either increasing the pulse duration (typically 200 μs) or stimulus amplitude to activate neurons at a greater distance from the activating electrode [54].

The magnitude and duration of the twitch response differ depending on the muscle

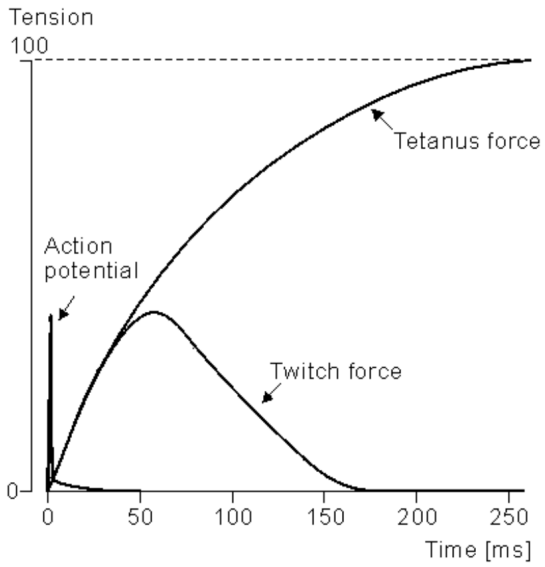

Figure 2.1: The contraction response (tension vs. time) for a single muscle fiber. The stimulus is described in the figure as "Action potential", generating a twitch. However, if the frequency of a train of stimuli is higher then the twitch frequency a smooth tension will occur, and instead of a twitch force we will have a tetanus force (maximum contraction for a longer time). [14].

fiber type. Each fiber in a bundle is innervated by a single motor neuron, but each motor neuron activates several fibers. The group of fibers activated by a single motor neuron is called a motor unit. All fibers in a motor unit are of a similar type and large diameter muscle fibers are innervated by large diameter neurons. Consequently, motor units producing the largest forces are those innervated by axons of large diameter. Conversely, small forces are produced by small diameter axons. Skeletal muscle fibers can be separated into three general groups according to their physiological and metabolic properties [54]:

- **Fast twitch, glycolytic (FG)**: These fibers depend mainly on glycolytic metabolism and less in the oxidative metabolism. When stimulated, the twitch contraction is of short duration and the response to repeated stimulation shows a rapid fatigue and slow recovery. In a mixed muscle this fiber tends to be found near the periphery.

- **Fast twitch, oxidative (FO)**: Distinct from the FG group are the fast twitch oxidative, which utilize both oxidative as well as glycolytic metabolism. The response to repeated stimulation is slower to fatigue and quicker to recover than for the FG fiber type.

- **Slow twitch, oxidative (SO)**: These fibers have the smallest cross-sectional area of the three groups. They have a low capacity for glycolytic metabolism. Their twitch response is longest in duration and lowest in magnitude (the fusion frequency is the lowest) of the three groups. Repeated stimulation causes less fatigue, and recovery is rapid. These fibers tend to lie in the central region of a muscle bundle.

The natural order of recruitment is the development of small forces from **SO** fibers followed, ultimately, by the largest forces due to recruitment of the **FG** motor units. The **FO** fibers contribute in the midrange. Thus the forces needed to maintain posture for a long period are derived from the **SO** type fiber. In other hand, nerve fiber recruitment properties elicited by ES differ from those elicited by normal physiologic means, it follows the principle of "reverse recruitment order" wherein the nerve stimulus threshold is inversely proportional to the diameter of the neuron. Thus, large-diameter nerve fibers, which innervate larger motor units, are recruited preferentially [55].

The output surface of ES devices can be constant-voltage, constant-current or a hybrid form of output. The advantage of the constant-voltage setup is that current density determines the potential for tissue damage. As the impedance of the skin increases, current decreases. However, constant-voltage stimulators have a variable motor response. Constant-current stimulators have better contraction consistency and repeatability with less variability in resistance [56]. The strength of the resultant muscular contraction, as said before, can be determined by varying the stimulus amplitude, pulse width or pulse frequency. In most applications a fused muscle contraction is desirable. To achieve this, stimulation frequencies of up to 50 Hz are recommended [57].

Several models of nerve stimulation based on principles of electrophysiology have been developed, like the model for excitation of myelinated nerve studied by McNeal (1976) [58] or simulations for a unmyelinated axon studied by Rattay (1987) [59], providing a starting point toward the elucidation of more realistic models, and some of the insights gained have wider applicability. Presently, the applications can be subdivided in:

- **Electrotherapy:** Rehabilitation; Spinal cord injury, stroke, sensory deficits, and neurological disorders(with Neural prostheses) [19]; Urinary incontinence [20]; Transcranial electrostimulation (TES) as a method to elicit electronarcosis, electrosleep and electroanalgesia or pain relief [21, 22]; Treatment of lower limbs venous insufficiency in pregnant women [23]; By means of electrostimulation of the acoustic nerve profoundly deaf patients can, in the best cases, reach almost complete speech understanding without lip reading [24].

- **Physical conditioning:** fitness; active recovery; optimizing physical performance by improvement of maximum strength of a muscle (muscular tonus) in less time [25, 17, 26].

In conclusion, despite ES may hold much promise supporting several medical fields, there are many scientific and technical challenges that need to be surpassed. Validation section presents tests of an electrostimulator being controlled by our software, and in the future stimuli protocols should be designed to perform tests in real applications.

2.7.2 Biosignals Acquisition

Biosignals acquired through specific sensors placed on the body, which convert a physical measurement into an electrical output, need to be: processed, passed from analog to digital data (through the sampling process), so then, they can be manipulated in a computer. This conversion process is called analogue-to-digital conversion (ADC). ADC comprises sampling and the quantization of continuous value at fixed intervals, rounding the continuous value to the nearest discrete unit [60]:

- **Sampling**: is the conversion of a continuous signal into a discrete time series. This process raises problems like aliasing (data distortion), and to secure a accurate discrete reconstruction the sampling must be applied (theorem mathematically expressed by Nyquist). This theorem affirm that a continuous time signal can be entirely reverted from its samples if, and only if, the sample rate is greater than twice its highest frequency component, F (the original signal bandwidth) [46, 45], i.e. $f_s > 2F$.
- **Quantization**: is the assignment for each discrete value with a discrete amplitude. The number of bits available for data storage are connected with different ADC resolutions. A quantizer with **n** bits is capable of representing a total of possible 2^n amplitude values. Normally the ADC converters use 8, 12 or 16 bits [47].

Biomedical devices to acquire biosignals have exclusive safety requirements, results from the interaction with the human body [61]. These devices are usually connected to data storage devices, using one of the several available data transmission protocols [61].

2.7.3 Biosignals Processing

After acquisition, in order to extract meaningful information and understand a particular physiologic system or event the raw, discrete biosignals should pass through four steps to produce classifier-ready data, as shown in Figure 2.2.

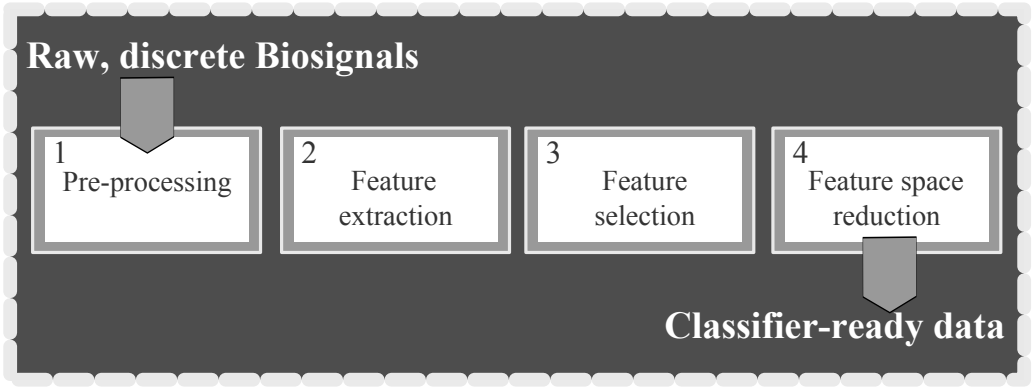

Figure 2.2: Biosignal process engine.

The pre-processing is a hand-selection of the signals that should be analysed. Then, extract the common statistical features from each type of the noise-filtered biosignals.

Followed by automatic feature selection using classification algorithms. Finally, feature space reduction for maximizing between-class scatter and minimize within-class scatter, resulting in a low-dimension representation of optimally clustered class features [62].

In short, the evolving technology provide tools for digital signal processing of mass data, making it human readable and easier to extract relevant information, with the goal of improving the discernment of physiological meaning from the original parameters [63, 47].

3

Human Computer Interaction

This HCI provides the liberty to create and sequentially compile electrical impulses in a temporal session, and with this information control via WEB the actuation of electrical devices, like Biomedical devices, Biosignals Acquisition Systems, LEDs, electrical motors, between others. To demonstrate the potential of the tool, in the validation area (Section 5.1) the software was tested with an electrostimulator showing its capabilities in controlling it and the contribution for ES.

In Biomedicine, the configuration of a device should only be performed by clinical experts, where they can adapt sessions to each patient. Overly complicated user interfaces and large, bulky designs can deter users from operate the device on a day to day basis. For example, home health care devices should have sessions programmed and, ultimately, the patient will only start or stop a session prepared for him. Thus, to economize time, simplify interfaces and separate this tool for experts and non-experts the software was divided in Configuration and Control. Configuration, where clinical experts can configure a Device with sessions for each channel. Control, where the user just need to choose the device and session previously configured, then start the session. If for some reason the session should not stop automatically (before the stop time is reached), the user can stop the session manually.

Commercial software are often limited by a variety of factors including cost, source code inaccessibility, hardware compatibility, and more. Consequently, researchers write custom software routines that are specific and most of the times can't be transferable across platforms. Also, these software usually have complicated, confuse and large interfaces that deter usability and acceptability [27, 28, 29]. The problems in question are also verified in ES. To surpass them, a dynamic, flexible way was idealized to automate the actuation of stimuli. A tool emerged from this idea, providing a user-friendly software

solution with a multi-purpose platform - different devices can be controlled automatically within a period of time, and, for example, allows the user to employ the software in different types of ES applications (sport, therapy and research).

Manufacture such tool lead to the development of a novel and innovative Knowledge Acquisition System (KAS) for the configuration part. This KAS by itself is a powerful mechanism (the most important part of the HCI) for creating ways to store information semantically structured. With the data semantically structured and with a human readable programming language, JSON (see Section 2.5), problems like data weakly structured making information disperse, uncorrelated, non-transparent, and difficult to access or share, are easily solved.

The HCI is integrated in a software for biosignals acquisition and processing from PLUX - Wireless Biosignals, S.A., enabling the control of a generic device. In this way, actuation, acquisition and processing is possible in a closed-loop cycle [64].

In this chapter all HCI pieces are described in detail and how they fit together. The next sections are divided in configuration and control.

3.1 Configuration

Configuration, the division where experts on different areas are able to design and store sessions needed for the devices they use. These sessions are sequences of electrical impulses. The fact that professionals can configure the actuation of a device, then in a different part of the software apply that configuration, establishes a frontier between configuration and control. It allows a separation between expert and non-expert, important in health care devices (home care devices, for example). Additionally, Configuration expands software usability because different devices can be configured. Its structure was designed to be simple, intuitive and user-friendly, in this manner users easily learn how to manipulate it and wont spend a lot of time configuring, enlarging the probability of software acceptance.

This division will be explained from a high-level of abstraction (interface with the user - how to configure a device) to a low-level (the Knowledge Acquisition System) in the next sections. Note that, the KAS further rises the usability because it can be used to easily construct any knowledge configuration, and by consequence, design new HCI, different from the one described in this project. This notion will be explained in more detail in Chapter 4.

In addition, to aid on the understanding of the interface within Appendix B a figure presents the configuration environment.

3.1.1 How to Configure

As shown in Figure 3.1, to control we first need to configure the device and create temporal sessions, i.e. sequences of electrical impulses (modes), for each channel of the device.

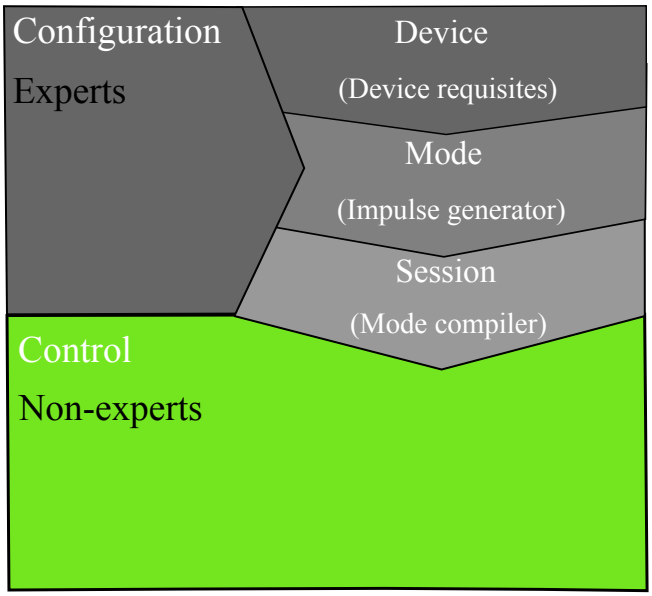

Figure 3.1: Configuration Diagram.

Thus, the configuration is hierarchically structured in Device, Mode and Session. This structure was strategically designed to meet the next requirements:

- Generic configuration, allowing the configuration of different devices;
- Configuration in sequence guiding the user;
- Simple, intuitive and user-friendly;
- Allow the user to generate any kind of impulse dependent on the device requirements and compile the impulses in a temporal session.

The user has the possibility to store information through three different editors: Basic, a simple and user-friendly interface with the minimum required fields to fill, intended for the basic and fast users; Source, an editor to upload or create JSONs, intended for users with more knowledge of the data structure, or for the ones that just want to copy and paste information; Advanced, a form generated automatically from Schema specifications (more information about JSON Schema in Section 2.5), which shows directly how the information will be saved in the JSON. These editors are explained in more detail in Section 4.3.

The following subsections describe each part of the configuration and how they are connected.

3.1.1.1 Device

The user can compose a new device by first setting some fundamental requisites for the impulses in the Device Configuration (limiting some properties in the impulse generator in the Mode Configuration).

The information required by the user is specified in Table 3.1. Through these specifications a Schema, named "Device.json", was implemented, and the three editors in Device Configuration are based on it (this mechanism is discussed in more detail in Chapter 4). The properties "Device Name" and "MAC Address" are required, the others become default, so in the Basic Editor the only fields are "Device Name" and "MAC Address". If the user wants to fill the other properties he needs to use the Source or Advanced editor.

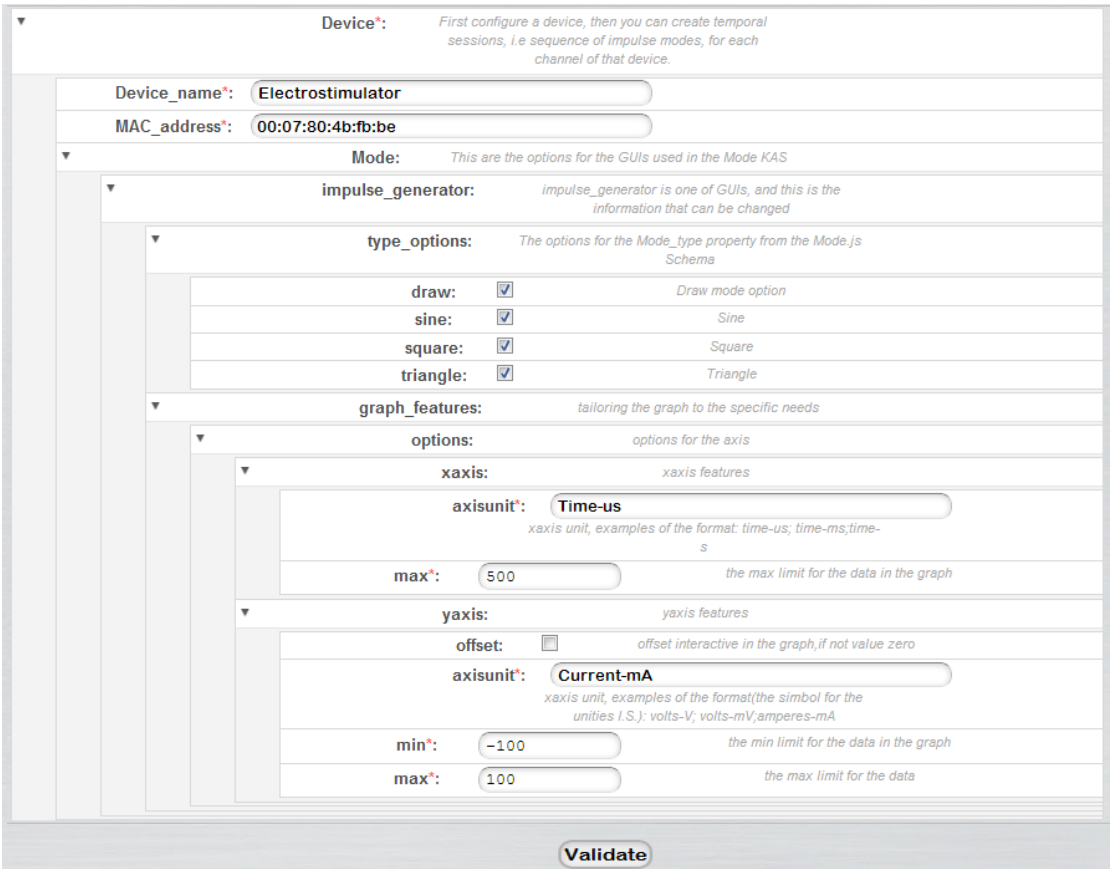

Figure 3.2: Device Configuration of an electrostimulator in Advanced Editor. The italic grey letters are the description of each property.

To illustrate the result of a Device Configuration an electrostimulator was configured. In the case of an electrostimulator specific options need to be satisfied. The specifications for the potentials and limits of ES are dependent of the application and the electrostimulator. The requirements of the electrostimulator we used for this project are:

- Pulse amplitude:
  0-100 mA (1 mA step). So the y axis data limits are: max = 100 mA; min = -100 mA

Table 3.1: Information required in Device Configuration.

Property	Definition	Default	Restrictions
Device Name	The name of the device.	This property is required, so it has no default value.	The name should be a string without spaces. Instead of spaces use "_".
MAC Address	Short for Media Access Control Address, a hardware address that uniquely identifies each device.	This property is required, so it has no default value.	The MAC Address should have this structure: "xx:xx:xx:xx:xx:xx".
Type Options	Property that specifies the impulse generator types active in the Mode Configuration. The types are square, sin, triangle and draw (defined in Section 3.1.1.2).	square, sine, triangle and draw are active.	
Axis Units	They are the units for the impulse data, in the x axis and the y axis. This property is important, because by defining the units, when parsing the information to the hardware, it permits to convert the numbers with the right units. Also, it enables a tool in the graph from Mode Configuration to show that units (see Figure 3.4). The format has to be "value-unit", examples: "time-s" specifies time in seconds; "time-us" specifies time in microseconds; "current-mA" specifies current in miliamperes; "potential-KV" specifies potential in kilovolts	"x-" specifies x axis without units, and "y-" specifies the y axis without units.	The format is restricted to "value-unit", value can be anything but the unit needs to be in the correct format, for example volts: GV(GigaVolts); MV(MegaVolts); KV(KiloVolts); V(Volts); mV(miliVolts); uV(microVolts); nV(nanoVolts).
Data Limits	Limits for the impulse in Mode Configuration (see Figure 3.4). It's important for the graph axis size and for the user to insert some restrictions in the data.	x axis: max limit = 10000; min limit = 0. y axis: max limit = 1000; min limit = -1000	It should be a integer number.
Offset	It's an option to activate or disable the offset in the graph. If the offset is disabled the user can't change it in the Mode Configuration, thus it stays at zero.	False	

- Pulse Width:
  0-500 µs (5 µs step). So the x axis data limits are: max = 500 µs; min = 0 µs
- Pulse frequency:
  1-200 Hz (1 Hz step)
- Number of channels:
  2

- Number of modes:
  4
- Waveform type:
  Rectangular, triangular, sinusoidal, customized waveform (constant potential with no offset). So the four types of impulse generation need to be active and the offset should be disable.

Considering the requirements, Figure 3.2 presents the configuration of an electrostimulator. After this information is saved the user can project modes and sessions for this device.

3.1.1.2 Mode

In the Mode Configuration, after defining the device requirements, the user may project new impulses. Attention, Mode and Session Configuration are only enabled if Device Configurations were saved previously.

Table 3.2: Information required in Mode Configuration.

Property	Definition	Restrictions
Parent Name	The name of the parent configuration. In this case the name of the Device. The name is inserted automatically.	The name should be a string without spaces. Instead of spaces use "_".
Mode Name	The name of the mode.	The name should be a string without spaces. Instead of spaces use "_".
Offset	The data offset. The default value is zero.	Offset value has to be within data limits.
Graph Points	The array with the necessary points. These correspond to the interactive points from the graph in Basic Editor.	The points can't surpass the data limits defined in the Device Configuration.
Draw	It belongs to the information retrieved from draw method. Divided in an array with 100 points, and a string with an equation. These two fields are only required when the user projects the impulse in Basic Editor with the Draw type.	
Frequency	Frequency representing impulse per second. The default value is zero, meaning that the next impulse begins right after the first, like a periodic wave.	Ranges from 0 to 200 Hz.

The information required by the user is specified in Table 3.2. Through this specifications a Schema, named "Mode.json", was implemented, and the three editors in Mode Configuration are based on it (this mechanism is discussed in more detail in Chapter 4).

In this case all fields are required, thus the simple, intuitive and fastest way to fill them is the Basic Editor. In it, Mode provides four types of impulse configuration: **square**, rectangular pulse that can be changed within the data limits; **sine**, sinusoidal pulse that

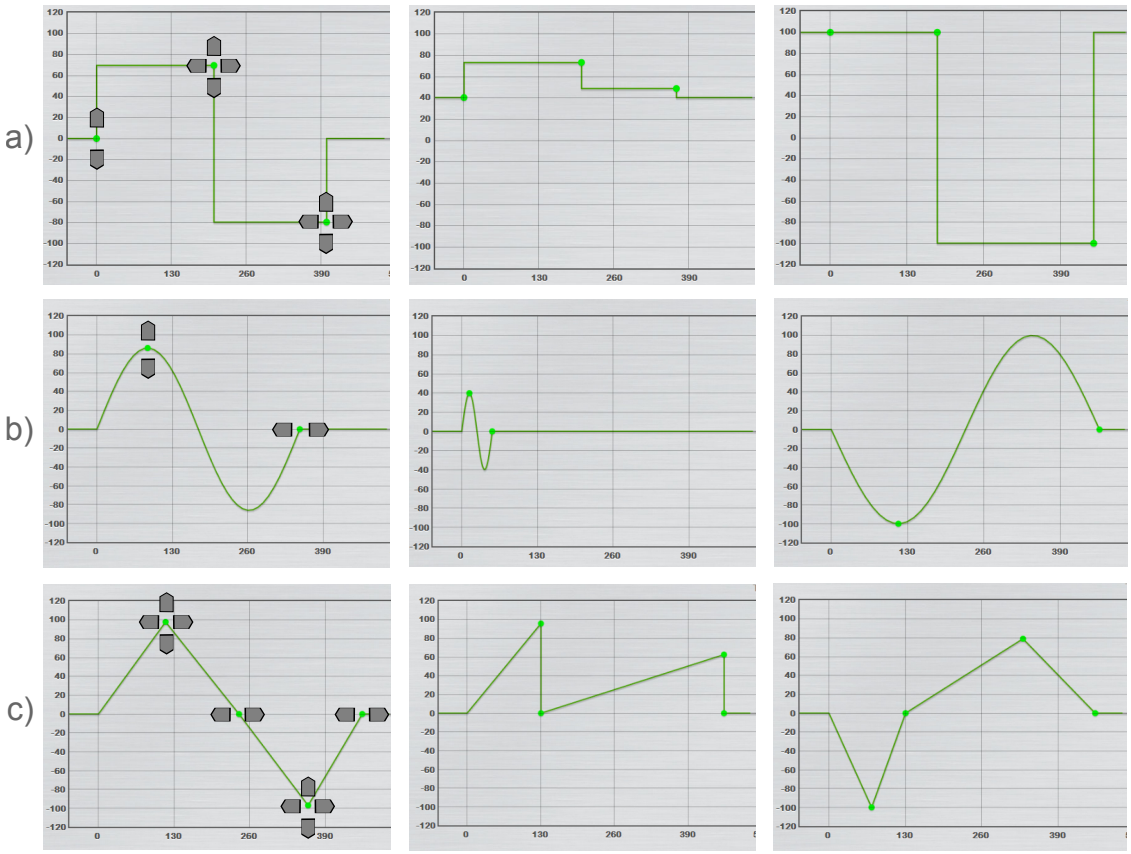

Figure 3.3: Example of the default impulses with the restrictions specified in the Device Configuration for the Electrostimulator: a)square with offset active, b)sine and c)triangle. The arrows mark the direction in which the interactive points can be moved. In each row are presented examples of the respective impulse after moving the interactive points.

can be changed within the data limits; **triangle**, triangular pulse that can be changed within the data limits; **draw**, a novel and innovative impulse generator. The user is also able to choose the offset (if active in Device Configuration) in the graph and the frequency (impulse per second) by moving a slider.

In the three default types square, sine and triangle, the end user just has to click and pull the interactive points (as you can see in each row of Figure 3.3): **square** has two interactive points that can be moved along x an y axis within the data limits, if offset is active a third interactive point materialize to move the data along y axis; **sine** also has two, one for the amplitude and other for the period, if offset is active a third interactive point materialize to move the data along y axis; **triangle** has four with two points that just move along the x axis, if offset is active a fifth interactive point materialize to move the data along y axis. Additionally, to help the user the value of a interactive point is presented when the mouse hovers that point(with the information defined in the property "axisunit" in Device Configuration), see Figure 3.4. The types described allow the end user to conceive a lot of different impulses, but to be able to design any type we had to devise a new form of impulse generation, the **draw** method.

The **draw** method enables the user to draw an impulse within the data limits. From

the draw 100 points are retrieved and will be processed by an algorithm. The idea behind this algorithm it's to search the points that seem connected to each other, then propose a series of simple equations to describe the links. The best are selected, tweaked, and again tested against the data. Next, the algorithm repeats the cycle over and over, until it finds equations that have a good probability of modulating the data. Then, by choosing one of the equations, the user is presented with a 100 points graph based in the equation and can compare it to the 100 points from the drawing. If it fits the user objectives he can save the equation, if not, the user can tweak the equation manually and see the effects, or just save the 100 points from the draw. This algorithm will have large benefits when finished, because it provides a mechanism of storing the draw data mathematically structured. In this way, the user will have the possibility of re-editing the data by changing some parameters in the stored equations. So, instead of just providing a draw method to construct impulses, an equation method is also available to the user for creating impulses (important to simplify the generation of impulses when the impulse can be described by an equation and only few parameters need to be changed).

This algorithm is still in development stage, it's based in the work of Schmidt and Lipson, the Eureqa, a highly praised symbolic regression program (see Section 2.3 for information regarding symbolic regression) [38].

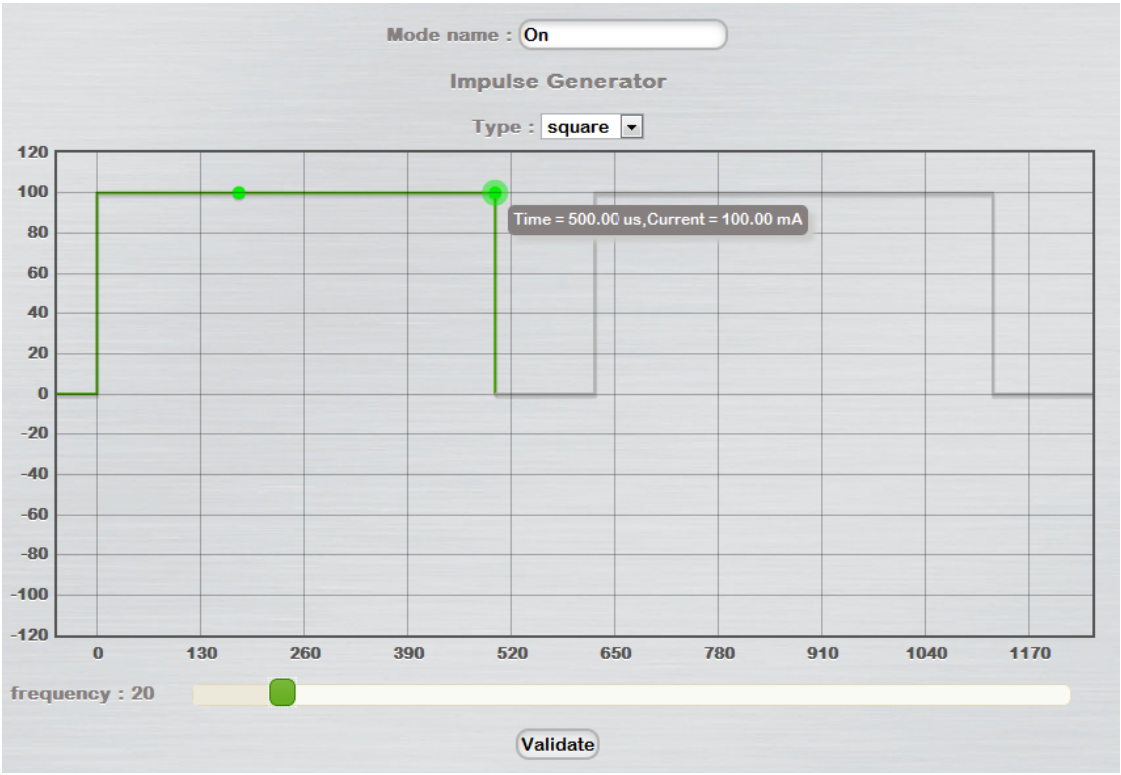

Figure 3.4: Mode Configuration for an electrostimulator in Basic Editor. In one of the interactive points a tool to show the value of the point is presented ("Time-us" and "Current-mA" was the information defined in Device Configuration). Data limits are: for x axis, max = 500 µs and min = 0 µs; for y axis, max = 100 mA and min = -100 mA. The mode configured is "On": rectangular pulse with 500 µs, amplitude 100 mA, and 20 Hz of pulse frequency (i.e. 20 pulses per second).

Figure 3.4 shows an example of how a mode can be configured in Basic Editor for the electrostimulator defined previously at Device section. Therefore, the impulse can be manipulated within the data limits for x axis, 0 - 500 µs, and for y axis, -100 - 100 mA. The mode configured is "On", a rectangular pulse with 500 µs, amplitude 100 mA, and 20 Hz of pulse frequency, i.e. 20 pulses per second.

Advanced and Source editor can also be used. The user can interchange between the three editors and manipulate the information in each one of them. Nevertheless, using Advanced and Source in Mode Configuration is not the fastest option.

With modes stored, the user can define temporal sessions by compiling the modes. In the next section this process is explained in more detail.

3.1.1.3 Session

The user can create a new session for a specific device if modes were previously saved for that device. A session is a sequence of modes, and each channel of a device can have one session programmed. A device has a maximum number of programmable modes, but they can be repeated infinitely. This mechanism is very important to program stimulation protocols, automating the control.

Table 3.3: Information required in Session Configuration.

Property	Definition	Restrictions
Parent Name	The name of the parent configuration. In this case the Device. The name is inserted automatically.	The name should be a string without spaces. Instead of spaces use "_".
Session Name	The name of the session.	The name should be a string without spaces. Instead of spaces use "_".
Modes to Use	This property is an array where the user has the possibility to add the modes that will be programmed in the hardware and will be used to compose a session.	The user can only define modes saved previously. The number of modes added can't be superior to the maximum capacity of the hardware.
Channel Session	Array where an item is a session for one channel. To create the session, the user needs to define "Loop Session" and the "Temporal Lines" for the session. "Loop Session", enables the user to choose if a session will be repeated until the stop defined by the user. "Temporal Lines", is an array where one item represents one temporal line, here the user compiles the modes: first by choosing a mode from "Modes to Use", then specify the duration for that mode.	The stop in "Loop Session" is in hours. The duration of the mode in "Temporal lines" is in seconds.

The information required by the user is specified in Table 3.3. Through this specifications a Schema, named "Session.json", was implemented, and Source and Advanced

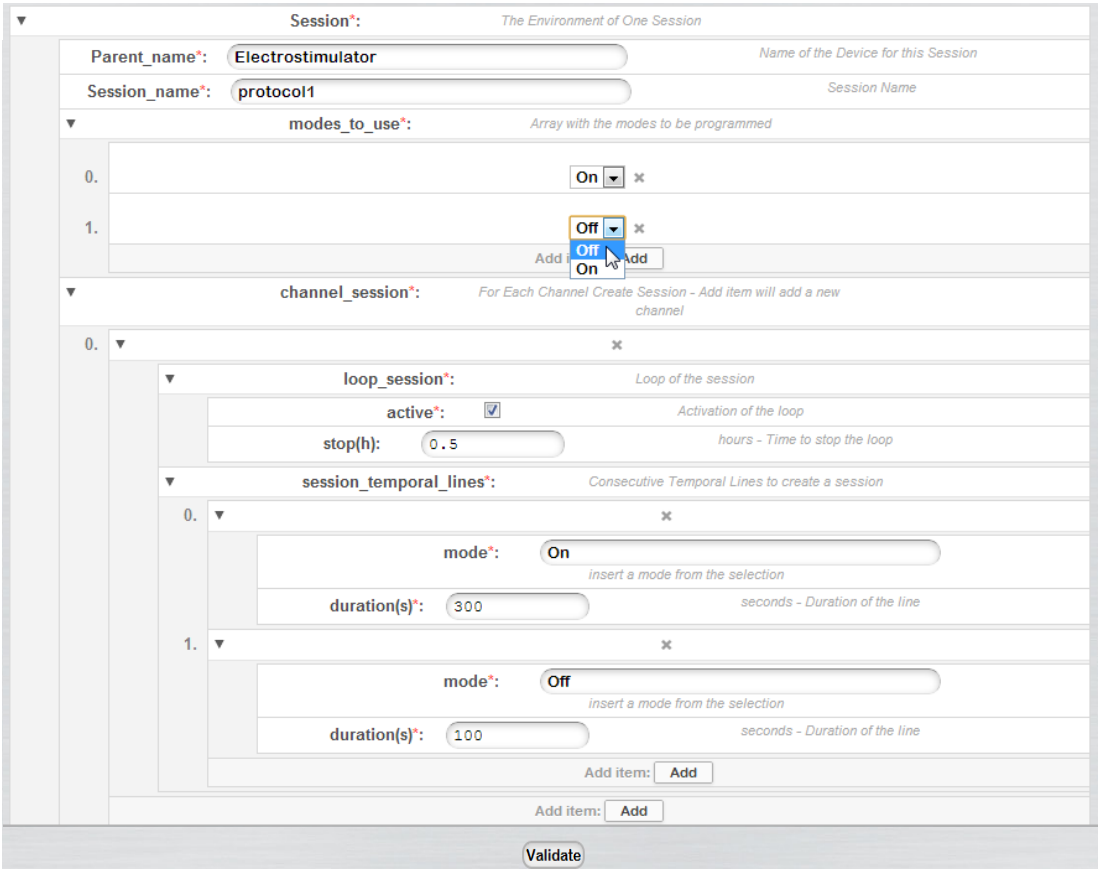

Figure 3.5: Session Configuration for an electrostimulator in Advanced Editor. One channel is programmed with two modes designed previously in Mode Configuration (they are selected through a special GUI, a pull down menu with the modes from Mode Configuration): "On", 300 seconds activated and, "Off", during 100 seconds. This sequence is repeated during half an hour by activating the loop for that specific session. The grey letters are the description of each property.

editor in Session Configuration are based on it (this mechanism is discussed in more detail in Chapter 4). The Basic editor has no interface defined because the Advanced editor can't be simplified.

To illustrate the configuration a session will be programmed for the electrostimulator defined in the Device section. The electrostimulator used in this project has two channels that can be programmed with four modes: first we define the modes that will be used, then we add channel sessions. In each channel session we have to define if the session has a loop, then we need to sequentially compile the modes. Figure 3.5 shows an example of a Session Configuration. Primarily, the modes that will be used need to be defined, four is the maximum that the hardware can memorize. After that, creation of a session for the channels required, in this case, only one channel is programmed with two modes (designed in Mode Configuration): "On.json", 300 seconds activated and, "Off.json", during 100 seconds (Off is a pulse with 500 µs and amplitude 0 mA). This sequence is repeated during half an hour by activating the loop for that specific session.

As a resume example in how to configure, Figure 3.6 shows a configuration sequence

example. First configure a Device, if Device Configurations are stored upload by clicking in symbol "+", now Mode Configuration and Session configuration are enabled for the Device chosen, in this case an "Electrostimulator". Next, the user clicks in button "conf_Mode" to configure Modes for the "Electrostimulator", after that the user configures sessions with the modes programmed. Finally, after the user store sessions for the device he can go to the control area by clicking in the "Actuation" button.

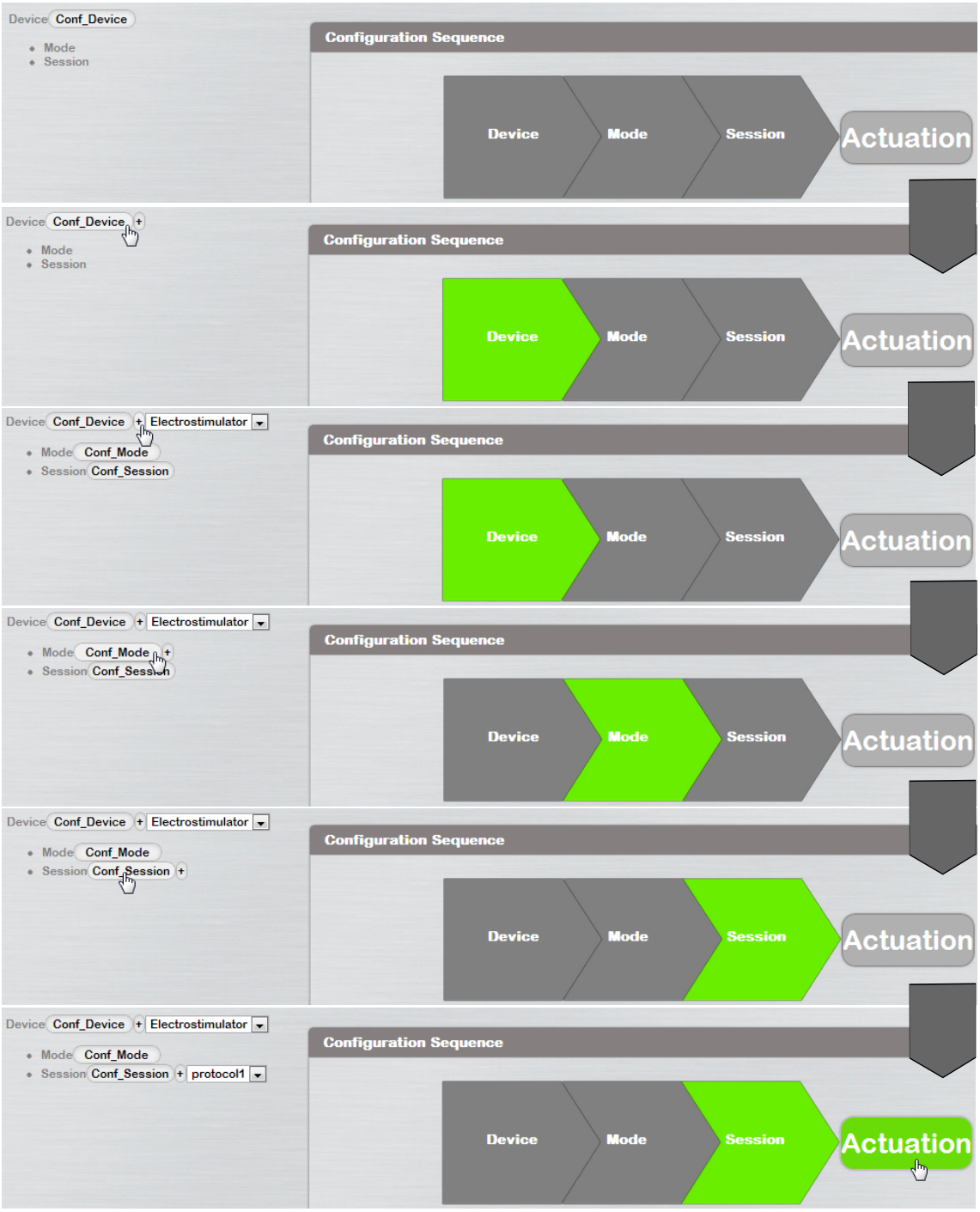

Figure 3.6: Hierarchy and configuration sequence from HCI.

3.1.2 Knowledge Acquisition System

The KAS developed in this project is the mechanism behind the configuration in the HCI. It provide methods to acquire knowledge from the end user and save this knowledge semantically structured. Despite the fact this tool aids in the architecture of the HCI, it still is independent of the later and is the fundamental tool devised for this work. Therefore, to highlight the KAS its description can be read in Chapter 4.

3.2 Control

Figure 3.7: Actuation of the session "protocol1" in the electrostimulator. The connection and the set up were correctly performed. The green start button means the session is running. Clicking in CH1 button opens a division with information about the session.

The HCI provides means to control different devices. One of the ascending examples is health care devices for monitoring and actuation, propelled by projects like Ambient Assisted Living (AAL) [65]. These devices need to satisfy certain norms, for example, patients could need to start and stop the device but they can't manually alter the sessions prepared for them, because they don't have the necessary expertise. Other norm, clinical professionals can actuate the devices via Internet, nevertheless, the process needs to be precise, secure, simple and fast.

The examples from the last paragraph served to delineate the control requirements. Conveniently, the HCI benefits from the division in configuration and control, assisting clinical professionals to automate different sessions of actuation for each patient. When needed the sessions can be easily started/stoped in Control by the patient (alter the session in Control is impossible), or by a clinical expert via net. As a result, the Control respects the norms depicted previously.

In the next sections the control is described in more detail.

3.2.1 How to Actuate

Control, division for actuation and acquisition of biosignals. Within Control the user first needs to choose the device, then connect to the hardware by using the MAC Address previously configured in Device Configuration or update it, if the connection is correctly established the user can select the session to program and finally set up the device with the information needed. When the device is ready, start/stop of the session is feasible. The user can also select channel information by clicking in the appropriate buttons, as it can be seen in Figure 3.7.

Figure 3.7 illustrates the actuation of the session "protocol1.json" in the electrostimulator: one channel is programmed with two modes, "On.json", 300 seconds active and, "Off.json", during 100 seconds. To start the session, first the connection was establish with the electrostimulator, second, the device was programmed with the session "protocol1". The information provided by clicking in "CH1" refers to the temporal lines for this particular channel (we hope in the future to have a real time graphic representation of the channel session instead).

In addition, during the session it is possible to acquire real time biosignals and synchronize them with the session, due to the fact Control is incorporated in a Biosignals Acquisition System from PLUX - Wireless Biosignals, S.A..

3.2.2 API

Different hardware have distinct commands, for each device to be controlled it's vital to insert specific commands into an Application Programming Interface (API), devised to translate the information saved in the Session Configuration to hardware language. Therefore, in the background an Application Programming Interface (API) makes the bridge between the software and the hardware, parsing the high level programming language to machine language enabling the control of the hardware.

Through this API the hardware sets up with the necessary requirements for the protocols of stimulation, and the start/stop of the session. It was designed with a degree of abstraction to enable quick location and implementation of the hardware commands in the right place.

Section 5.1 describes in more detail the API with the implementation of specific commands for an electrostimulator.

4

Knowledge Acquisition System

Internet is a Media type where countless data, growing every day, are shared and stored. This exponential growing entails serious problems, since data are typically weakly structured leading to a arduous management of information. The World Wide Web Consortium (W3C) devoted effort to solve this problem by promoting semantic structured data, like ontologies (see Section 2.4), enabling machines to perform more work involved in finding, combining, and acting upon information on the web. Pursuing the same path, a Knowledge Acquisition System (KAS) was developed. Written in JavaScript using JavaScript Object Notation (JSON) as the data structure and JSON Schema to define that structure, enabling new ways of acquiring and storing knowledge semantically structured (see Section 2.5). Taking this in account, its final objectives are: to provide programmers an easy mechanism to define ontologies, and through them automatically generate simple and intuitive forms with GUI to acquire knowledge from the end user, assisting in the fast re-edition (just by tweaking with the schemas) or construction of new configurations; enable the data to be stored semantically structured and human readable.

The objectives were attained, leading to the construction of a dynamic acquisition system. The forms to acquire information are not static and are easy to re-edit manipulating the schemas, instead of re-configure the whole software. For example, including new fields, delete or re-edit it's basic, the only thing that will be changed is the JSON Schema (versatile and simple to re-define).

KAS is the mechanism behind the Configuration in the HCI. As shown in Figure 4.1 an ontology is defined with the help of JSON Schemas, then these Schemas are interpreted in JavaScript enabling the conception of three editors. These editors provide different means to acquire the information from the user. Afterwards, this information is saved as JSONs in the server. In the Control, the JSONs are retrieved through websockets

from the server and, finally, the necessary information is sent to the device via an API.

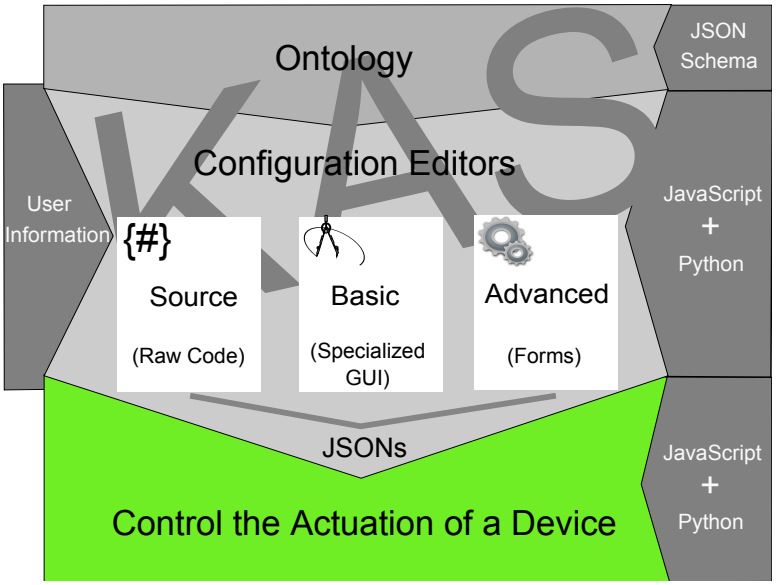

Figure 4.1: Illustration of how the KAS fits in the HCI.

KAS structure was design as a response to the HCI Configuration requisites:

- Generic configuration;
- Configuration in sequence guiding the user;
- Simple, intuitive and user-friendly.

However, it excels the response. The data structure combined with specialized GUI equip the fields of HCI, IR, KDD with an easy way to define data semantically structured and self describing (human readable), the JSON Schemas. Then, from the data definitions (JSON Schemas) automatically generate simple editors of information to acquire knowledge from the end user.

In the next sections the structure presented in Figure 4.2 is thoroughly depicted.

4.1 Data Structure

The easiness of access, storage, transmission of data and the exponential proliferation of Internet users enclose new complexities: authentication; scalable configuration management; security; huge masses of high dimensional and often weakly structured data (the main problem addressed in this project). Structuring data pursued by Semantic Web creates opportunities to improve and develop more adequate languages.

Methods and approaches to solve the data structure problems emerged from research in Human-Computer Interaction (HCI), Information Retrieval (IR), Knowledge Discovery in Databases and Data Mining (KDD). These methods assist end users to identify, extract, visualize and understand useful information from data. The establishment of

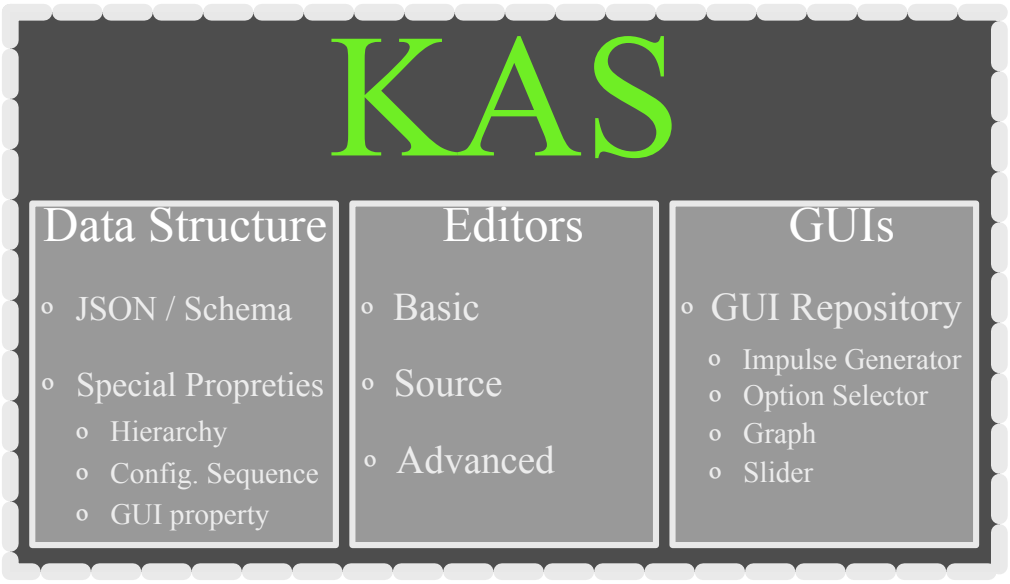

Figure 4.2: KAS Generic Diagram.

ontologies is one of the methods to solve the problem, possible with Semantic Web, holding much promise in manipulating information in ways that are useful and meaningful to the human user (see Section 2.4). Mechanisms to specify ontologies have spring in the late years, the Schema Language is one of them. Other great mechanism, developed by Crockford, is JavaScript Object Notation (JSON): a simple, lightweight and human readable text-data structure for information exchange (see Section 2.5). Together with JSON Schema, a Media Type (standard draft of options) that specifies a JSON-based format to define the structure of JSON data, they become an excellent way to define data semantically structured and human readable. But, the rewards in using this combination has other advantages discussed in this section.

JSON Schema specifies requirements for JSON properties and other property attributes with the following intentions:

- Validation (data integrity);
- Documentation;
- Interaction (UI generation - forms and code);
- Hyperlink Navigation.

JSON can be easily interpreted in any programming language and parsed to literal objects, thus manipulating the data is simple. JSON Schema is also in JSON format, therefore the programmer can easily define new properties, and use them to create taxonomies and inference rules.

In this project, as explained in Section 3.1.1, the Schemas "Device.json", "Mode.json" and "Session.json" were implemented according to the HCI specifications (defined in the respective tables in Section 3.1.1). Also, special properties were defined to meet some of

Table 4.1: Special properties needed within the configuration Schemas

Property	Values	Purpose
"hierarchy_type"	Three types: "Parent", "Child" and "None". Property required in the Schema. It only exists one "Parent" per hierarchy, the others are first Childs, second Childs, ... (like a tree). "None" means there's no hierarchy.	The purpose is to create an Hierarchy: relations of parenthood between Schemas, i.e. classes and subclasses of configuration. The class specifies some restrictions to the subclass. Because of this hierarchy the subclass can only be configured if the above class in the hierarchy is already configured. This formulation automatically generates in the HCI a Hierarchy Configuration (see Figure 3.6).
"parent_title"	Two types: "name of the parent" and "None". Property required in the Schema. Enables the relation between parent and child, like in Search Query Languages (Primary Keys and Foreign Keys). "None" is for the "Parent" of the hierarchy, or if there's no hierarchy.	
"config_sequence"	Two types: number in the sequence - 1,2,... and "None". Property required in the Schema. "None" means that the Schema does not belong to the sequence.	To guide the user through the configuration a sequence needs to be established between the Schemas (see Figure 3.6).
"GUI"	The value is represented by "key-GUI type", key corresponds to the property being specified and "GUI type" the GUI related to the property. Property not required in the Schema, It should only be used to specify that a certain property is related to a specific GUI.	Specialized GUI are stored in a JavaScript file named GUI repository. The purpose is to related Specialized GUI to properties in the Schema.

the requisites established for the Configuration. To establish classes, subclasses of configuration, i.e. an hierarchy, provide a configuration sequence, and allow the application to automatically perceive where to generate specific GUI, the properties defined in Table 4.1 were implemented in the Schemas. To demonstrate it, Figure 4.3 shows a resume of how the hierarchy and configuration sequence was specified, and Figure 3.6 the results in the HCI. Primarily the user configures a device, after storing the information the user can configure the modes and sessions for the device, first the modes then the sessions, and finally the user can actuate the sessions. Detailed information about the relation between GUI and Schemas is described in the next section.

Recapitulating, as shown in Figure 4.1, JSON Schema allows the definition of ontologies and will be in the core of the KAS, enabling: **interaction**, Schema serves as

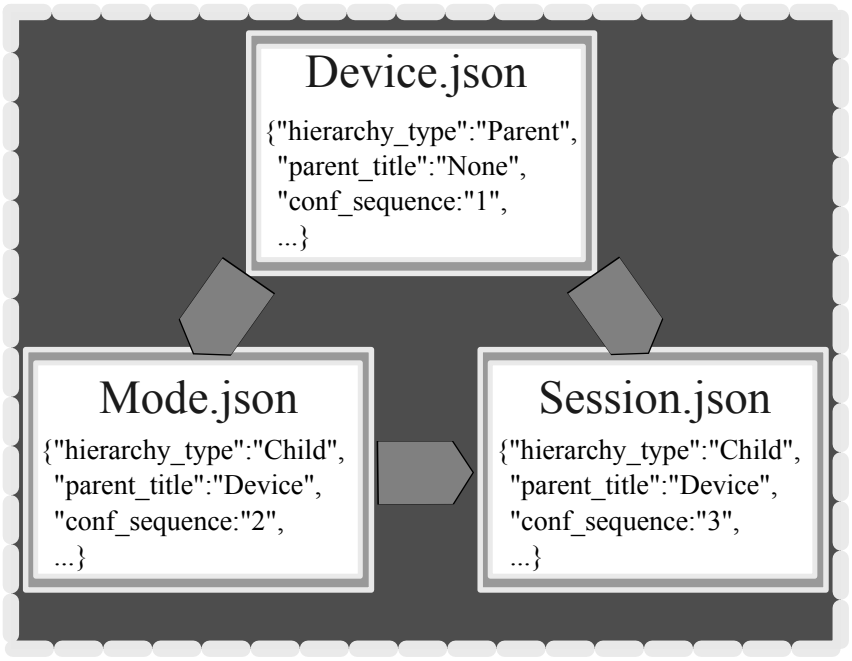

Figure 4.3: Example of the HCI Schemas concerning the special properties.

blueprint to architect the necessary forms, with or without GUIs, to define the editors for the Knowledge Acquisition System; **documentation**, APIs and low-level software infrastructures in JavaScript enable the transformation of the information acquired from the user in a JSON data structure defined by the Schemas (semantically structured and human readable), then the data (JSONs and Schemas) is stored in the server (Python) and retrieved to the client (JavaScript) through websockets (full-duplex communications channels over a single TCP connection); **validation**, the data is only stored if the structure and information agrees with the Schema and with the relations between Schemas.

With these mechanisms the software becomes a overarching system: robust; error - free; with human readable data structures (visualization and extraction of information is easier).

4.2 GUIs

One important part of the KAS is the automatic generation of simple forms derived from the data types defined in the Schema (see Figure 3.2). They provide a simple way of acquiring information. Although, sometimes, this method can be confusing and take a large amount of time in configuring. Thus, to answer the problem, specialized Graphic User Interfaces can be programmed and easily connected with data types, enabling: the construction of Basic Editor interfaces; Advanced Editor to have the possibility of default edition or in GUIs (see Figure 4.5).

For the HCI in question, special GUIs were developed to grant the user a straightforward, obvious and accessible configuration.

Impulse Generator is one of them, playing an important role by solving the problem of generate different impulses. It's a combination of three GUIs: Option Selector, Graph and Slider. The options to choose between are **square**, **sine**, **triangle** and **draw**, each one of them change the environment of the graph as described in Section 3.1.1.2. The change occurs due to the manipulation of different parameters, like graph data and graph structure. Default types **square**, **sine** and **triangle** have a special data structure generated by distinct functions for each type. These functions receive parameters correspondent to the interactive points, then, through its manipulation they help to construct the data resulting in the graphs depicted in Figure 3.3. Graph data limits and units are retrieved from the information saved in Device Configuration and they also form the parameters to built the graphs. **draw** method is distinct from the three default types described, it will change the entire canvas (place holding the graph) dividing it in a tool to draw and an equation editor. As described before, an algorithm will retrieve 100 points from the draw, compare them and propose a series of simple equations with more or less probability in describing the data. After that, the user has the possibility to tweak the equation and save the data mathematically structured in a function. The functions stored will provide means to architect the graph data, like in the default types. The slider, also analysed in Section 3.1.1.2, is used to simplify and restrict the frequency chose by the user.

Restrictions can be applied to Impulse Generator and its configuration state changes when information is uploaded. To restrict and define some of the impulse generator parameters, the next object was specified in the Schema "Device.json":

Listing 4.1: Impulse Generator Object

```
"impulse_generator": {
     "type": "object",
     "properties": {
         "type_options": {
             "type": "object",
             "properties": {
                 "draw": {
                    "type": "boolean",
                    "default": true,
                 },
                 "sine": {
                    "type": "boolean",
                    "default": true,
                 },
                 "square": {
                    "type": "boolean",
                    "default": true,
                 },
                 "triangle": {
                    "type": "boolean",
                    "default": true,
                 }}},
          "graph_features": {
```

```
                "type": "object",
                "properties": {
                    "options": {
                      "type": "object",
                      "properties": {
                        "xaxis": {
                          "type": "object",
                          "properties": {
                            "axisunit": {
                              "type": "string",
                          },
                            "max": {
                              "type": "number",
                        }}},
                        "yaxis": {
                          "type": "object",
                          "properties": {
                            "offset": {
                              "type": "boolean",
                            },
                            "axisunit": {
                              "type": "string",
                            },
                            "min": {
                              "type": "number",
                            },
                            "max": {
                              "type": "number",
}}}}}}}}}
```

This object is specific and can't be altered, because all pieces play its part as discussed in Table 3.1.

Activation of GUIs is another relevant mechanism, concerning the relation between the GUIs and the Schemas. To activate GUIs the "GUI" property, described in Table 4.1, needs to be specified. "GUI" property format is "key-GUI type". When specified, this properties are interpreted and the various combinations "key" - GUI are displayed sequentially in Basic Editor. In Advanced Editor the GUIs are connected with the specific "key" in the form (to open the GUIs the user has to click in the appropriate button). The "GUI types" available are: "string" and "number", related to a simple text GUI; "impulse_generator", related to the Impulse Generator GUI; "options", related to the Options Selector GUI; "Offset-graph" and "Graph_points-graph" are specific properties of the graph, they can't be altered; "mode_selection", is a specific pull down list of the modes configured used with the property(key) "modes_to_use" in Session Configuration on the Advanced Editor (see Chapter 3, Figure 3.5).

The next code resumes the Schema "Mode.json", demonstrating how to use "GUI" property:

Listing 4.2: Resume of "Mode.json" Schema with emphasis on "GUI" property

```
"Mode": {
    "type": "object",
    "properties": {
      "Parent_name": {
        "type": "string",
      },
      "Mode_name": {
        "type": "string",
        "GUI": "Mode_name-string",
      },
      "Impulse": {
        "type": "object",
        "GUI": "Impulse-impulse_generator",
        "properties": {
          "Type": {
            "type": "string",
            "GUI": "Type-options",
          },
          "Offset": {
            "type": "number",
            "GUI": "Offset-graph",
          },
          "Graph_points": {
            "type": "array",
            "GUI": "Graph_points-graph",
            "items": {
              "type": "object",
              "properties": {
                "x": {
                  "type": "number",
                },
                "y": {
                  "type": "number",
                }}}},
          "frequency": {
            "type": "number",
            "GUI": "frequency-slider",
}}}}}
```

The result of these specifications can be seen in Figure 4.4 and Figure 4.5. Figure 4.4 illustrates a Mode Configuration, "Bifasic", in Basic Editor. In Advanced editor, by clicking in the respective button the Impulse Generator interface aids the user in inserting the information in the "Impulse", increasing the speed of the configuration as it can be seen in Figure 4.5.

Resuming, the relation between the Schemas and the GUIs, allowed by the infrastructures developed amid KAS and GUI repository, supply the HCI with a mechanism to automatically spawn forms with GUIs out of Schemas.

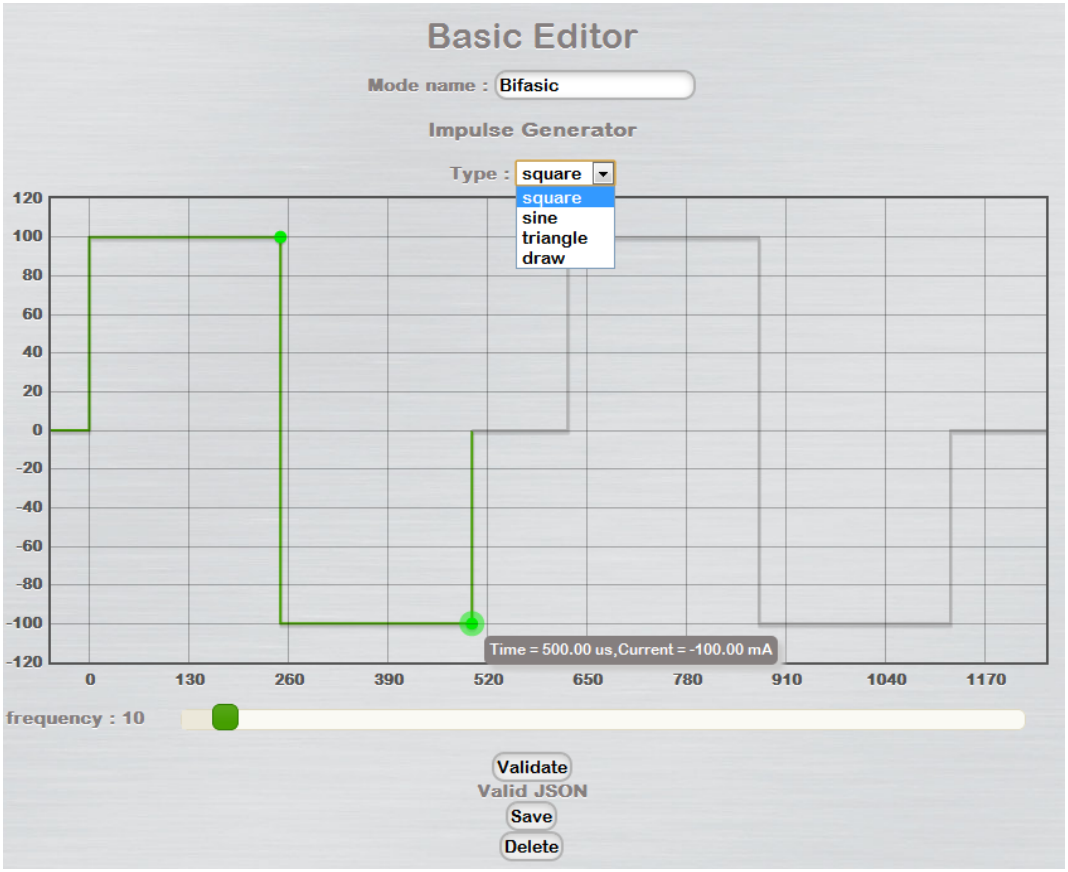

Figure 4.4: Example of a Mode Configuration in Basic Editor. The figure presents the options to choose, the tool for the interactive point, and after click in the validate button the appearance of the save an delete button if the information is valid.

4.3 Editors

The core for each configuration is a Schema, it assists not only in the specification of data semantically structured, but also in the fabric of three editors. Within the KAS, this three editors correspond to three functions receiving a Schema as parameter, then according to the requirements of each editor the function manipulates the Schema information to architect the forms and necessary interfaces. These three editors are entitled:

- Basic Editor (Specialized GUI): simple and user-friendly interface with the minimum required fields to fill. The information is saved in a JSON format defined by the Schema. An editor intended for basic users. It was already presented in Figure 4.4.

- Advanced Editor: a form generated automatically from the Schema specifications, with the possibility of edition in GUI. Information is also saved in JSON format defined by the Schema. The most profitable editor because it shows directly how the information will be saved in the JSON. This way, users understand the architecture of the data. Figure 4.5 presents the configuration of the mode "Bifasic", with

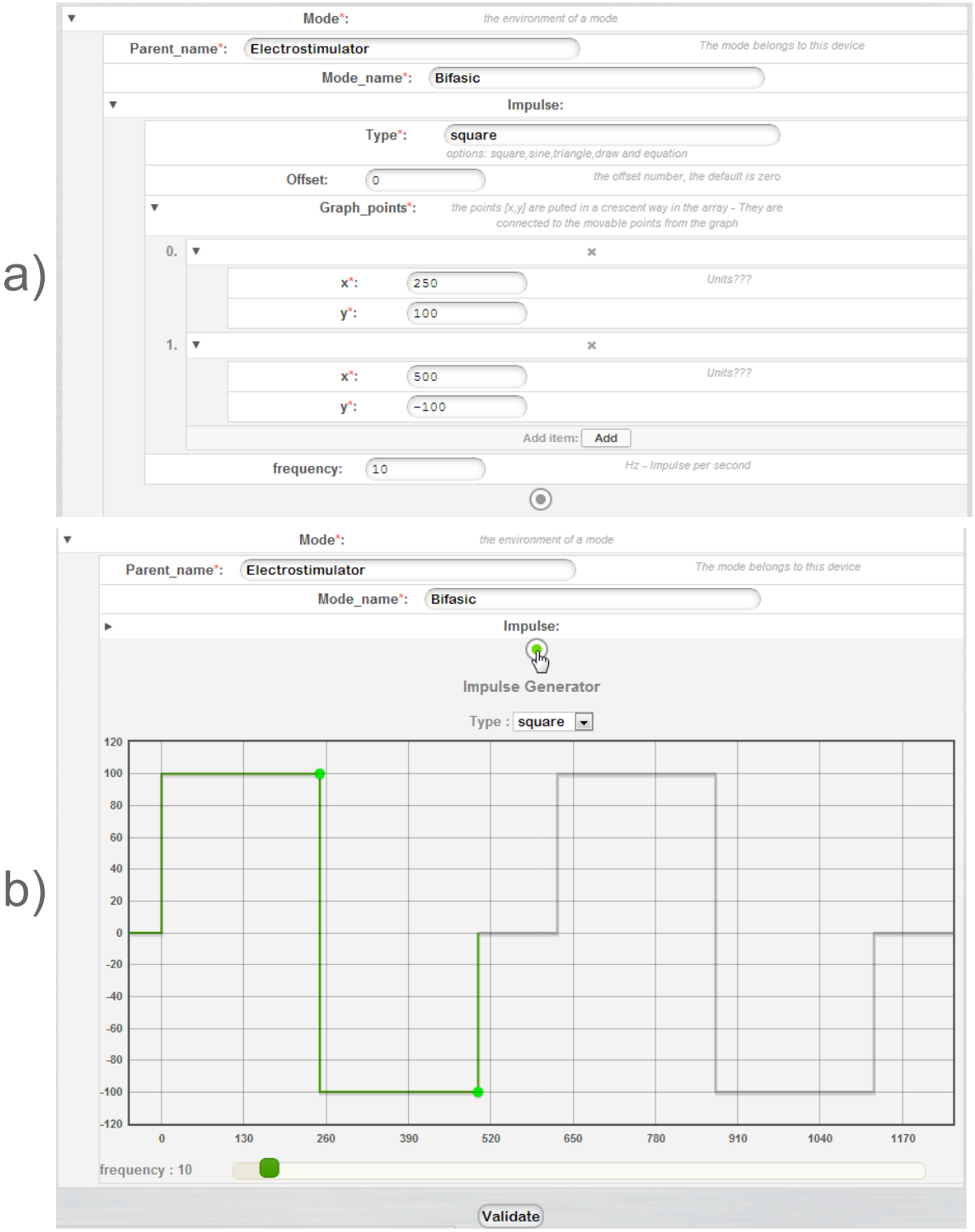

Figure 4.5: Example of a Mode Configuration in Advanced Editor. a)Presents a default configuration. b)By clicking in the button the user can configure the information in the Impulse Generator interface, then when the information is validated the "Impulse" fields are filled with the information from the Impulse Generator interface.

or without special GUI. Clicking on the specific button opens a graphic tool (canvas) for edition of points by moving them with the mouse, identical to the canvas from Basic Editor, as depicted before. Other example is Figure 3.5, illustrated in

Chapter 3, remarking the configuration of a session in Advanced editor.

- Source Editor: an editor to upload or create JSONs, where the user writes the JSON or copy and paste the JSON in the editor. Due to the fact JSON is compactly stored in a string, a format tool named "Pretty Print" is available. This Editor is intended for users with more knowledge of the data structure, or for the ones that just want to copy and paste information or upload. Figure 4.6 shows the Source Editor and the differences among JSON compacted and pretty JSON.

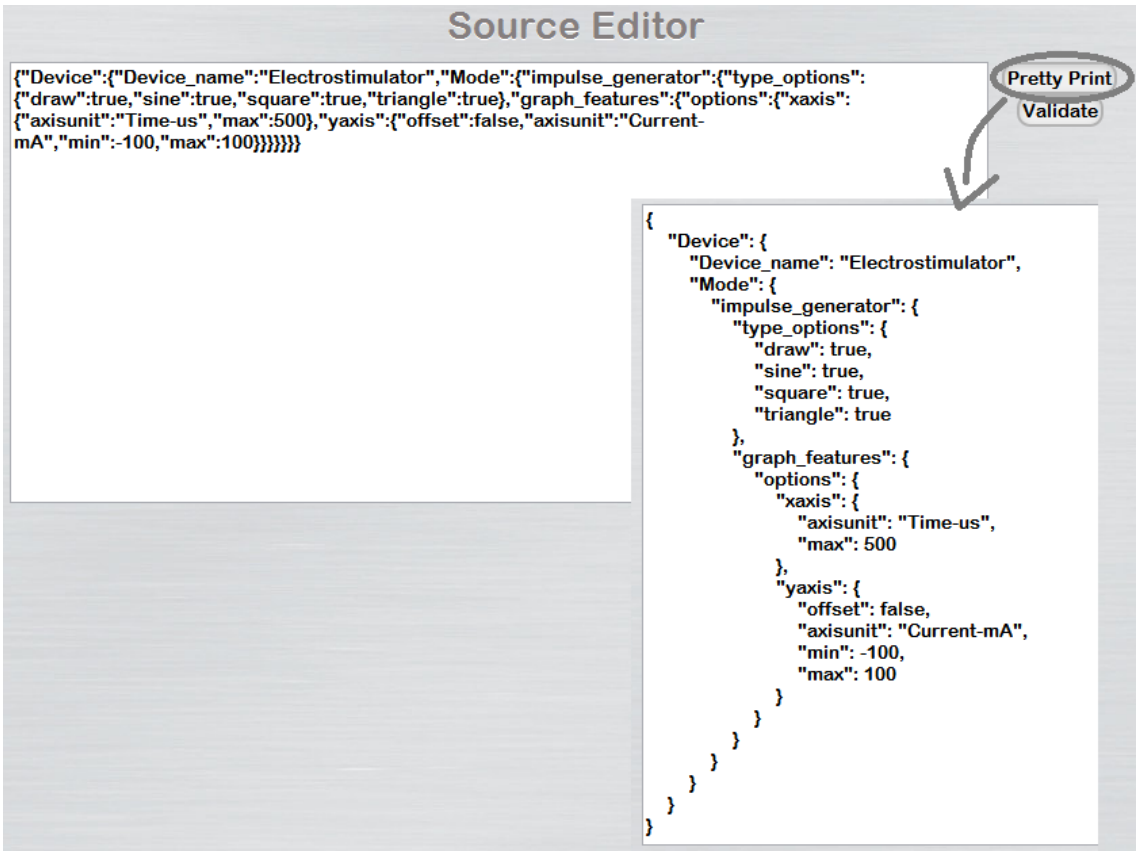

Figure 4.6: Example of Source editor and Pretty Print tool using the file "Electrostimulator.json" design in Device Configuration.

In the end, when all the information is set, in each editor is implemented a validation method. If the info agrees with the specifications from the Schemas save and delete is enabled, like in Figure 4.4, if not, reports with errors are generated. Note that it's possible to switch between the three editors (see Figure B.1 within Appendix B).

Three different editors aid the user to store knowledge semantically structured. This division was idealized to improve the acceptance and usability of the software, in the interest of supporting the different user necessities and to improve the software dynamic, design and performance.

5

Performance Evaluation

Several tests were conduced according to the requirements to evaluate the performance of the software and validate it. In the Table 5.1 the tests and results are discriminated elucidating the potential of the software and the necessity to optimize some parts.

The next section resumes an application of the HCI, used to test the configuration and control of an electrostimulator.

definition

5.1 Application

Electrostimulation is used in many areas (clinical, sports and research) to improve health lifestyle. Hardware and software solutions need to be orchestrated for administering electrical stimulus. Most of these solutions have specific purposes, only allowing limited stimuli protocols to be actuated. The tool manufactured in this project permits the configuration of electrical devices by sequentially compile electrical impulses in a temporal session. Applying this tool to control an electrostimulator concedes a solution to architect different stimuli protocols and actuate them.

An API makes the bridge between the software and the hardware, as referred before. The structure of this API, programmed in Python, was architect taking into account the session data structure and the easiness to locate and implement the hardware commands for each device in the right place. Primarily, the connection between the software and hardware is established using a function that receives the MAC address of a specific device as a parameter. Information regarding the connection is sent back to the software, if the connection was successfully completed, the user has the possibility to program

Table 5.1: Tests and results.

Requirements	Test	Results
Data semantically structured and human readable.	Evaluation of the data stored.	JSON data proved to be semantically structured and human readable.
Validation and documentation of the data acquired from the end user.	Verification of the process of validation.	The validation method provides a robust process to verify if the info acquired agrees with the schema specifications, if it does not, reports with errors appear. Nevertheless, the validation needs an upgraded to eval limits that subclasses inherited from classes.
	Verify if the information is documented properly.	Due to the fact the systems to acquire information are built directly from JSON Schemas, the information were always correctly documented.
Hierarchy and Configuration in sequence guiding the user.	Ask several users what they felt using the KAS.	All users documented the easiness in configuring, and because it is in sequence, they intuitively understand the structure of the configuration.
Generic configuration, allowing the configuration and control of different devices.	Test the control of different devices.	An electrostimulator and a LED were successfully controlled with distinct sessions. Although, to prove this requirement many more devices need to be tested.
Allow the user to generate any kind of impulse dependent on the device requirements and compile the impulses in a temporal session.	Test the generation of different types of impulse.	The three default types of impulse generator "square", "sine", and "triangle" were tested enabling the design of different types of rectangular, sin and triangular impulses, respectively. Regardless, without the "draw" method implemented, combining a draw and an equation editor, the user is not able to produce any kind of impulse.
Simple, intuitive and user-friendly software.	Ask different users to test the Configuration division .	They all documented how easy was to perform the configuration of a device without any external help. Nonetheless, they also referred the design need to be improved to make some parts more clear.
	Ask different users to test the Control division.	To control, the user just needs to set up a device by choosing one device and session to actuate, then start/stop the session is feasible. Due to this simple tasks the users remarked how fast and simple was to control a device when sessions are previously prepared.

sessions for this particular device. Thus, the second stage of the API is to use the information stored in a file session.json to program modes and sessions for each channel. In this stage, specific low-level commands to set the required information in the hardware for each device should be compiled, in this case the commands of the electrostimulator. Again, information is sent back to the software contemplating the set up, if the set up was successfully achieved, the user can start and stop the session and access to certain information concerning each channel session. This is the third stage where specific commands to start and stop the hardware are implemented. Finally, the last stage is to close

the connection between the software and the hardware each time a new device is selected to control, or when we terminate the Control application.

Resuming, the API constructed for the Control area is constituted by four sections: establish the connection, set up, start and stop the session and close the connection. These divisions can be visualized in Figure 5.1, in each sections specific commands need to be programmed to control the hardware. In regard of the electostimulator actuation, hardware commands were implemented and the tests results were positive: the user is, at the moment, able to design three distinct types of impulses (rectangular, sin and triangular) compile them in temporal sessions, and actuate them in the control. Nevertheless, improvements need to be made and they are discussed on Future Work within the next chapter.

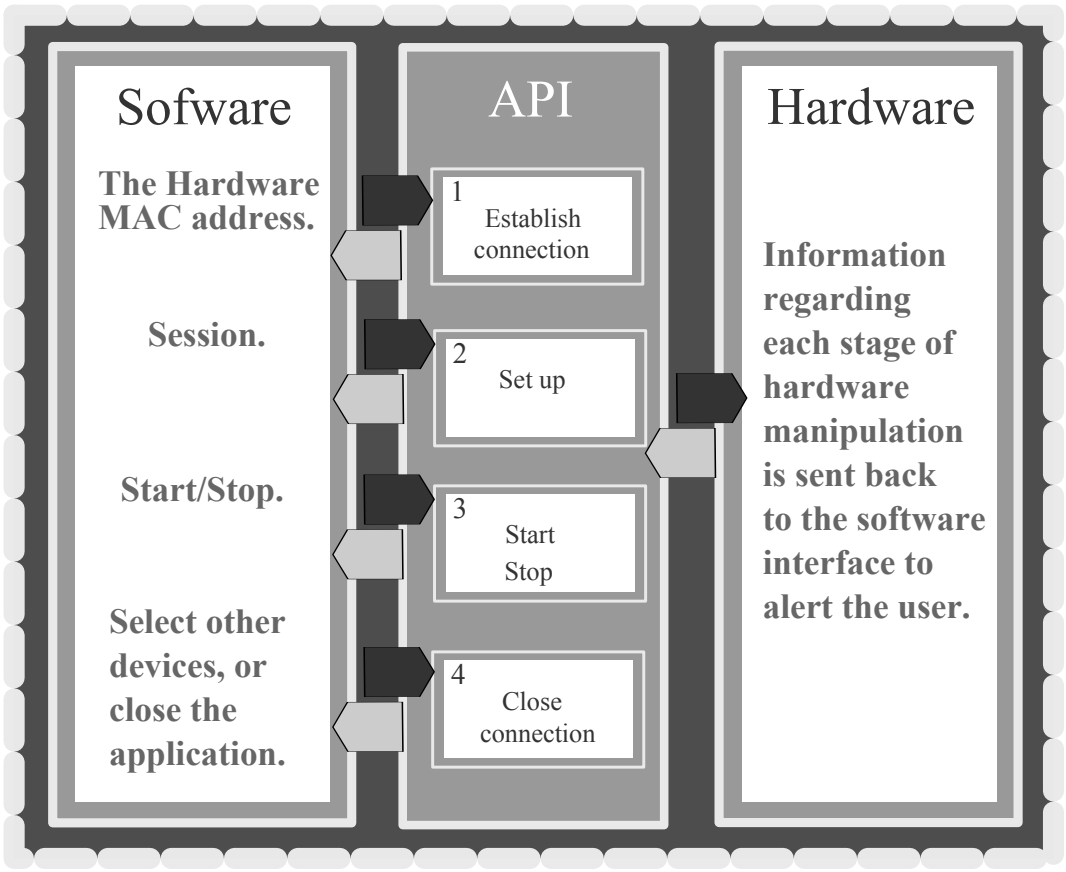

Figure 5.1: Diagram for the API, remarking the four sections developed, and the interchange of information.

6

Conclusions

Within this last chapter the work performed and its contributions are summarized, additionally future prospects are discussed. In other words, it outlines and evaluates the accomplishments and favourable implications (on the study areas) of the results attained. Due to the fact this is the prototype, software revisions, add ons and updates need to be performed in the future.

6.1 Contributions

This work was inspired, first hand, by the necessity to construct a Web-based software supporting the actuation of an electrostimulator, that would: be applied in different Electrostimulation applications; profit from Internet environments concerning exchange and store of information, for example, allow the configuration and control of a device at distance. However, its idealization surfaced from a higher level of abstraction, in behalf of the desire to develop a overarching software (without centralize it in Electrostimulation and be transferable across platforms). From this point of view, primarily the focus shifted towards the conception of a Web-based data structure (JSON and JSON Schema), the core of the software. Secondly, develop, in symbioses with the data structure, the foundations required to assemble the HCI.

The data structure was praised throughout the dissertation showing the pivotal place assumed in the software. It was envisioned to surpass Web-based data and biomedical problems.

Internet users growth and Internet flow of information encloses many problems when the data is weakly structured: entropy rises and information becomes disperse, uncorrelated, non-transparent and difficult to access and share. Nonetheless, research in HCI,

IR, KDD revealed methods to address these problems, assisting end users to identify, extract, visualize and understand useful information from data. For example, W3C proposed Semantic Web, which promotes semantic structured data, like ontologies, assisting machines in finding, combining, and acting upon information on the Web.

Using this knowledge, the data structure was built with a Schema language, JSON Schema, to specify JSON data semantically structured. Because JSON is a simple, human readable, lightweight and platform independent language key issues are elementary, like exchange, visualization, extraction or storage of information (JSON has no relevant semantic or syntax commands, it's an object notation with the format "key":"value"). The fact that it's easy to understand useful information from the data and different programming languages can access it, also plays an important role in solving biomedical problems. Transparent and correlated data turns its decoding and analyses manageable with any software, thus the data can be considered universal. For this reason, hospitals, clinics, and others will be able to transfer and manipulate information in a simplistic manner and saving money in the process.

As pointed before JSON Schema is effortless to employ, results from its simple syntax and the liberty the programmer has to produce all kinds of taxonomies and inference rules (applications can be developed directly from the Schema).

In this project a Knowledge Acquisition System was devised applying the Schema to: construct simple, intuitive forms and specialized GUI with configuration hierarchies and sequences, for acquiring information and guide the user through the configuration; validate the information acquired, and finally, if the information agrees with the Schema specifications store it semantically structured. Taking in account these purposes, three editors were designed to meet the expectations and necessities of different users: Basic, a simple and user-friendly interface with the minimum required fields to fill, intended for the basic and fast users; Source, an editor to upload or create JSONs, intended for users with more knowledge of the data structure, or for the ones that just want to copy and paste information; Advanced, a form with specialized GUIs generated automatically from Schema specifications, which shows directly how the information will be saved in the JSON. For these reasons, KAS is a multi-purpose mechanism promoting a fast, simple, intuitive re-edition and creation of new configurations, due to the automated generation of forms and GUIs from the Schemas(with hierarchies and sequences of configuration to guide the user, specialized GUIs to simplify the acquisition of knowledge). Besides, KAS is independent of the HCI constructed in this project, it can be used to create other robust HCIs without loosing large amounts of time, since creating editors to acquire knowledge is easy and fast.

Data structure and the KAS are the foundations of the HCI, conveying the mechanisms to fulfill the requirements established for the project.

Our HCI aims to automate the control of biomedical devices (and other electrical devices) by sequentially compile electrical impulses, allowing the design of actuation sessions that later can be applied. In order to achieve this objective, a frontier was conceived

dividing the HCI in Configuration and Control. The goal was to simplify the control of a device by automating the actuation sessions and separate the software to experts and non-experts, needed in some biomedical devices. Accordingly, in Configuration, experts on different areas are able to program and store sessions needed for the devices they use, then experts and non-experts can actuate the sessions in Control division.

Configuration was built using the KAS, for that, three Schemas were implemented to construct the needed interfaces: "Device" - the first configuration - for acquiring the device requisites in order to restrict the impulse generation; "Mode" - the second configuration - a section to design different impulses with ease, there are three default GUI types helping the user in the creation of rectangular, sin and triangular impulses, plus, it will be available a draw editor combined with an equation editor; "Session", the third configuration, where the user chooses and compiles the modes for the channels of the devices. With this configuration sequence the user is able to intuitively automate sessions for different devices, and save/delete or modify sessions previously stored. Moreover, the three default GUI types are graphs that inherit the restrictions provided in the "Device", within the user just has to move interactive points to design rectangular, sin and triangular pulses, a feature befit to the new technologies of mobile touch screens. The draw combined with the equation editor it's an indispensable tool to accomplish the control of practically all electrical devices, although it's still in development stage.

Changing division, Control allows non-experts users to actuate the sessions stored for the different devices. Control is a simple and intuitive interface developed to set up the device and start/stop the session. In addition, because the Control is incorporated in a Biosignals Acquisition System from PLUX - Wireless Biosignals, S.A., real time acquisition and signal analysis is feasible and can help to devise a tool with a closed-loop cycle (biofeedback control).

Another important feature is the Web-based platform: Configuration and Control can be carried at distance between users of the same software; exchange and storage of information are far more dynamic then a software built with non Web programming language; evolution and adaptation to new technologies (wireless biosignals, mobile solutions).

Evaluation and validation were performed through several accurate and objective tests. These tests had the intention to easily design and save different types of stimuli protocols, and actuate them in an electrostimulator, surpassing the specific use of many Electrostimulation software. The results obtained were able to validate this first version of the software and confirm its potential. Nevertheless, some goals stay unfinished and new ones arise, these are outlined in the next section.

In the context of this research work two papers were published, one regarding the structures of the HCI, and the other the aforementioned application of the software on Electrostimulation and it's benefits. The first paper was accepted for oral presentation in WINSYS 2012 conference and published in the conference proceedings. The second was submitted to BIOSTEC 2013 conference. These publications are presented in Appendix A.

In general, this project equips several areas (KAS, HCI, IR, KDD, Electrostimulation, generic device control,...) with simple but powerful and robust tools for the emerging technologies linked to the Internet. KAS aids in the simple, intuitive and fast construction and re-construction of forms and specialized interfaces to acquire any type of knowledge, by specifying JSON Schemas with taxonomies and inference rules. Consequently, the KAS in symbiosis with the data structure assists in the conception of new Web-based HCIs with correlated and transparent data semantically structured. The HCI assembled in this project aims to configure and control all electrical devices, the foundations to achieve it are already set in motion, from now on further work needs to be performed. Therefore, is important to state, this is a software to surpass boundaries, envisioned for researchers, clinics, sports, and other stakeholders. Moreover, this dissertation was compiled envision simplicity, fast understanding of the most important structures developed and its potentials. To achieve this level a lot of work was spent in simplifying what seemed difficult, reducing the content to the essential. That being said, although the software looks simple, behind it a large quantity of distinct gears were assembled, carrying innovation in different areas.

6.2 Future Work

The software presented is still in its pristine form. Therefore, in order to mature it a plethora of tasks need to be carried out:

- **Further validation:** A more exhaustive performance evaluation study is still needed with real time implementations and different case studies in Electrostimulation, and in other areas. Thus, acquiring requisites from the end user to optimize the HCI. Likewise, to improve the KAS more tests involving the use of different Schemas to create new HCIs are required.
- **Updates/Upgrades:** Constant improvement and optimizations are essential to grow the software. Design, code cleaning, implementation of other specialized GUI, are some of the updates to be executed. Furthermore, the API developed for the HCI will also need updates for the devices that need to be controlled. In the control area for each channel show the graphic session running in real time.
- **Draw editor combined with Equation editor:** The development of this fundamental mechanism, to create any kind of impulse and store it mathematically structured, is crucial to complete the HCI.
- **Real Time integration/ Biofeedback Control:** In biosignals acquisition devices, like an electrostimulator, the implementation of real time biofeedback control can assist in some applications to create a closed loop cycle where the program can learn automatically and actuate without human revision, also, it can analyse the response signals caused by the stimuli.

Bibliography

[1] F. Hayes-Roth, D. Waterman, and D. Lenat. Building expert systems. 1984.

[2] M. Negnevitsky. *Artificial intelligence: a guide to intelligent systems*. Addison-Wesley Longman, 2005.

[3] L. Kleinrock. On some principles of nomadic computing and multi-access communications. *Communications Magazine, IEEE*, 38(7):46–50, 2000.

[4] K. Lyytinen and Y. Yoo. The next wave of nomadic computing: a research agenda for information systems research. 2001.

[5] J. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury. *Feedback control of computing systems*. Wiley Online Library, 2004.

[6] M.S. Schwartz. *Biofeedback: A practitioner's guide*. The Guilford Press, 2003.

[7] H.J. Lowe, E.C. Lomax, and S.E. Polonkey. The world wide web: a review of an emerging internet-based technology for the distribution of biomedical information. *Journal of the American Medical Informatics Association*, 3(1):1–14, 1996.

[8] S. Laxminarayan and P. Yadav. Biomedical information technology: Internet and beyond. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 3, pages 1242–1243. IEEE, 1996.

[9] J.F. Jekel, D.L. Katz, and J.G. Elmore. *Epidemiologia, bioestatística e medicina preventiva; Epidemiology, biostatistics and preventive medicine*. Artmed, 2006.

[10] P.V. Prabhu. *Handbook of human-computer interaction*. North Holland, 1997.

[11] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[12] Hendler J. Berners-Lee, T. and O. Lassila. The semantic web. *Scientific American Magazine*, 2001.

[13] N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson, and M.A. Musen. Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE*, 16(2):60–71, 2001.

[14] J. Malmivuo and R. Plonsey. *Bioelectromagnetism*, volume 34. PETER PEREGRINUS LTD, 1996.

[15] M. Piccolino. Animal electricity and the birth of electrophysiology: the legacy of luigi galvani. *Brain research bulletin*, 46(5):381–407, 1998.

[16] T.A. Thrasher and M.R. Popovic. Functional electrical stimulation of walking: function, exercise and rehabilitation. In *Annales de readaptation et de medecine physique*, volume 51, pages 452–460. Elsevier, 2008.

[17] T. Marqueste, F. Messan, F. Hug, J. Laurin, E. Dousset, L. Grelot, and P. Decherchi. Effect of repetitive biphasic muscle electrostimulation training on vertical jump performances in female volleyball players. *International Journal of Sport and Health Science*, (0):1003310051, 2010.

[18] F. von Lewinski, S. Hofer, J. Kaus, K.D. Merboldt, H. Rothkegel, R. Schweizer, D. Liebetanz, J. Frahm, and W. Paulus. Efficacy of emg-triggered electrical arm stimulation in chronic hemiparetic stroke patients. *Restorative Neurology and Neuroscience*, 27(3):189–197, 2009.

[19] S.F. Cogan. Neural stimulation and recording electrodes. *Annu. Rev. Biomed. Eng.*, 10:275–309, 2008.

[20] M. Perrigot, B. Pichon, A. Peskine, and K. Vassilev. Électrostimulation et rééducation périnéale de l'incontinence urinaire et des troubles mictionnels non neurologiques. In *Annales de réadaptation et de médecine physique*, volume 51, pages 479–490. Elsevier, 2008.

[21] V.P. Lebedev, AV Malygin, AV Kovalevski, SV Rychkova, VN Sisoev, SP Kropotov, EM Krupitski, LI Gerasimova, DV Glukhov, and GP Kozlowski. Devices for noninvasive transcranial electrostimulation of the brain endorphinergic system: Application for improvement of human psycho-physiological status. *Artificial organs*, 26(3):248–251, 2002.

[22] D.F. MAYOR and M.S. MICOZZI. Energy medicine east and west: a natural history of qi (paperback). *Recherche*, 67:02, 2011.

[23] A. Le Tohic, H. Bastian, M. Pujo, P. Beslot, R. Mollard, and P. Madelenat. Effets de l'électrostimulation par veinoplus® sur les troubles circulatoires des membres inférieurs chez la femme enceinte. étude préliminaire. *Gynécologie Obstétrique & Fertilité*, 37(1):18–24, 2009.

[24] H. Motz and F. Rattay. A study of the application of the hodgkin-huxley and the frankenhaeuser-huxley model for electrostimulation of the acoustic nerve. *Neuroscience*, 18(3):699–712, 1986.

[25] M. Siff. Applications of electrostimulation in physical conditioning: a review. *The Journal of Strength & Conditioning Research*, 4(1):20, 1990.

[26] F. Brocherie, N. Babault, G. Cometti, N. Maffiuletti, and J.C. Chatard. Electrostimulation training effects on the physical performance of ice hockey players. *Medicine & Science in Sports & Exercise*, 37(3):455, 2005.

[27] Benjamin A. Suter, Timothy O'Connor, Vijay Iyer, Leopoldo T. Petreanu, Bryan M. Hooks, Taro Kiritani, Karel Svoboda, and Gordon M. G. Shepherd. Ephus: multipurpose data acquisition software for neuroscience experiments. *Frontiers in Neural Circuits*, 4:100, 2010.

[28] T. Keller, M.R. Popovic, I.P.I. Pappas, and P.Y. Müller. Transcutaneous functional electrical stimulator "compex motion". *Artificial organs*, 26(3):219–223, 2002.

[29] P.P. Breen, G.J. Corley, D.T. O'Keeffe, R. Conway, and G. ÓLaighin. A programmable and portable nmes device for drop foot correction and blood flow assist applications. *Medical engineering & physics*, 31(3):400–408, 2009.

[30] M.R. Cohen and E. Nagel. *An Introduction to Logic and Scientific Methods*. Routledge and Paul, 1963.

[31] J. Conklin. Hypertext: An introduction and survey. *Computer supported cooperative work: A book of readings*, pages 423–476, 1988.

[32] N. Bevan. Measuring usability as quality of use. *Software Quality Journal*, 4(2):115–130, 1995.

[33] A. Holzinger. Usability engineering methods for software developers. *Communications of the ACM*, 48(1):71–74, 2005.

[34] E.H. Shortliffe. Health care and the next generation internet. *Annals of internal medicine*, 129(2):138–140, 1998.

[35] A. Dix. *Human-computer interaction*. Prentice hall, 2004.

[36] P. Compton and R. Jansen. A philosophical basis for knowledge acquisition*. *Knowledge acquisition*, 2(3):241–258, 1990.

[37] I. Zelinka, Z. Oplatkova, and L. Nolle. Analytic programming–symbolic regression by means of arbitrary evolutionary algorithms. *Int. J. of Simulation, Systems, Science and Technology*, 6(9):44–56, 2005.

[38] D.R. Stoutemyer. Can the eureqa symbolic regression program, computer algebra and numerical analysis help each other? *Arxiv preprint arXiv:1203.1023*, 2012.

[39] H. Thompson et al. Xml schema. w3c working draft, may 2001, 2000.

[40] N. Klarlund, A. Møller, and M.I. Schwartzbach. Dsd: A schema language for xml. In *Proceedings of the third workshop on Formal methods in software practice*, pages 101–111. ACM, 2000.

[41] D. Crockford. The application/json media type for javascript object notation (json). 2006.

[42] K. Zyp. A json media type for describing the structure and meaning of json documents. 2011.

[43] D. Flanagan. *JavaScript: the definitive guide*. O'Reilly Media, 2006.

[44] M.F. Sanner et al. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61, 1999.

[45] F. Coito R. S. H. Ramos and M. Ortigueira. Análise de sinais em engenharia biomédica. *FCT-UNL*, 2009.

[46] J.D. Bronzino. *The biomedical engineering handbook*, volume 2. CRC Pr I Llc, 2000.

[47] J.D. Enderle and J.D. Bronzino. *Introduction to biomedical engineering*. Academic Pr, 2011.

[48] EJ Ciaccio, SM Dunn, and M. Akay. Biosignal pattern recognition and interpretation systems. *Engineering in Medicine and Biology Magazine, IEEE*, 12(3):89–95, 1993.

[49] F. Theis and A. Meyer-Base. *Biomedical Signal Analysis: Contemporary Methods and Applications*. The MIT Press, 2010.

[50] Gary S. Aston-Jones and George R. Siggins. Electrophysiology. *American College of Neuropsychopharmacology*, 2000.

[51] L. Galvani. *De viribus electricitatis in motu musculari commentarius*, volume 7. Ex Typographia Instituti Scientiarum Bologna, 1791.

[52] H. Sun and J.G. Webster. Estimating neuromuscular stimulation within the human torso with taser® stimulus. *Physics in medicine and biology*, 52:6401, 2007.

[53] J.T. Mortimer. Motor prostheses. *Comprehensive Physiology*, 1981.

[54] J. Malmivuo and R. Plonsey. *Motor prostheses. In Handbook of Physiology, Section 1: The Nervous System. Motor Control Part I*, volume 2. American Physiological Society, Bethesda, Md., 1981.

[55] E.R. Arbuthnott, IA Boyd, and KU Kalu. Ultrastructural dimensions of myelinated peripheral nerve fibres in the cat and their relation to conduction velocity. *The Journal of physiology*, 308(1):125–157, 1980.

[56] L.R. Sheffler and J. Chae. Neuromuscular electrical stimulation in neurorehabilitation. *Muscle & nerve*, 35(5):562–590, 2007.

[57] LL Baker, DR McNeal, LA Benton, BR Bowman, and RL Waters. Neuromuscular electrical stimulation: A practical guide. downey, ca: Los amigos research and education institute, 1993.

[58] D.R. McNeal. Analysis of a model for excitation of myelinated nerve. *Biomedical Engineering, IEEE Transactions on*, (4):329–337, 1976.

[59] F. Rattay. Ways to approximate current-distance relations for electrically stimulated fibers. *Journal of theoretical biology*, 125(3):339–349, 1987.

[60] A. Sedra and K.C. Smith. Microelectronic circuits, 2004.

[61] M. Akay. *Wiley encyclopedia of biomedical engineering*. Wiley-Interscience, 2006.

[62] M. Benovoy, J.R. Cooperstock, and J. Deitcher. Biosignals analysis and its application in a performance setting. In *Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing*, pages 253–258. Citeseer, 2008.

[63] S.M. Kuo, B.H. Lee, and W. Tian. *Real-Time Digital Signal Processing*. Wiley Online Library, 2006.

[64] B.J. Broderick, P.P. Breen, and G. ÓLaighin. Electronic stimulators for surface neural prosthesis. *Journal of Automatic Control*, 18(2):25–33, 2008.

[65] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. Müller. Ambient intelligence in assisted living: enable elderly people to handle future interfaces. *Universal Access in Human-Computer Interaction. Ambient Interaction*, pages 103–112, 2007.

Publications

Over the course of this project, two articles were submitted. The first publication, entitled *'Knowledge Acquisition System Based on JSON Schema. Implementation of a HCI for Actuation of Biosignals Acquisition Systems.'*, was presented in the *'9th International Conference in Wireless Information Networks and Systems'*(WINSYS 2012), held in Rome, July 2012. WINSYS is part of ICETE, the *'International Joint Conference on e-Business and Telecommunications'*. This paper was also accepted as a full paper, published in the conference proceedings, under an ISBN reference, on paper and on CD-ROM support, and will be available at the SciTePress Digital Library. The second paper, entitled *'Multi-purpose Electrostimulator Software'*, was submitted to BIODEVICES 2013 which is a co-located conference of the *'6th International Joint Conference on Biomedical Engineering Systems and Technologies'* (BIOSTEC 2013), that will be held in Barcelona in February 2013. This paper is not accepted yet, the result will be known mid November 2012.

A.1 *WINSYS 2012*

Knowledge Acquisition System Based on JSON Schema. Implementation of a HCI for Actuation of Biosignals Acquisition Systems.

Knowledge Acquisition System Based on JSON Schema
Implementation of a HCI for Actuation of Biosignals Acquisition Systems

Nuno Costa[1], Tiago Araujo[1,2], Neuza Nunes[2], Hugo Gamboa[1,2]
[1] *CEFITEC, Departamento de Física, FCT, Universidade Nova de Lisboa, Lisbon, Portugal*
[2] *PLUX - Wireless Biosignals, S.A., Lisbon, Portugal*
f_nm.costa@campus.fct.unl.pt, s_taraujo87@gmail.com

Keywords: Knowledge Acquisition System : Human Computer Interaction : Ontology : Schema Language .

Abstract: Large amounts of data, increasing every day, are stored and transferred through the internet. These data are normally weakly structured making information disperse, uncorrelated, non-transparent and difficult to access and share. Semantic Web, proposed by the World Wide Web Consortium (W3C), addresses this problem by promoting semantic structured data, like ontologies, enabling machines to perform more work involved in finding, combining, and acting upon information on the web. Pursuing this vision, a Knowledge Acquisition System was created, written in JavaScript using JavaScript Object Notation (JSON) as the data structure and JSON Schema to define that structure, enabling new ways of acquiring and storing knowledge semantically structured. A novel Human Computer Interaction framework was developed with this knowledge system, enabling the end user to, practically, configure all kinds of electrical devices and control them. With the data structured by a Schema, the software becomes robust, error – free and human readable. To show the potential of this tool, the end user can configure an electrostimulator, surpassing the specific use of many Electrophysiology software. Therefore, we provide a tool for clinical, sports and investigation where the user has the liberty to produce their own protocols by sequentially compile electrical impulses.

1 INTRODUCTION

The easiness of access, storage, transmission of data and the exponential proliferation of internet users enclose new complexities: authentication; scalable configuration management; security; huge masses of high dimensional and often weakly structured data (the main problem addressed in this project). Structuring data pursued by Semantic Web leaves many opportunities open because there is always room for improving and to develop more adequate languages. Methods and approaches to solve the problem emerged from research in Human-Computer Interaction (HCI) (Prabhu, 1997), Information Retrieval (IR), Knowledge Discovery in Databases and Data Mining (KDD) (Fayyad et al., 1996). These methods assist end users to identify, extract, visualize and understand useful information from data.

The establishment of ontologies is one of the methods to solve the problem, possible with Semantic Web, holding much promise in manipulating information in ways that are useful and meaningful to the human user. Ontologies are collections of information with specific taxonomies and inference rules to define relations between terms. A taxonomy defines classes of objects and relations among them, and inference rules provide advanced ways of relating information by deduction. Classes, subclasses and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. The structure and semantics provided by ontologies make it easier for an entrepreneur to provide a service, making its use completely transparent. Ontologies can enhance the functioning of the Web in many ways, like relating information on a Web page to the associated knowledge structures and inference rules, thus creating robust and clean applications (Berners-Lee and Lassila, 2001).

Mechanisms to specify ontologies have spring in the late years, the Schema Language is one of them. Given the fact that Xtensible Markup Language (XML) allows users to add arbitrary structure to their documents, but lacks in describing the structures, Schema was developed as a notation for defining a set of XML trees (Thompson et al., 2000). From this concept, a set of specifications were established

to create a Schema for JavaScript Object Notation (JSON), a self descriptive language for data storage and transmission, enabling the description of the data structure. A useful Schema notation must have some specific properties: identify most of the syntactic requirements that the documents in the user domain follow; allow efficient parsing; be readable to the user; concede limited transformations corresponding to the insertion of defaults; be modular and extensible to support evolving classes (Klarlund et al., 2000). This language aids in the creation of structured data and automated tools to present the data in a human readable form, making easier the extraction and visualization of useful information from data. Therefore, in this field of structuring data, Schema largely supersedes Document type Definitions (DTDs) for markup language.

One example of this procedure is Protégé, an open source Ontology Editor and Knowledge Acquisition System. Protégé is a framework written in Java and uses Swing (Java GUI widget toolkit) to create complex user interfaces. These interfaces provide the user with tools to construct domain models and knowledge-based applications with ontologies. As a graphical tool for ontology editing and knowledge acquisition, it can adapt to enable conceptual modelling with new and evolving Semantic Web languages (Noy et al., 2001). Protégé lets us, like the Schema, create domains in a conceptual level without having to know the syntax of the language ultimately used on the Web to create interfaces, passing information, between other features. We can concentrate on the concept types (integers, arrays, strings, ...) and relationships in the domain and the facts about them that we need to describe.

Using this mechanism in the project for structuring the data, a Human Computer Interaction was created with an innovative Knowledge Acquisition system for controlling the actuation of a Biosignals Acquisition System. One of the purposes, and as a practical example, is to control an electrostimulator enabling the user to create their own protocols, pursuing a non-specific software for Electrostimulation. Therefore, it allows the user to employ the software in different types of Electrostimulation applications:

- **Electrotherapy:** Rehabilitation (von Lewinski et al., 2009); Spinal cord injury, stroke, sensory deficits, and neurological disorders (Cogan, 2008); elicit electronarcosis, electrosleep and electroanalgesia (Lebedev et al., 2002);
- **Physical conditioning:** fitness; active recovery; optimizing physical performance by improvement of maximum strength of a muscle (muscular tonus) in less time (Siff, 1990).

This work presents a novel Knowledge Acquisition System based on JSON Schema with the specific focus on creating user interfaces to configure biosignals acquisition devices, and with this information control the actuation of that device. Ultimately, the software will pursue usability and acceptability, because ease of use affects the users performance and their satisfaction, while acceptably affects whether the product is used or not (Holzinger and Leitner, 2005). Our proposal defines Application Programming Interfaces (APIs) and low-level infrastructures to create a system for controlling a biosignals acquisition device, where the acquisition and actuation parameters are acquired from the user. At the core of this system is a JSON Schema enabling validation (data integrity), interaction (UI generation - forms and code) and documentation.

2 DATA STRUCTURE

2.1 JSON

JSON is a simple, lightweight and human readable text-data structure for information exchange. The approach for information exchange is simpler than XML, by the less verbose structure of the notation. Interpreting JSON is native in some languages with the existence of several support libraries that make JSON a platform independent language (Crockford, 2006). JSON structure is composed of name/value pairs separated by comma, curly brackets holds objects and square brackets holds arrays. Values can be numbers, strings, booleans, arrays, objects and null. In the example below is an object containing information of an address and phone number:

```
{"address":{
     "streetAddress": "21 2nd Street",
     "city":"New York"
 },
 "phoneNumber":
 [{
     "type":"home",
     "number":"212 555-1234"
 }]
}
```

Considering these features, JSON was selected as the data structure of this work, and is defined by JSON Schema.

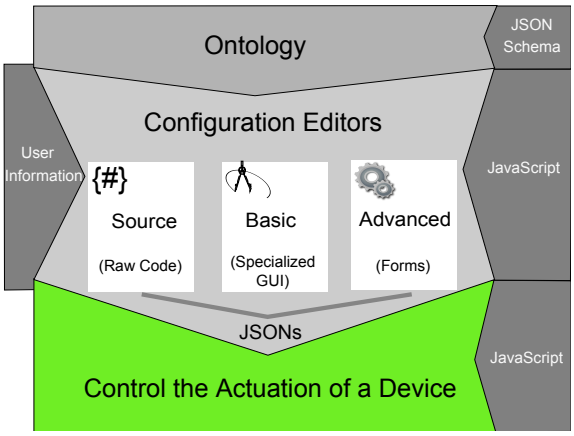

Figure 1: Generic diagram showing the flow of information.

2.2 JSON Schema

A JSON Schema is a Media Type (standard draft of options) that specifies a JSON-based format to define the structure of JSON data, providing a contract (set of rules) required in a given application, and how to interact with the contract. Accordingly, JSON Schema specifies requirements for JSON properties and other property attributes with the following intentions:

- Validation (data integrity);
- Documentation;
- Interaction (UI generation - forms and code);
- Hyperlink Navigation.

JSON Schema is also a JSON with a compact implementation and can be used on the client and server. Specifications are organized in two parts (Zyp, 2011):

- **Core Schema Specification:** primary concerned with describing a JSON structure and specifying valid elements in the structure.
- **Hyper Schema Specification:** define elements in a structure that can be interpreted as hyperlinks, in others JSON documents and elements of interaction (This allows user agents to be able to successfully navigate JSON documents based on their Schemas).

Below is an example of a JSON Schema defining the structure for the JSON example showed before:

```
{"type":"object",
 "required":false,
 "properties":{
    "address": {
      "type":"object",
      "required":true,
      "properties":{
          "city": {
              "type":"string",
              "required":true
          },
          "streetAddress": {
              "type":"string",
              "required":true
          }
        }
    },
    "phoneNumber": {
        "type":"array",
        "required":false,
        "items":{
            "type":"object",
            "required":false,
            "properties":{
               "number": {
                  "type":"string",
                  "required":false
                },
                "type": {
                  "type":"string",
                  "required":false
                }
            }
}}}}
```

As shown in the diagram, Figure 1, JSON Schema allows the definition of ontologies and will be in the core of the program, enabling: **interaction**, Schema serves as blueprint to architect the necessary forms, with or without Graphical User Interfaces (GUI), and define the other editors for the Knowledge Acquisition System; **documentation**, APIs and low-level software infrastructures in JavaScript enable the transformation of the information acquired from the user in a JSON data structure defined by the Schemas, then the data (JSONs and Schemas) is stored in the server and retrieved to the client (JavaScript) through websockets (full-duplex communications channels over a single TCP connection); **validation**, JSON data can be imputed directly in a raw editor (a knowledge acquisition system) by the end user. Nevertheless, the data is only stored if the structure agrees with the Schema.

With these mechanisms the software becomes a powerful HCI system: robust; error - free; with human readable data structures, therefore, visualization and extraction of information is easier.

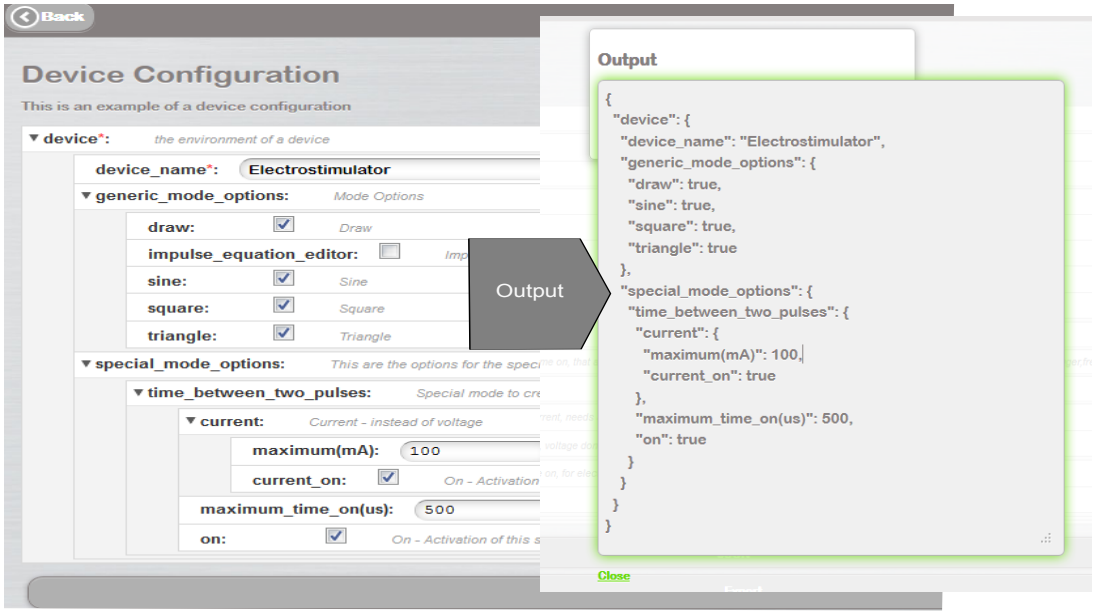

Figure 2: Configuration in Advanced Editor the mode options for an electrostimulator. Initial version of the software.

3 HUMAN COMPUTER INTERACTION

Based on the method for data structure, described in the last section, a Human Computer Interaction was developed with a novel Knowledge Acquisition system for controlling the actuation of Biosignals Acquisition Systems. As shown in Figure 1, an ontology is defined with the help of JSON Schemas, then these Schemas are interpreted in JavaScript enabling the conception of three editors. These editors provide different means to acquire the information from the user. Afterwards, this information is saved as JSONs in the server. In the control, the JSONs are retrieved from the server, through websockets, and parsed in JavaScript. Finally, the necessary information is sent to the device via APIs. As a practical example, the HCI provides mechanisms to control an electrostimulator by enabling the user to configure protocols/sessions for actuation of the device.

The HCI is integrated in a software for biosignals acquisition and processing from PLUX - Wireless Biosignals, S.A., enabling the control of a generic device. In this way, actuation, acquisition and processing is possible in a closed-loop cycle (Broderick et al., 2008).

The main purpose of this project is to provide the end user the liberty to create and sequentially compile electrical impulses. This enables the user to create actuation sessions for electrical devices. In a particular

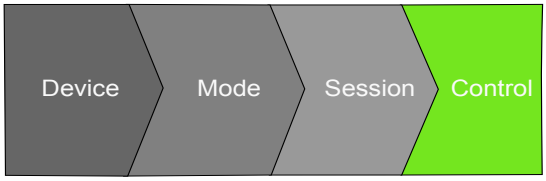

Figure 3: Diagram representing the direction to configure then control. First, configure a new device, then modes and finally the sessions using the modes created.

example, enables the user to configure an electrostimulator and design sessions to test different types of Electrostimulation protocols.

The software is divided in configuration (the knowledge acquisition system based on JSON Schemas), and control for the actuation of a device. Basically, as shown in Figure 3, to control we need to configure a device, and create temporal sessions, i.e sequence of impulse modes, for each channel of that device. Consequently, the configuration is divided in device, mode and session, each one with the respective JSON Schema to structure the data, and directly or indirectly create the editors. In the next sections,

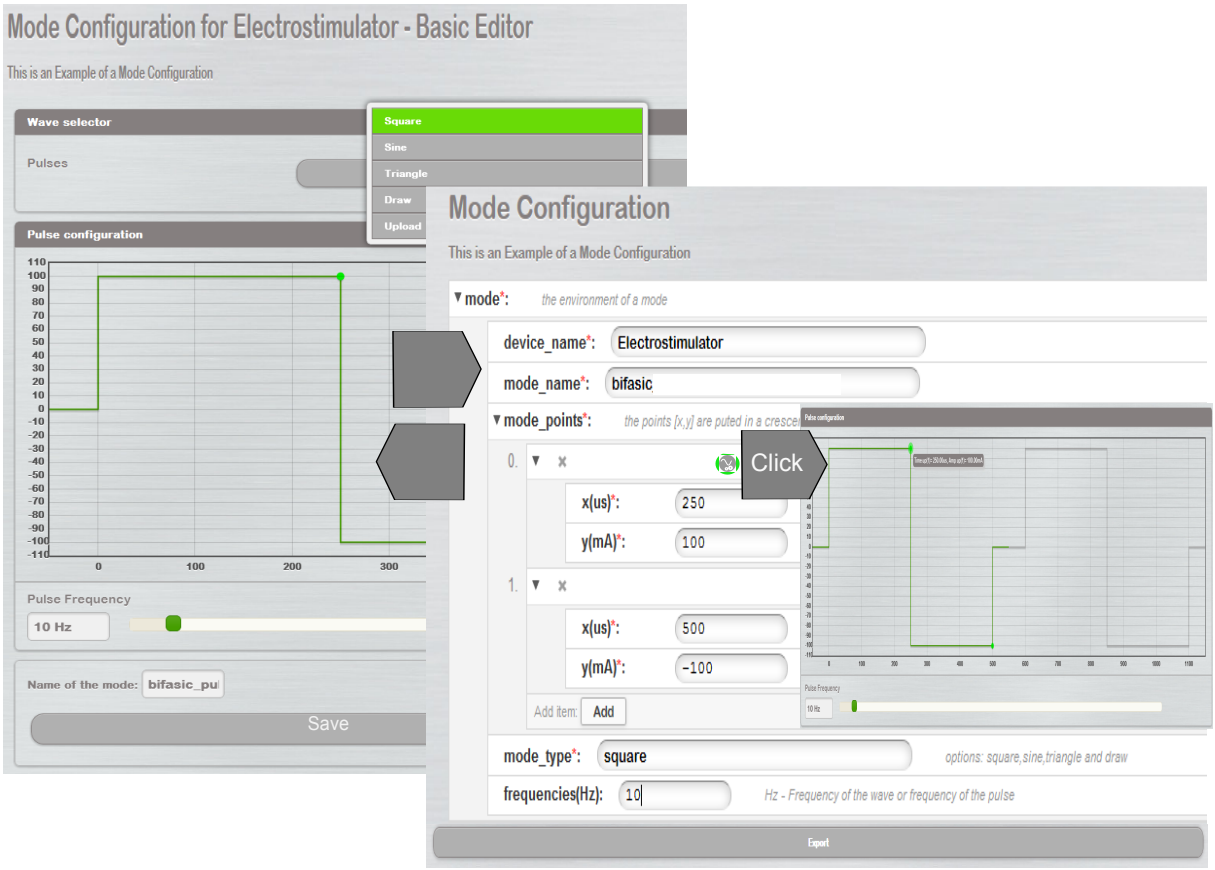

Figure 4: Configuration of the mode in Basic Editor and Advanced Editor. Initial versions of the software.

these ideas are explained in more detail accompanied with images for the specific case of an electrostimulator.

3.1 Configuration

3.1.1 Device

The user can compose a new device by setting the mode options for that device. There are two types of options, generic and special. Special stands for special modes, in this case the special mode enables the creation of pulses instead of only periodic waves, unlike a generic wave generator. For example, the special mode is necessary to configure an electrostimulator. We can see in Figure 2 the configuration with the special parameters for Electrostimulation: maximum current, 100 mA, and maximum time on, 500 µs. After fill in the form, created directly from the Schema, a JSON is exported and stored in the server.

3.1.2 Mode

The user can project a new mode for a specific device. The mode permits the configuration of pulses (special mode options activated), and waves (special mode options deactivated) depending on the mode options. In Figure 4 is possible to view how a mode can be configured in two different editors (described in the editors section). The figure shows the creation of bifasic mode for the electrostimulator, a pulse with positive and negative time on of 250 µs, amplitude up 100 mA and amplitude down -100 mA, and 10 Hz of pulse frequency.

3.1.3 Session

The user can create a new session for a specific device. A session is a sequence of modes, and each channel of a device can have one session programmed. A device has a maximum number of programmable modes, but they can be repeated infinitely. Figure 5 shows the configuration of a session. Primarily, the definition of the modes that will be used, four

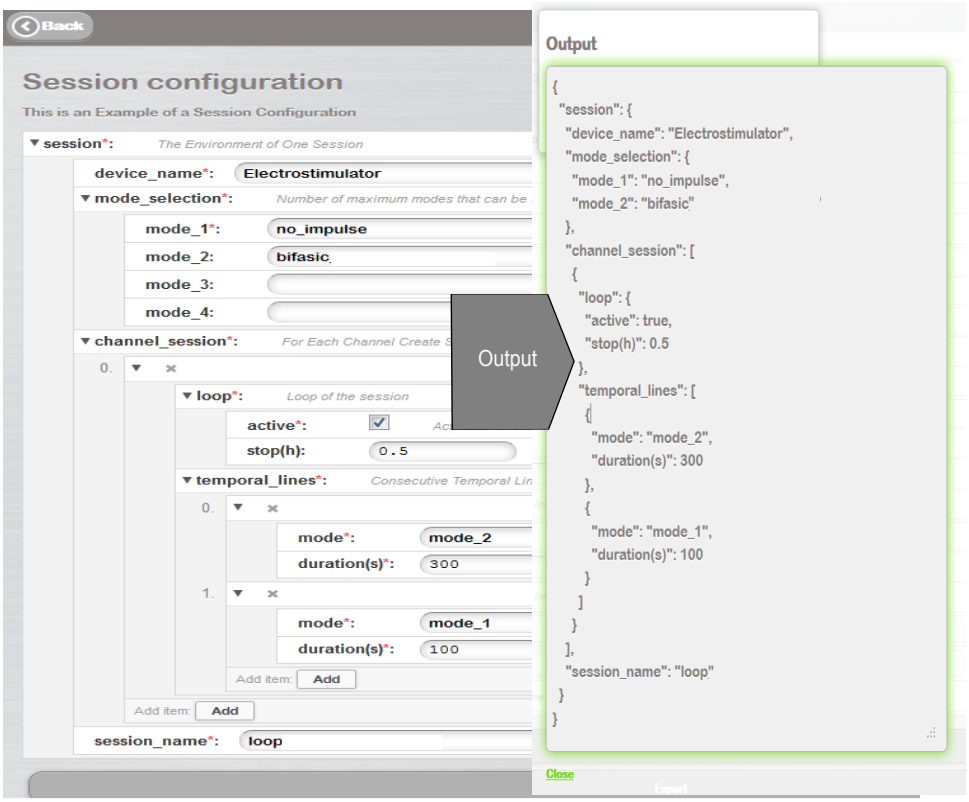

Figure 5: Configuration of a session in Advanced Editor. Loop during half an hour with bifasic mode, 300 seconds activated, and no_impulse 100 seconds activated. Initial version of the software.

is the maximum that the hardware can memorize. Afterwards, the creation of a session for the channels required, in this case, only one channel is programmed with two modes, bifasic, 300 seconds activated and, no_impulse, during 100 seconds. This sequence is repeated during half an hour by activating the loop for that specific session. The session is named protocol1.

3.1.4 Editors

The core for each configuration stated before is a Schema, and the end user has the possibility to choose between three editors (like in Figure 1):

- Basic Editor (Specialized GUI): simple and user-friendly interface with the minimum required fields to fill. The information is saved in a JSON format defined by the Schema. This editor is intended for the basic users. It can be seen in Figure 4.
- Advanced Editor: a form generated automatically from the Schema specifications, with the possibility of edition in GUI. Information is also saved in JSON format defined by the Schema. The most profitable editor because it shows directly how the information will be saved in the JSON. This way, users understand the architecture of the data. Figures 2, 4 and 5 have the configuration of a device, a mode and a session, respectively, in the Advanced Editor.
- Source Editor: an editor to upload or create JSONs, where the user writes the JSON (like the JSONs exported in figures 2 and 5) or copy and paste the JSON in the editor. JSONs are valid and can be saved if they agree with JSON Schema, if not, reports with errors are generated. This Editor is intended for users with more knowledge of the data structure, or for the ones that just want to copy and paste information or upload.

  Note that it's possible to travel between the three editors, and the information is inherited between them. Also, in Advanced Editor, GUIs are directly related with special types from the JSON Schema, and can be used to fill the necessary fields. Like in the example Figure 4, clicking on the specific button opens a graphic tool (canvas) for edition of points by moving them with the mouse, identical to the canvas from Basic Editor.

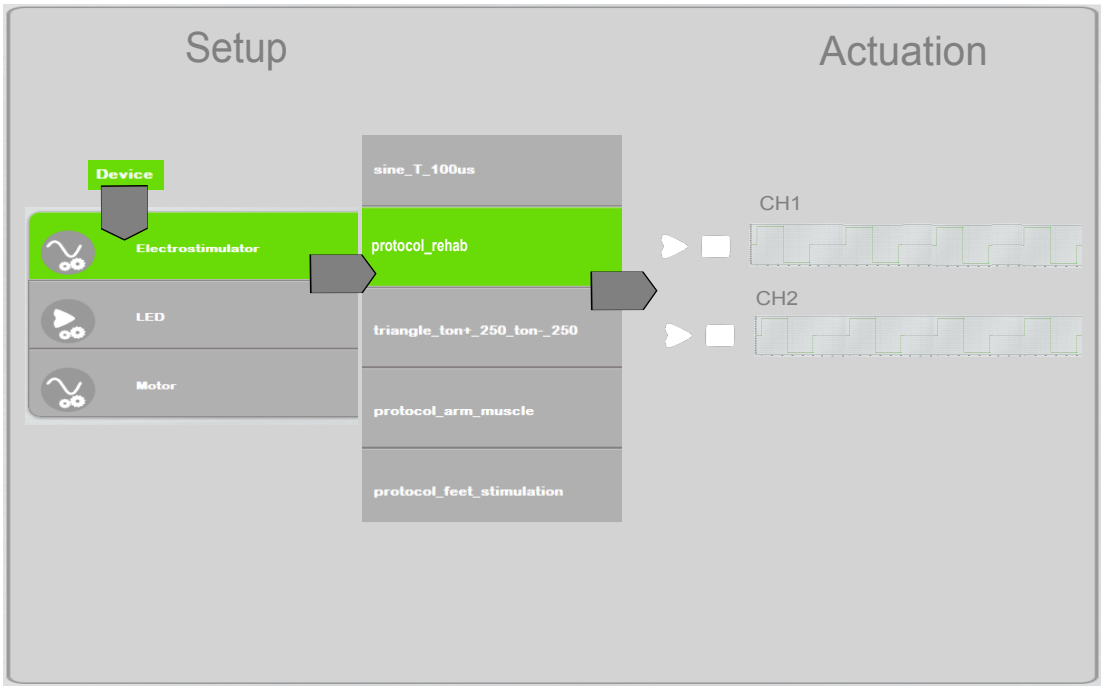

Figure 6: Actuation of an electrostimulator with the session protocol_rehab. This session has two channels with different sessions configured.

3.2 Control

The control is the area for actuation and acquisition of biosignals. In here, the user first needs to set up the device, for that he must choose the device and the session to program. When the device is ready, start/stop of the session is feasible. In the course of the session is possible to acquire real time biosignals and synchronize them with the session. This part is in development stage: Figure 6 presents a moke-up what could be in the final stage. The control area will have many device sessions templates. Therefore, a quick approach by the user will enable the simple selection of a pre-defined session and start the actuation immediately.

4 CONCLUSIONS

This project envisions a software with powerful capabilities. The most important parts are the infrastructures that conduct the flow of information between the Schema, the editors, the GUIs and finally the process of saving in JSON data. After the APIs and low-level infrastructures are programmed is easy and fast to create information editors with other Schemas, develop configuration interfaces for that editors and allow the engineering of new human computer interactions. This explains the impressive usability of this knowledge acquisition system. For this reason, is important to refute that the main mechanisms behind this software are JSON, the human-readable data structure, and JSON Schema that defines the structure of JSON, allowing interaction (creation of forms with specialized GUIs), documentation and validation, producing an error-free, semantically structured and human-readable knowledge acquisition system, contributing to the HCI system robustness.

In Biomedicine, the configuration part should only be manipulated by clinical professionals, where they can adapt sessions to each patient. Overly complicated user interfaces and large, bulky designs can deter patients from using the device on a day to day basis, so, ultimately, the patient will only start or stop a session prepared for him. Following this idea the software was separated in configuration, where clinical professionals can configure a device with sessions for each channel, and control, where the user just need to choose the device and session (set up the device) then start the session. If for some reason, the session should not stop automatically (before the stop time is reached), the user can stop the session. This HCI is being tested in the Electrostimulation area to support clinical professionals, sportsman and investigators to control electrostimulators by enabling the creation of different kind of protocols, empowering the software usability.

This software, above all, pursues usability, for this

reason, in future stages of the project, the following tasks will be conducted: user studies, extended unit tests, usability tests, and usability expert evaluation.

REFERENCES

Berners-Lee, T., H. J. and Lassila, O. (2001). The semantic web. *Scientific American Magazine*.

Broderick, B., Breen, P., and ÓLaighin, G. (2008). Electronic stimulators for surface neural prosthesis. *Journal of Automatic Control*, 18(2):25–33.

Cogan, S. (2008). Neural stimulation and recording electrodes. *Annu. Rev. Biomed. Eng.*, 10:275–309.

Crockford, D. (2006). The application/json media type for javascript object notation (json).

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Holzinger, A. and Leitner, H. (2005). Lessons from real-life usability engineering in hospital: from software usability to total workplace usability. *Empowering Software Quality: How can Usability Engineering reach these goals*, pages 153–160.

Klarlund, N., Møller, A., and Schwartzbach, M. (2000). Dsd: A schema language for xml. In *Proceedings of the third workshop on Formal methods in software practice*, pages 101–111. ACM.

Lebedev, V., Malygin, A., Kovalevski, A., Rychkova, S., Sisoev, V., Kropotov, S., Krupitski, E., Gerasimova, L., Glukhov, D., and Kozlowski, G. (2002). Devices for noninvasive transcranial electrostimulation of the brain endorphinergic system: Application for improvement of human psycho-physiological status. *Artificial organs*, 26(3):248–251.

Noy, N., Sintek, M., Decker, S., Crubézy, M., Fergerson, R., and Musen, M. (2001). Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE*, 16(2):60–71.

Prabhu, P. (1997). *Handbook of human-computer interaction*. North Holland.

Siff, M. (1990). Applications of electrostimulation in physical conditioning: a review. *The Journal of Strength & Conditioning Research*, 4(1):20.

Thompson, H. et al. (2000). Xml schema. w3c working draft, may 2001.

von Lewinski, F., Hofer, S., Kaus, J., Merboldt, K., Rothkegel, H., Schweizer, R., Liebetanz, D., Frahm, J., and Paulus, W. (2009). Efficacy of emg-triggered electrical arm stimulation in chronic hemiparetic stroke patients. *Restorative Neurology and Neuroscience*, 27(3):189–197.

Zyp, K. (2011). A json media type for describing the structure and meaning of json documents.

A.2 *Biodevices 2013*

Multi-purpose Electrostimulator Software.

Multi-purpose Electrostimulator Software

Nuno Costa[1], Tiago Araujo[1,2], Neuza Nunes[2], Hugo Gamboa[1,2]
[1] *CEFITEC, Departamento de Física, FCT, Universidade Nova de Lisboa, Lisbon, Portugal*
[2] *PLUX - Wireless Biosignals, S.A., Lisbon, Portugal*
f_nm.costa@campus.fct.unl.pt, s_taraujo87@gmail.com

Keywords: Human Computer Interaction : Electrostimulation : Multi-purpose Software.

Abstract: Nowadays, most of the software for electrostimulation is made with specific purposes, and in some cases they have complicated user interfaces and large, bulky designs that deter usability and acceptability. A novel Human Computer Interaction framework was developed enabling the end user to configure and control an electrostimulator, surpassing the specific use of several electrostimulator software. In the configuration the user is able to compile different types of electrical impulses (modes) in a temporal session, and this session can be actuated in the control. To help the user in creating any type of protocol (session) we devised three standard impulse generator (rectangular, sin and triangular) and a new way of creating electrical impulses by drawing, then fitting this data and process it with a mathematical algorithm for finding simple equations to describe the data. With it, the user has the possibility to choose the best equation that fits the draw and store it mathematically structured, thus adding not only a draw editor, but also an equation editor. Therefore, we provide a tool for clinical, sports and investigation where the user is free to produce their own protocols by sequentially compile electrical impulses.

1 INTRODUCTION

In 1791, scientist Luigi Galvani first showed that electricity, when applied to a frog's leg, could cause muscle twitches (Piccolino, 1998). In the intervening centuries, scientists learned much about how Electrical Stimulation or Electrostimulation (ES) affects muscle tissue and have tried to apply that knowledge to muscles paralysed by neuromuscular disease to create both therapeutic and functional effect. This field of medicine is known variously as Electrical Muscle Stimulation or Electromyostimulation (EMS), Neuromuscular Electrical Stimulation (NMES) and Functional Electrical Stimulation (FES) electromyostimulation, consisting in nerve manipulation through electrical pulses aiming muscular contraction or sensory response for various applications (Malmivuo and Plonsey, 1996). Nowadays, the applications can be divided in two main areas:

- **Electrotherapy:** Rehabilitation (von Lewinski et al., 2009); Spinal cord injury, stroke, sensory deficits, and neurological disorders (with Neural prostheses) (Cogan, 2008); Urinary incontinence (Perrigot et al., 2008); Transcranial Electrical Stimulation (TES) as a method to elicit electronarcosis, electrosleep and electroanalgesia (for pain relief) (Lebedev et al., 2002)(MAYOR and MICOZZI, 2011); Treatment of lower limbs venous insufficiency related symptoms in pregnant women (Le Tohic et al., 2009); electrostimulation of the acoustic nerve profoundly deaf patients can, in the best cases, reach almost complete speech understanding without lip reading (Motz and Rattay, 1986); and many others.
- **Physical conditioning:** fitness; active recovery; optimizing physical performance by improvement of maximum strength of a muscle (muscular tonus) in less time (Siff, 1990)(Marqueste et al., 2010) (Brocherie et al., 2005) .

Although ES may hold much promise there are many technical challenges that need to be surpassed. Commercial software solutions for electrostimulators grow every day, but these are often limited by a variety of factors including cost, source code inaccessibility, hardware compatibility, and more (Breen et al., 2009)(Keller et al., 2002). Consequently, a strong tradition in neurophysiology research is to write custom software routines. While superb for the specific tasks at hand, these custom solutions rarely offer the flexibility and extensibility needed for them to be transferable across platforms, hardware configurations, and experimental paradigms without significant modifica-

tions. Therefore, in present time, software/hardware solutions to provide a device with a multi-purpose platform (sport, therapy or investigation), a dynamic software, which enables the user to create their own protocols, is needed (Suter et al., 2010)(Prochazka et al., 1997). To contribute in solving this necessity, we created a WEB based Human Computer Interaction (HCI) that allows the user to employ the software in different types of ES applications, pursuing a non-specific software.

This work further depicts a novel HCI based on a Knowledge Acquisition System developed previously (Costa et al., 2012). The specific focus of this tool is to compile electrical impulses in different sessions for actuation of an electrostimulator, enabling the user to design different types of protocols. To surpass this challenge, innovative ways of compile and create electrical impulses were implemented.

Ultimately, the software will pursue usability and acceptability because ease of use affects the users performance and their satisfaction, while acceptably affects whether the product is used or not (Holzinger and Leitner, 2005).

The next sections are divided in: Human Computer Interaction, a resume of the structure developed in a previous study; Configuration, a description of the methods provided to the user to compile electrical impulses; Control, a brief description of the simplicity in actuate a session.

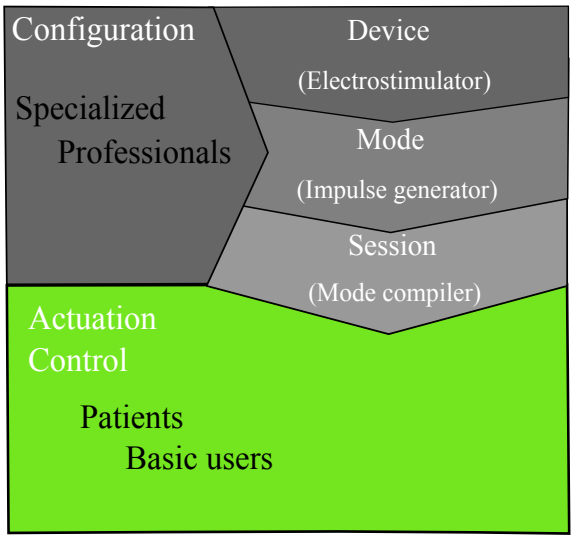

Figure 1: Generic diagram of the HCI.

2 HUMAN COMPUTER INTERACTION

A novel Human Computer Interaction was developed with the intention of controlling the actuation of electrical devices, like Biosignals Acquisition Systems, LEDs, motors, between others. Nevertheless, to show the potential of the tool, the biggest and the first challenge that led us to idealize it, was to control an electrostimulator for different types of applications. To demonstrate it, this article shows the capabilities of the software in controlling an electrostimulator and the contribution for Electrostimulation.

The HCI is integrated in a WEB based software for biosignals acquisition and processing from PLUX - Wireless Biosignals, S.A., enabling the control of a generic device. In this way, actuation, acquisition and processing is possible in a closed-loop cycle (Broderick et al., 2008).

The main purpose of this project is to provide the end user the liberty to create and sequentially compile electrical impulses. This enables the user to create actuation sessions for electrical devices. In this case, enables the user to configure an electrostimulator and design sessions to test different types of Electrostimulation protocols.

In Biomedicine, the configuration of a device should only be performed by clinical professionals, where they can adapt sessions to each patient. Overly complicated user interfaces and large, bulky designs can deter patients from using the device on a day to day basis, so, ultimately, the patient will only start or stop a session prepared for him. Following this idea the software was divided in configuration, where clinical professionals can configure an electrostimulator with sessions for each channel, and control, where the user just needs to choose the device and session previously configured, then start the session. If for some reason, the session should not stop automatically (before the stop time is reached), the user can stop the session. Basically, as shown in Figure 1, to control we need to configure the electrostimulator, and create temporal sessions, i.e. sequence of impulse modes, for each channel of that device. Consequently, the configuration is hierarchically structured in device, mode and session. In the control we only need to choose the device, then the session for that device that will be actuated, and next start the session. The stop can be manual or automatic (end of the session). This mechanism makes the control very simple.

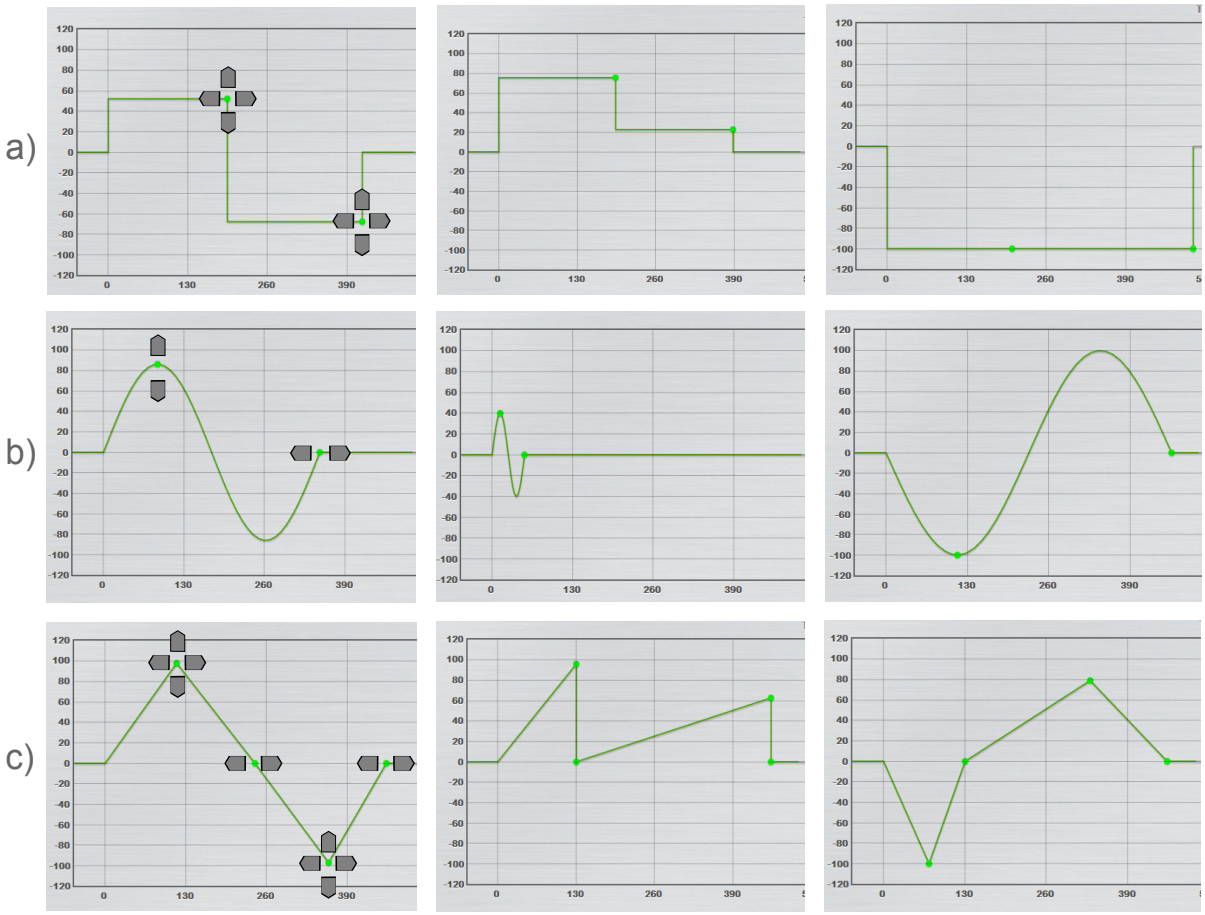

Figure 2: The default impulses: a)square, b)sine and c)triangle. The arrows mark the direction in which the interactive points can be moved.

3 CONFIGURATION

3.1 Device Requisites

In the case of an electrostimulator, some specific options need to be satisfied. The specifications for the potentials and limits of ES is dependent of the application and the electrostimulator. The requisites of the electrostimulator we used for this project, are:

- Pulse amplitude:
  0-100 mA (1 mA step)
- Pulse Width:
  0-500 µs (5 µs step)
- Pulse frequency:
  1-200 Hz (1 Hz step)
- Number of channels:
  2
- Number of modes:
  4
- Waveform type:
  Rectangular, triangular, sin, customized waveform (constant potential with no offset)

3.2 Mode

After the specification of the device requisites the user can project a new mode for a specific device. The mode permits four types of impulse configuration: square, rectangular pulse that can be changed within the data limits; sine, sin pulse that can be changed within the data limits; triangle, triangular pulse that can be changed within the data limits; draw, a novel and innovative impulse generator. The user is also able to choose the frequency (impulse per second) by moving a slider. In the three default types square, sine and triangle, the end user have to click and pull the interactive points (as it can be seen in Figure 2): square have two interactive points that can be moved along x an y axis within the data limits; sine also have two, one for the amplitude and other for the period; triangle have four with two points that just move along

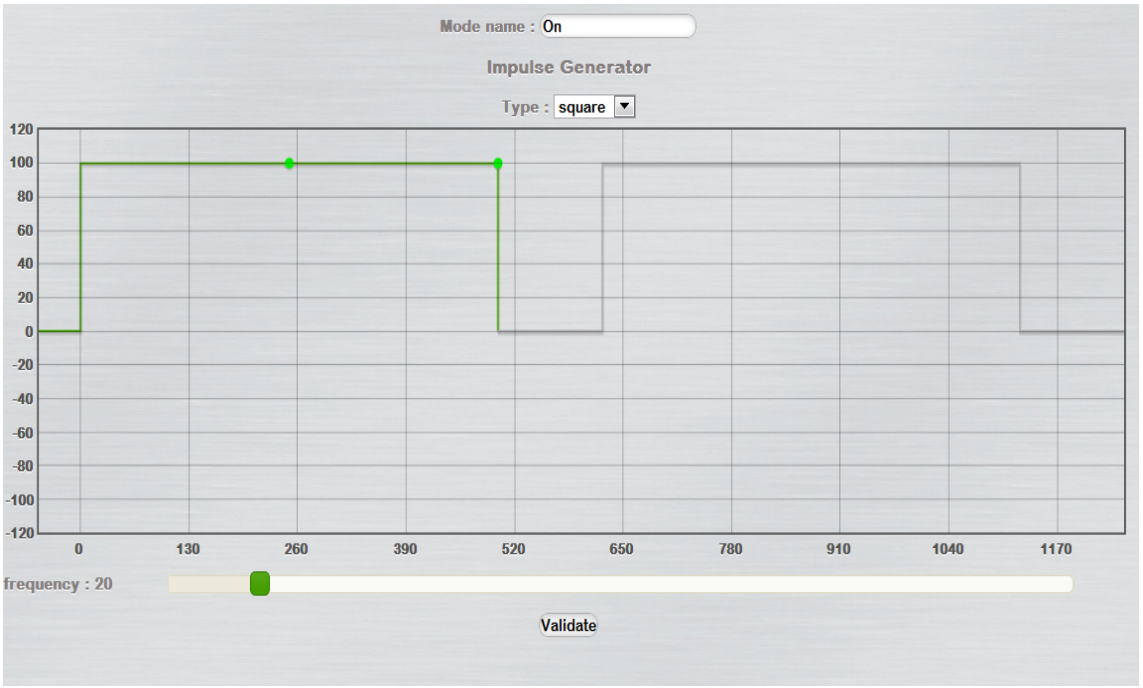

Figure 3: Configuration of the mode "On" for the Electrostimulator. Initial version of the software.

the x axis. These types allow the end user to create a lot of different impulses, but to be able of creating any type we had to devise a new form of impulse generation, the draw method. In this new type, the user has the possibility to draw an impulse within the data limits, from this draw we retrieve 100 points that will be processed by an algorithm. The idea behind this algorithm it is to search within the 100 points that seem connected to each other, then propose a series of simple equations to describe the links. The best are selected, tweaked, and again tested against the data. Next, the algorithm repeats the cycle over and over, until it finds equations that have a good probability of modulate the data. Then, by choosing one of the equations, the user can see a graph with 100 points based in the equation and compare it to the drawing. If it fits the user objectives, save the equation, if not, the user can tweak the equation manually, and see the effects, or just save the 100 points from the draw. This algorithm will have large benefits when finished, because it provides a mechanism of storing the draw data mathematically structured, in this way, the user will have the possibility of re-editing the data by changing some parameters in the stored equations. So, instead of just providing a draw method to create impulses, an equation method is also available to the user for creating impulses, very important to simplify the impulse generation when the impulse can be described by an equation and only few parameters need to be changed.

This algorithm is still in the initial stage and is based in the work of Schmidt and Lipson, the Eureqa, a highly praised symbolic regression program (Stoutemyer, 2012).

In Figure 3 is possible to view an example of how a mode can be configured within the data limits for x axis 0 - 500 µs and for y axis -100 - 100 mA . The figure shows the creation of "On" mode for the electrostimulator, a rectangular pulse with 500 µs, amplitude 100 mA, and 20 Hz of pulse frequency, i.e. 20 pulses per second.

3.3 Session

The user can create a new session for a specific device. A session is a sequence of modes, and each channel of a device can have one session programmed. A device has a maximum number of programmable modes, but they can be repeated infinitely. This mechanism is very important to program stimulation protocols, automating the control. In our electrostimulator two channels can be programmed with four modes: first we define the modes that will be used, then we add channel sessions. In each channel session we have to define if the session has loop, then we need to sequentially compile the modes. Figure 4 shows an example of session configuration. Primarily, the definition of the modes that will be used, four is the maximum that the hardware can memorize. Afterwards, the creation of a session for the channels re-

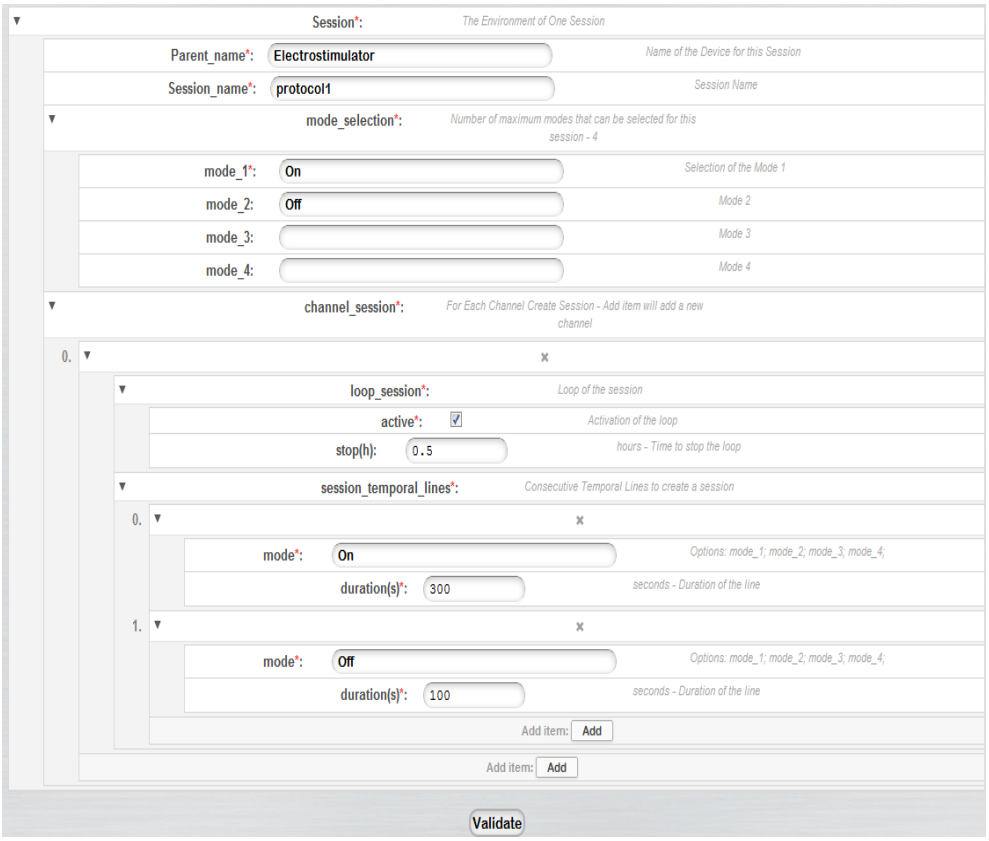

Figure 4: Configuration of one channel session with the modes "On" and "Off". Initial version of the software.

quired, in this case, only one channel is programmed with two modes: On, 300 seconds activated and, Off, during 100 seconds (Off is a pulse with 500 µs and amplitude 0 mA). This sequence is repeated during half an hour by activating the loop for that specific session.

4 CONTROL

The control is the area for actuation and acquisition of biosignals. In here, the user first needs to choose the device, in our case the electrostimulator, and the session to program. When the device is ready, start/stop of the session is feasible. In the course of the session is possible to acquire real time biosignals and synchronize them with the session. Also, note that in the background, an API (Application Programming Interface) makes the bridge between the software and the hardware, parsing the high level programming language to machine language enabling the control of the hardware. Trough this API we set up the hardware with the necessary for the protocols of stimulation, and start/stop the session.

Figure 5 presents the control diagram. The control area will have many device sessions templates. Therefore, a quick approach by the user will enable the simple selection of a pre-defined session and start the actuation immediately.

5 CONCLUSIONS

A strong tradition in neurophysiology research is to write custom software routines. These software are specific and rarely offer the flexibility and extensibility needed for them to be transferable across platforms, hardware configurations, and experimental paradigms without significant modifications. Also, most of the times these software have confuse, large and bulky interfaces that can deter users from using the software. Electrostimulation is one of the areas that suffers from these problems. For these reasons, and pursuing usability and acceptability, we are developing a HCI that has a great potential to be transferable across different types of electrical device control, and to demonstrate it, we are able to produce different types of sessions to control the actuation of an electrostimulator. For this purpose we provide mech-

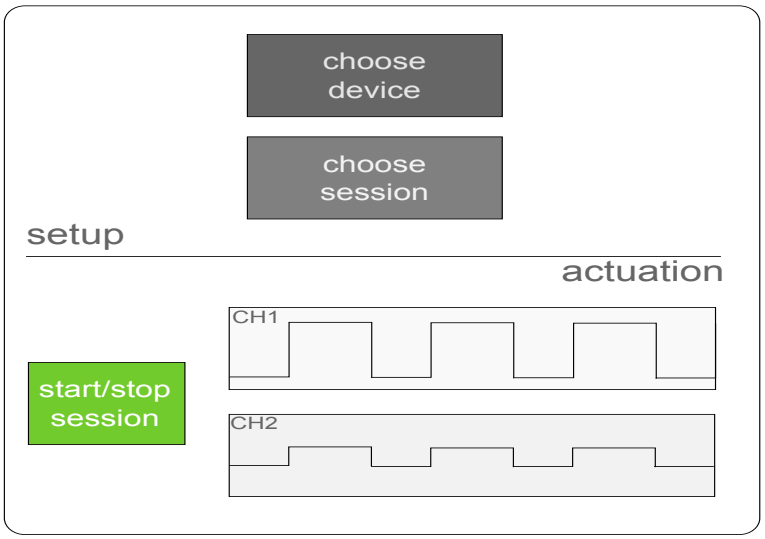

Figure 5: Diagram for control

anisms to configure and control a device. The most important mechanisms are: the separation in configuration and control, where clinical professionals can configure a device with sessions for each channel, and control, where the user just needs to choose the device and session (set up the device) then start the session; the sequence of configuration (Device, Mode, Sessions), that supplies the user with a structured configuration; the tools in Mode, enabling the end user to shape the impulses at is own desire, and with the draw, plus equation algorithm, implemented the user will be able to produce any kind of impulse and store information mathematically structured; the compilation of modes in Session allowing the creation of temporal sessions for the actuation of an electrostimulator, i.e. permits to program stimulation protocols. Thus, creating a dynamic, flexible way of automate the actuation of stimuli.

This software, above all, pursues usability, for this reason, in future stages of the project, the following tasks will be conducted: user studies, extended unit tests, usability tests, and usability expert evaluation. Also, the draw algorithm needs to be further developed to make the software even more dynamic and extensible.

With this powerful tool we provide a user-friendly software solution with a multi-purpose platform for sport, therapy and research.

REFERENCES

Breen, P., Corley, G., O'Keeffe, D., Conway, R., and ÓLaighin, G. (2009). A programmable and portable nmes device for drop foot correction and blood flow assist applications. *Medical engineering & physics*, 31(3):400–408.

Brocherie, F., Babault, N., Cometti, G., Maffiuletti, N., and Chatard, J. (2005). Electrostimulation training effects on the physical performance of ice hockey players. *Medicine & Science in Sports & Exercise*, 37(3):455.

Broderick, B., Breen, P., and ÓLaighin, G. (2008). Electronic stimulators for surface neural prosthesis. *Journal of Automatic Control*, 18(2):25–33.

Cogan, S. (2008). Neural stimulation and recording electrodes. *Annu. Rev. Biomed. Eng.*, 10:275–309.

Costa, N., Araujo, T., Nunnes, N., and Gamboa, H. (2012). Knowledge acquisition system based on json schema. implementation of a hci for actuation of biosignals acquisition systems.

Holzinger, A. and Leitner, H. (2005). Lessons from real-life usability engineering in hospital: from software usability to total workplace usability. *Empowering Software Quality: How can Usability Engineering reach these goals*, pages 153–160.

Keller, T., Popovic, M., Pappas, I., and Müller, P. (2002). Transcutaneous functional electrical stimulator "compex motion". *Artificial organs*, 26(3):219–223.

Le Tohic, A., Bastian, H., Pujo, M., Beslot, P., Mollard, R., and Madelenat, P. (2009). Effets de l'électrostimulation par veinoplus® sur les troubles circulatoires des membres inférieurs chez la femme enceinte. étude préliminaire. *Gynécologie Obstétrique & Fertilité*, 37(1):18–24.

Lebedev, V., Malygin, A., Kovalevski, A., Rychkova, S., Sisoev, V., Kropotov, S., Krupitski, E., Gerasimova, L., Glukhov, D., and Kozlowski, G. (2002). Devices for noninvasive transcranial electrostimulation of the brain endorphinergic system: Application for improvement of human psycho-physiological status. *Artificial organs*, 26(3):248–251.

Malmivuo, J. and Plonsey, R. (1996). *Bioelectromagnetism*, volume 34. PETER PEREGRINUS LTD.

Marqueste, T., Messan, F., Hug, F., Laurin, J., Dousset, E., Grelot, L., and Decherchi, P. (2010). Effect of repetitive biphasic muscle electrostimulation training on vertical jump performances in female volleyball players. *International Journal of Sport and Health Science*, (0):1003310051.

MAYOR, D. and MICOZZI, M. (2011). Energy medicine east and west: a natural history of qi (paperback). *Recherche*, 67:02.

Motz, H. and Rattay, F. (1986). A study of the application of the hodgkin-huxley and the frankenhaeuser-huxley model for electrostimulation of the acoustic nerve. *Neuroscience*, 18(3):699–712.

Perrigot, M., Pichon, B., Peskine, A., and Vassilev, K. (2008). Électrostimulation et rééducation périnéale de l'incontinence urinaire et des troubles mictionnels non neurologiques. In *Annales de réadaptation et de médecine physique*, volume 51, pages 479–490. Elsevier.

Piccolino, M. (1998). Animal electricity and the birth of electrophysiology: the legacy of luigi galvani. *Brain research bulletin*, 46(5):381–407.

Prochazka, A., Gauthier, M., Wieler, M., and Kenwell, Z. (1997). The bionic glove: an electrical stimulator garment that provides controlled grasp and hand opening in quadriplegia. *Archives of physical medicine and rehabilitation*, 78(6):608–614.

Siff, M. (1990). Applications of electrostimulation in physical conditioning: a review. *The Journal of Strength & Conditioning Research*, 4(1):20.

Stoutemyer, D. (2012). Can the eureqa symbolic regression program, computer algebra and numerical analysis help each other? *Arxiv preprint arXiv:1203.1023*.

Suter, B. A., O'Connor, T., Iyer, V., Petreanu, L. T., Hooks, B. M., Kiritani, T., Svoboda, K., and Shepherd, G. M. G. (2010). Ephus: multipurpose data acquisition software for neuroscience experiments. *Frontiers in Neural Circuits*, 4:100.

von Lewinski, F., Hofer, S., Kaus, J., Merboldt, K., Rothkegel, H., Schweizer, R., Liebetanz, D., Frahm, J., and Paulus, W. (2009). Efficacy of emg-triggered electrical arm stimulation in chronic hemiparetic stroke patients. *Restorative Neurology and Neuroscience*, 27(3):189–197.

B

Configuration Environment

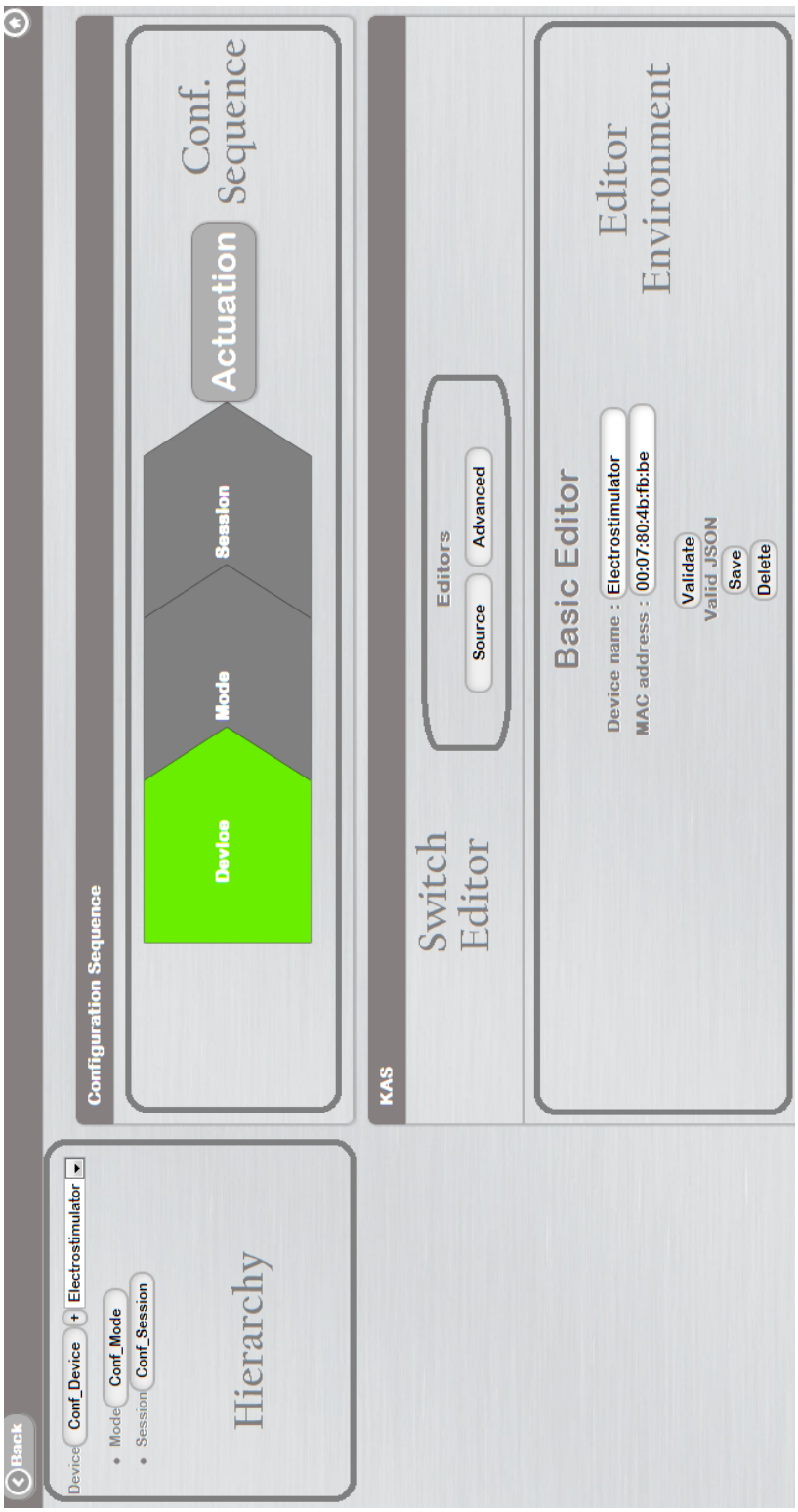

Figure B.1: Configuration Environment.