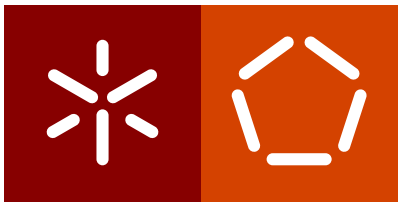
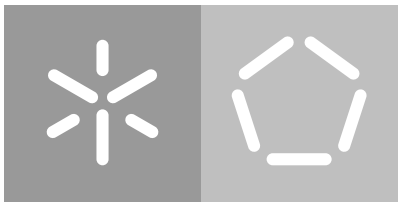


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Tiago Alexandre Fernandes de Lima

Development of intuitive graphical applications for interaction with a smart wearable orthotic system

December 2018



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Tiago Alexandre Fernandes de Lima

Development of intuitive graphical applications for interaction with a smart wearable orthotic system

Master dissertation
Master Degree in Informatics Engineering

Dissertation supervised by
Cristina Manuela Peixoto Santos

December 2018

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Professor Cristina Manuela Peixoto dos Santos for the opportunity to work on the present project and for the recurring valuable feedback that guided and improved my work.

Additionally, I would like to convey my appreciation for all the members that participated in the SmartOs project. I would like to thank Paulo Félix, Pedro Fernandes and Joana Figueiredo for their availability, patience and incessant support to this work.

I would also like to express my thanks for the valuable technical input and motivation received from my friends.

Last but not least, I would like to thank my family: my parents and my brother for their unbroken support and encouragement throughout this work and my academic progress.

ABSTRACT

The ideal path to treatment might not always be easily extrapolated from the work with previous patients, this could implicate a situation where physical rehabilitation could demand experimentation, traditionally made through observational data which might be costly and less precise.

With this in mind SmartOs, a multi-disciplinary project that consists on the creation of a smart wearable orthotic system was started. SmartOs is a modular system whose main goal is to help people with physical, or more specifically, gait impairments, this tool incorporates an array of sensors as well as powered orthotic devices to provide monitoring and assistance capabilitites.

To ensure that the system can be integrated in a clinical context, tools with which to control and monitor the system are needed, these were not yet developed for the SmartOs, as interaction required resorting to technical usage of the low-level interfaces of the system. As such the development of these tools will be the focus of this dissertation.

An Android application was developed to provide intuitive control over SmartOs's parameters, also ensuring that therapy sessions are configured without functional compromise for the user or the system.

In the scope of this dissertation, a desktop application was also designed, developed and written in Qt to monitor multiple parameters in real time or at a further date, through the usage of plots and other graphical components in a dashboard design tailored to the user. Additionally basic management of therapy sessions and the establishment of a standardized format with which to marshal local session data and record therapies to local files was implemented.

The capability and correctness of the applications were validated throughout development by continuous feedback from knowledgeable peers. The focus on intuitiveness and ease of use was validated by testing the applications with non-technical individuals and the execution of a standardized questionnaire, System Usability Scale, in which both applications obtained a score in the range of "Good" to "Excellent".

The developed applications culminated in an effective abstraction of the multiple modules of the SmartOs system, presenting great potentiality in regards to the expedition of the configuration process and monitoring of the multiple modules of the system through the development of simple yet complete interfaces that allow to complete proposed objectives with minimal user input whilst ensuring safe and organic usage of the system.

RESUMO

O caminho para a reabilitação física de cada pessoa pode não ser sempre facilmente extrapolado a partir casos prévios. A reabilitação pode assim necessitar de experimentação, tradicionalmente realizada com dados observacionais, podendo estes ser custosos e menos precisos.

É com isto em mente que o SmartOs, um projeto multi-disciplinar que consiste na criação de um sistema ortótico inteligente vestível, foi criado. O SmartOs é um sistema modular cujo objetivo principal consiste em ajudar pessoas com problemas físicos, mais especificamente ao nível da marcha. Esta ferramenta incorpora um conjunto de sensores e dispositivos ortóticos ativos para fornecer recursos de monitorização e assistência da marcha respetivamente.

De modo a assegurar a usabilidade do sistema, são necessárias ferramentas para controlar e monitorizar o referido sistema, no entanto estas ainda não se encontram desenvolvidas para o SmartOs. O foco desta dissertação é então o desenvolvimento destas ferramentas, as quais foram projetadas, desenvolvidas e validadas nesta dissertação.

A aplicação Android foi desenvolvida para proporcionar o controlo intuitivo de parâmetros do SmartOs, bem como para assegurar a configuração de uma sessão de terapia sem qualquer compromisso funcional para o utilizador e o sistema.

Ainda no contexto desta dissertação, foi escrita uma aplicação desktop em Qt permite a monitorização de múltiplos parâmetros em tempo real ou numa altura posterior, recorrendo ao traçamento de gráficos e outros componentes interativos num dashboard adaptado ao usuário. Também deverá gerir sessões de terapia que serão gravadas localmente sob um formato estandardizado no qual serão empacotados os dados de cada sessão.

A capacidade e correção das aplicações foram validadas durante o desenvolvimento através do feedback continuo de colegas integrados no projeto. O foco na intuição e facilidade de uso das aplicações foi validado através do teste das mesmas com indivíduos não técnicos e da execução de um questionário estandardizado, a Escala de Usabilidade de Sistemas. As aplicações obtiveram um resultado na gama do “Bom” a “Excelente”.

As aplicações desenvolvidas resultaram numa abstração efetiva dos múltiplos módulos do sistema SmartOs, apresentando grande potencialidade em relação à expedição do processo de configuração e monitorização dos múltiplos módulos do sistema através do desenvolvimento de interfaces simples mas completas que permitem a execução dos objetivos propostos através da minimização do input do utilizador, garantindo, ainda assim, a segurança e utilização orgánica do sistema.

CONTENTS

1 INTRODUCTION 1
- 1.1 Motivation and Problem Statement 1
- 1.2 Goals 3
- 1.3 Thesis Outline 4

2 STATE OF THE ART 6
- 2.1 Fitness Application 6
- 2.2 Clinical Application 10
- 2.3 Application in the Academic Community 15
- 2.4 Critical Overview 17

3 SMARTOS 21
- 3.1 Description 21
  - 3.1.1 SmartOs Architecture and Submodule Integration 22
- 3.2 Current Standing and Critical Analysis 24

4 SMARTOS CONTROL INTERFACE 25
- 4.1 Central Controller Integration 25
  - 4.1.1 Design and integration of Bluetooth communication 26
  - 4.1.2 Establishing a Communication Standard 27
- 4.2 Design and Implementation 32
  - 4.2.1 Early considerations and technology choices 32
  - 4.2.2 Application principles 34
  - 4.2.3 Basic Application Structure and Mockups 35
  - 4.2.4 Application Architecture and Navigation 39
- 4.3 Final Application and Critical Analysis 42
  - 4.3.1 Application Validation and Testing 49

5 SMARTOS MONITORING INTERFACE 57
- 5.1 Central Controller Integration 57
  - 5.1.1 Design and Integration of Wifi communication 58
  - 5.1.2 Establishing a Communication Standard and Data Marshaling 59
- 5.2 Design and Implementation 62
  - 5.2.1 Early considerations and technology choices 62
  - 5.2.2 Application principles 63
  - 5.2.3 Basic Application Structure and Mockups 64
  - 5.2.4 Application Architecture and Navigation 66

5.3 Final Applications and Critical Analysis 69
5.3.1 Application Validation and Testing 75
6 CONCLUSIONS 79
6.1 Future Work 81

A S.C.I. AND S.M.I. REAL TIME SESSION 86

LIST OF FIGURES

Figure 1	Sensoria's Fitness Products	7
Figure 2	FEETME Sport Monitoring System [18]	8
Figure 3	FEETME Diagnosis	8
Figure 4	Fitbit Ecosystem	9
Figure 5	F-Scan's FootMat Software [38]	10
Figure 6	XSens MVN Analyse Software [40]	11
Figure 7	Hexoskin professional solutions	11
Figure 8	OpenSignals (r)evolution [13]	12
Figure 9	eHealth Monitoring System	13
Figure 10	Lokomat System	14
Figure 11	eHealth Monitoring System	15
Figure 12	CTG Analyser	16
Figure 13	Wearable Sensors System	17
Figure 14	External Module Communication proposed in [28]	22
Figure 15	Labeled S.C.I command messages	30
Figure 16	Labeled S.C.I report messages	32
Figure 17	Early mobile application concepts seen in [28]	33
Figure 18	Application Activities	36
Figure 19	Applicattion Mockups	37
Figure 20	System Architecture	39
Figure 21	Therapy Process Menus from beginning to end	44
Figure 22	N-ary options and Absolute Value options	45
Figure 23	Walkthrough pattern and guidance tactics	46
Figure 24	Custom Monitoring Layouts	47
Figure 25	Help Button and Spotlight	48
Figure 26	Communication Process	48
Figure 27	S.C.I. interaction Flowchart	50
Figure 28	S.C.I. development feedback loop	51
Figure 29	Grading Scale for the SUS	52
Figure 30	Relabeled components	55
Figure 31	Mandatory text update	55
Figure 32	Labeled S.M.I desktop messages	60
Figure 33	Abstract representation of JSON file content	61

Figure 34 Early desktop application concepts seen in ”A real-time architecture for smart wearable orthoses system” [28] 62
Figure 35 S.M.I. mockups 65
Figure 36 Desktop Application Architecture 67
Figure 37 Database Tables 69
Figure 38 Establishing connection to the C.C.U 70
Figure 39 Patient and Session Management 71
Figure 40 Importing a previously recorded session 72
Figure 41 Real Time Therapy Observation 74
Figure 42 Review Therapy Session 74
Figure 43 S.M.I. interaction Flowchart 76
Figure 44 Interaction between major SmartOs’s components over time 86

LIST OF TABLES

Table 1	Systems Overview	18
Table 2	SUS scores for the S.C.I.	53
Table 3	SUS scores for the S.M.I.	77

ACRONYMS

API Application Programming Interface. 2.2, 2.4, 4.2, 5.3

BC Bluetooth Communication class. 4.1
BiRDLAB Bioinspired/Biomedical Robotic Devices. 1.0
BPM Beats per Minute. 2.3

C.C.U Central Control Unit. 3.1, 3.2, 4.1–4.3, 5.0–5.3, 6.0, 6.1, A.0

ECG Electrocardiogram. 2.3, 2.4

GUI Graphical User Interface. 2.2, 2.3, 4.3

IED Industrial Electronics Department. 1.0
IoT Internet of Things. 6.1

L2CAP Logical link control and adaptation protocol. 4.1
LLOS Low-Level Orthoses System. 3.1, 3.2, 4.1
LTE Long-Term Evolution. 6.1

OS Operating System. 2.4, 5.2

RFCOMM Radio Frequency Communication. 4.1
RPI3 Raspberry Pi 3. 3.1, 3.2, 4.1, 5.1

S.C.I SmartOs Controller Interface. 1.2, 4.0–4.3, 5.0, 5.1, 5.3, 6.0, 6.1, A.0
SDK Software Development Kit. 4.3
SDP Service Discovery Protocol. 4.1
S.M.I SmartOs Monitoring Interface. 1.2, 4.3, 5.0–5.3, 6.0, 6.1, A.0
SUS System Usability Scale. 4.3, 5.3, 6.0

TCP Transmission Control Protocol. 4.1, 5.1

UI User Interface. 1.1, 4.2, 4.3, 5.2, 5.3
UX User Experience. 1.1

WMSS Wearable Multi-Modular Sensory System. 3.1, 3.2, 4.1, 4.2

1

INTRODUCTION

The present dissertation was developed as the final project for attribution of a Master's Degree in Informatics Engineering.

This project focuses on the development of two graphical applications for the control, monitoring and management of the therapy sessions realized by a portable rehabilitation system, the SmartOs. The intent of these applications is both to facilitate the process by which mechanically assisted therapy is realized as well as provide the means by which a trained professional can analyze the development of a treatment process.

The present dissertation is part of a multi-disciplinary project developed in the Bioinspired/Biomedical Robotic Devices (BiRDLAB) of the Industrial Electronics Department (IED) in Universidade do Minho under the guidance and supervision of Professor Cristina Manuela Peixoto dos Santos.

1.1 MOTIVATION AND PROBLEM STATEMENT

Rehabilitating one's physical illness or impairment does not usually have an expeditious or evident solution. The healing process can encompass a multitude of physicians and therapists which must rely on their experience and training, as well as the reports and feedback of their patients to find each person's road to treatment.

As such, there is no all encompassing solution for the gaps in this field, so problems should be tackled individually and methodically. One obstacle in particular lies in the motor rehabilitation field, where **automatized treatments** could provide better results, potentially maximizing both diagnostic and recovery efficiency.

Whilst experience and communication might be **integral components** for optimal recovery [25], the ideal path to treatment might not always be easily extrapolated from the work with previous patients. Furthermore, patient feedback could sometimes be inaccurate, or maybe difficult to explain [14]. This could implicate a situation where physical rehabilitation could demand experimentation, traditionally made through observational data which might be costly and less precise.

The uncertainty that could arise from using observational data could potentially be consolidated through the usage accurate measurements that could, in turn, help guide the clinician towards an optimal treatment plan. With this in mind the SmartOs, a multi-disciplinary project that consists on the creation of a smart wearable orthotic system was started.

The development of advanced solutions for use in the medical domain could be a field with great potential for improvement. As such, technological advancement could provide a way to more effectively recover from sporadic malaises. In this sense SmartOs, being a portable rehabilitation device, **could possibly be contributory** in helping guide both patient and clinicians alike throughout therapy.

In its **current state**, this system only allows for **low level interaction**, likely rendering itself useless for anyone without the know-how and programming expertise to operate it.

It is with this in mind that **we think it is necessary to create intuitive graphical applications** with which to **interact** with the currently existing system, the SmartOs, providing users with straightforward interfacing options. Moreover it is important to create a system which follows a natural design language, as the adoption of a piece of software could be, in many instances, defined by the quality of the user interaction with the system.

It could be argued that an unresponsive system is eventually bound to fail, as poor User Interface (UI) or User Experience (UX) implementation, and consequent lack of understanding of the system's visual language could deter its usage by potential users.

Furthermore, SmartOs integrates an orthotic system and a wearable multimodal sensory system. The modularity of the system allows it to perform a panoply of gait therapies and aid the evaluation of gait performance.

Therapies fulfilled by the SmartOs system **use different modules** that can act in **conjunction or as standalone devices**. As such, there is the need for a graphical interface with which to **activate and configure** the functionalities and modules to be used in a gait evaluation scenario.

Moreover, it is necessary to evaluate the gait patterns of the subjects partaking therapies in **real-time**, thus generating the need for an application with the **ability to monitor data from the currently existing system**.

In its current state, the SmartOs **does not possess interfaces with which to interact** graphically with the system. **Resorting to low-level** interfaces that **can only be used by a technical user** to activate the multiple modules of the system.

Furthermore, monitoring of the system can only be effectuated **after a therapy session** by using MATLAB scripts to parse files with session data saved to the local storage of the system throughout the duration of the therapy.

Ultimately, the main motivation for this Thesis is to allow for an innate usage of a system that could be described as complex by effectively abstracting the end user from all the implementation intricacies that culminated in the development of the multiple sensory and

therapeutic modules that compose it whilst not losing any of the features these might provide.

1.2 GOALS

As introduced in the precedent sections of this document, SmartOs could be described as a tool without an approachable controller.

The system was conceptually designed to communicate with external devices through wireless communication, namely **Wi-Fi** and **Bluetooth** for communication with **desktop** and **mobile** devices, respectively. In this sense, the main goal of this dissertation is the creation of two applications with which to graphically control and monitor the SmartOs system, the SmartOs Controller Interface (S.C.I) and SmartOs Monitoring Interface (S.M.I) respectively, granting usability and intuitiveness to the end-user.

As such, the main objectives to be achieved are:

- Update the SmartOs codebase to enable wireless communication between an external device and the central control unit of the SmartOs, integrating a **module for Bluetooth communication and a module for WiFi communication**.
- Establishment of **standardized messages** for interaction with mobile and desktop applications, the S.C.I and S.M.I respectively.
- Design and establishment of requirements and principles for the S.C.I and S.M.I.
- Development of an Android application that will allow the user to control the adjustable parameters of the SmartOs system intuitively. I.e, providing a **means of communication between the mobile device and the powered orthotics and sensors** in order to establish treatment options. Additionally, define **sensory information to be collected** in a manner that is user-friendly by employing a graphical approach with a natural design language.
- Creation of a desktop application that will allow the **monitoring of multiple parameters in real time**, through the usage of plots and other graphical components in a dashboard design tailored to the user. As well as the establishment of a standardized format with which to **marshal session data** and consequently record a therapy to a local file, and development of interfacing components with which to **review them at a further date**, i.e. not in real time.
- Validation of the time-effectiveness, robustness, and portability of the applications by recording and testing their usage with the actual SmartOs modules using different devices (different smartphones, PCs) and through feedback of technical users.

Furthermore, validation of the intuitiveness and usability of the applications by involving technical users.

1.3 THESIS OUTLINE

The current dissertation is composed by 6 chapters. This section aims to give a brief description of each chapter and its content, with the exception of the present chapter. This document is therefore structured as follows:

- **Chapter 2** is focused on the investigation of similar work to that of the one being covered in this dissertation. It is divided into four sections, each covering a different field of application of technologically assisted physical recovery or tracking devices, culminating in a comparison and critical analyses of the multiple solutions presented.

- **Chapter 3** is dedicated to the introduction of the SmartOs system, comprised of two subsections, the first of which intents to introduce the existing system and its current functionality and tasks.

  The second subsection delves into the architecture of the aforementioned system, namely the process by which subsystems can be added to the SmartOs through integration with the central control unit of the system.

- **Chapter 4** introduces the first of two applications to be developed in this dissertation, more specifically the mobile application, the SmartOs Control Interface. It is comprised by 3 sections.

  The initial section details the efforts realized to prepare the currently existing system to accommodate a new graphical application for **commandeering** the entire system.

  The second section demonstrates the process of creating an **architecture and solid foundation** for the development of the aforementioned control tool ensuring that guarantees basic guidelines of usage and security.

  Followed by a last section in which the finalized version of the application and analyzed in regards to the **goals of the thesis** and set guidelines and consequent **validation** of the application with technical and non-technical users.

- **Chapter 5** introduces the development of the desktop application.

  The first section now demonstrates the preparations needed for the integration of the now monitoring application to the existing system. Again a **structure and architecture** is proposed for the establishment of said application along with some additions to its basic guidelines such as **minimality and automation**.

The last section analyzes the behavior of the application and fulfillment of proposed tasks and consequent **validation** of the application with technical and non-technical users.

- **Chapter 6** concludes the dissertation and examines the process and resulting products of the work developed throughout this thesis, whilst setting some ideas for work in the future.

2

STATE OF THE ART

The growth in monitoring and health technology has culminated in the development of multiple new solutions in what can be called the eHealth field, i.e., *the cost-effective and secure use of information and communications technologies in support of health and health-related fields* [30]. However, the eHealth field is **broad, fomenting a variety of use-cases**. The following sections aim to provide some categorization and overview of services and research papers deemed to be of consequence to the scope of this dissertation.

The categorization that follows will be divided into **three fields** of application. The first section will cover the application of eHealth solutions within the **fitness field** and their importance in the current scenario. A second section will examine the application of eHealth services in a **clinical context**. Followed by a third section in which solutions employed in the **academic community** will be examined.

Moreover, whilst the present dissertation aims to provide tooling for meeting the needs of SmartOs, a device that focuses on motor rehabilitation, the basic concepts for the tools to be implemented in the system are universal. As such the state of the art will focus on solutions for **monitoring** and **control** of sensory systems within the health environment in general. Motor rehabilitation/monitoring devices will still be presented and prioritized.

The reviewed solutions will also be compared, culminating in a critical analysis of the current state of the art.

2.1 FITNESS APPLICATION

Exploring this industry attains importance in this context as the solutions proposed in this domain target, usually, the average smartphone owner, as such, the products presented in this section should be more likely to take into account the importance of creating interfaces that abstract the standard, non-technical user from the product elaborateness.

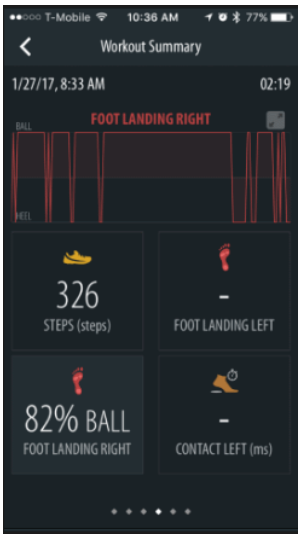
The complexity of these systems should be lower than the typical medical device, as the ultimate goal is to provide a **cost-effective yet rigorous way to track physical fitness**, being limited, usually, in its ability to aid physical rehabilitation.

Nonetheless, in spite of having fitness as a main goal some of these companies occasionally use their technological expertise developing parallel projects to venture into the realm of physical therapy and rehabilitation.

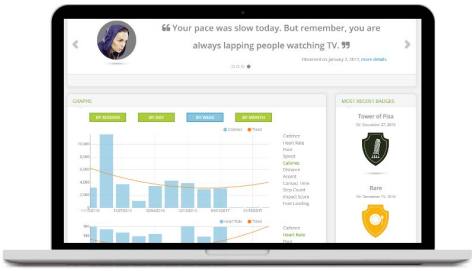
One of such solutions comes by the way of **Sensoria Fitness** [33] whose fitness devices include **heart rate** monitoring systems as well as **pressure sensitive sensitive sensors**(textile sensors [34]), which come in the form of socks, as shown in Figure 1a.

These allow both for the analysis and recording of running/walking cadence and evaluation of effort. This hardware is backed up by its software counterpart which comes, in this case, in the form of two iOs applications, as seen in Figure 1b, helping the user achieve walking and running goals. Additionally an online dashboard, shown in Figure 1c, complements both mobile applications allowing for a personalized tracking of physical evolution.

(a) Pressure Sensitive Socks [34]



(b) iOS Application [33]



(c) Sensoria's Web Solution [33]

Figure 1.: Sensoria's Fitness Products

Another practical implementation of a fitness solution is proposed by **FEETME SPORT** [18], shown in Figure 2.

This solution is similar to that of Sensoria Fitness, albeit in this specific solution there is no heart monitoring capability. An **insole** is used in lieu of the pressure sensitive socks, ultimately allowing for the **analysis and tracking of the user's movement patterns** and **posture** through **desktop and mobile apps** that intend to aid the optimization of one's gait efficiency.

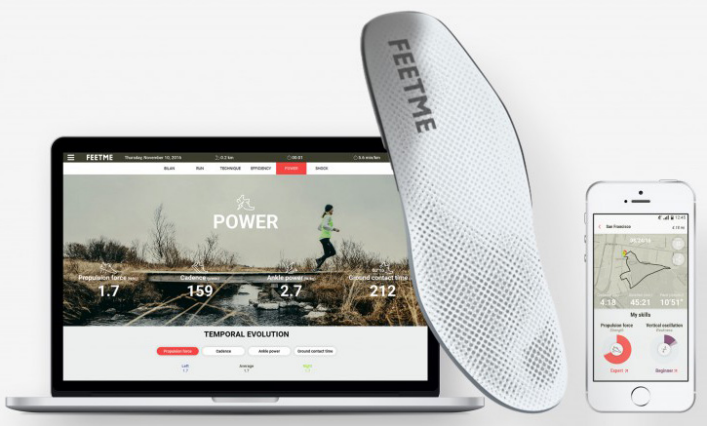
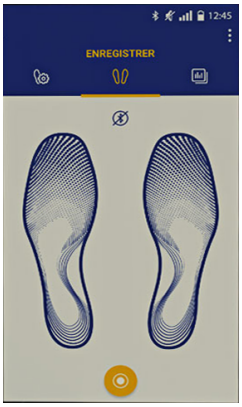


Figure 2.: FEETME Sport Monitoring System [18]

Information is provided regarding both enterprises' ability to operate within the rehabilitation domain, but ultimately only **FEETME** presents a fully developed product specialized in the aforementioned field, the **FEETME Diagnosis**, shown in Figure 3. This solution is comprised of a mobile application, the **FEETME Holter** [17] for interfacing with the insoles with the **sole purpose of data acquisition**, as shown in Figure 3a.

Acquired data would in turn be represented and analysed in the desktop counterpart, the **FEETME Podiatrist** [19], allowing for the study of **gait biomechanics and posture through the graphical representation of pressure** parameters, as seen in Figure 3b.



(a) FEETME Holter [17]

(b) FEETME Podiatrist [19]

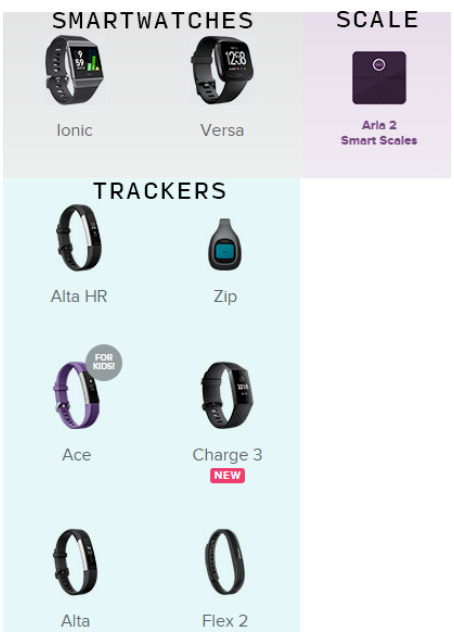
Figure 3.: FEETME Diagnosis

Also on the field of fitness tracking and monitoring **Fitbit** [3] proposes a solution unrelated to gait analysis. Fitbit provides an array of products for **tracking of physical health**, these can range from different smart-watches and trackers to a weighing smart scale, as seen in Figure 4a.

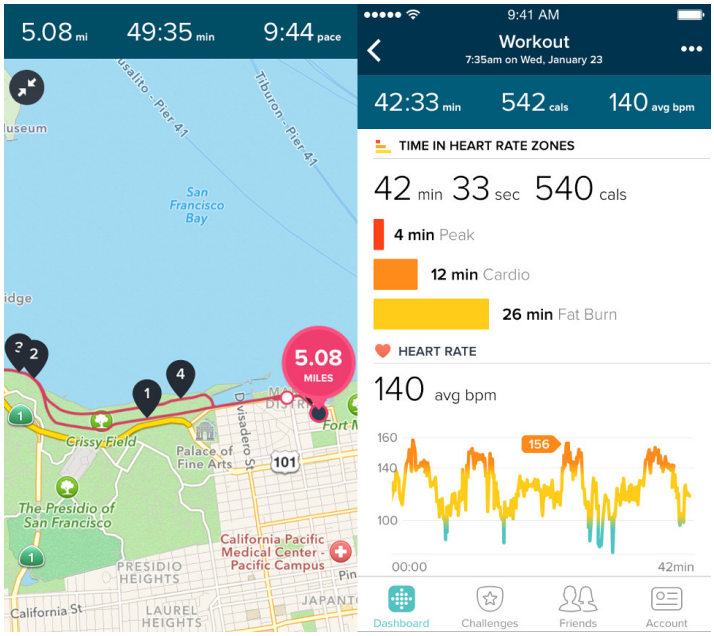
Figure 4b shows a the mobile application used to connect to the multiple smartwatches and trackers over Bluetooth, the Fitbit App [2]. These sync real time data to the application such as **distance ran and heart rate over time**.

Workout sessions can be synced in **real time or alternatively synced after each workout** with a mobile device. Once connected to the internet the Fitbit App is able to send data to the user's account, also providing a cloud based solution.

Fitbit dashboard [2], an application for **dashboard monitoring of recorded workout sessions and evolution** is also available for desktop, as seen in Figure4c.



(a) Fitbit products[3]



(b) Fitbit App [2]

(c) Fitbit Dashboard [2]

Figure 4.: Fitbit Ecosystem

A further analysis of the technical characteristics of the previously introduced solutions is covered in section 2.4. These characteristics are presented in table 1.

2.2 CLINICAL APPLICATION

The goal of this section is to present solutions integrated in a clinical environment, either in the realm of physical rehabilitation by applying active physiotherapy or solutions that can facilitate diagnosis of different pathologies through the visualization of sensory data.

Tekscan [35] offers an array of solutions for measuring **force and pressure**, the most relevant to the current context being **F-Scan** [36]. This foot pressure measurement system can also be bundled with Tekscan's software for clinicians, the **FootMat** [38], illustrated in Figure 5, allowing for the evaluation of foot function and the establishment of real-time metrics claiming its effectiveness through reports of certified podiatrists [37].

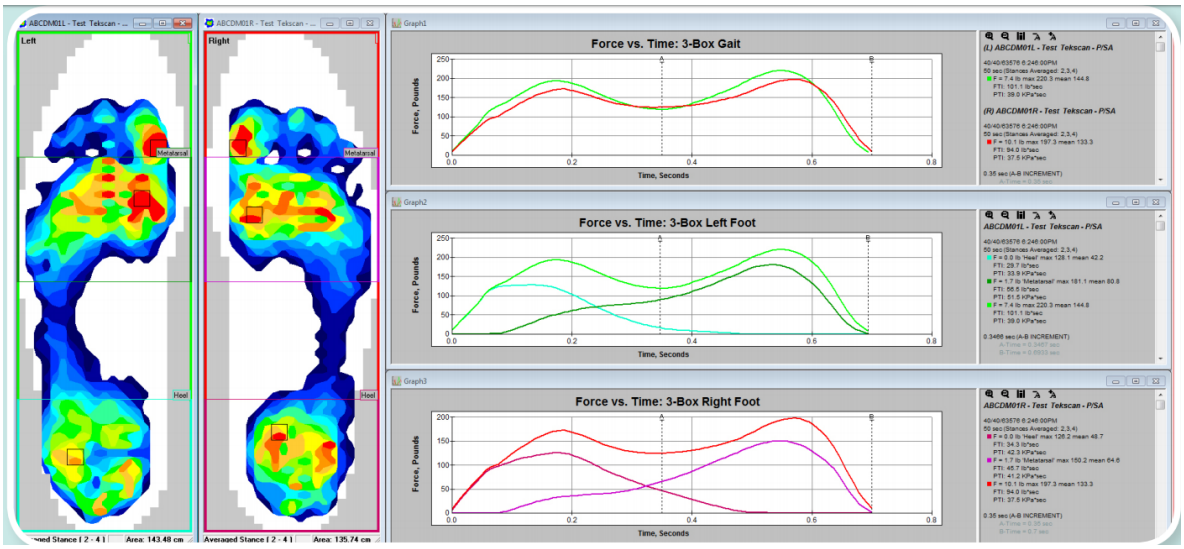


Figure 5.: F-Scan's FootMat Software [38]

Treatments can range from the usage of the collected metrics to guide the design of customized orthotics to the education of patients' on their pathologies and potential treatment possibilities.

Another similar approach to the establishment of a metric driven system for physical recovery is presented by **XSens** [39], more specifically **XSens MVN Analyse** [40], shown in Figure 6.

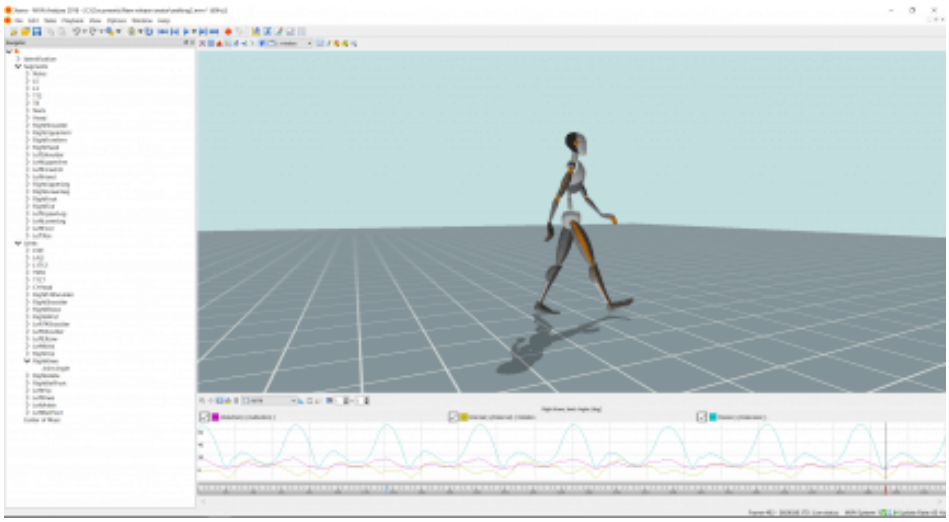
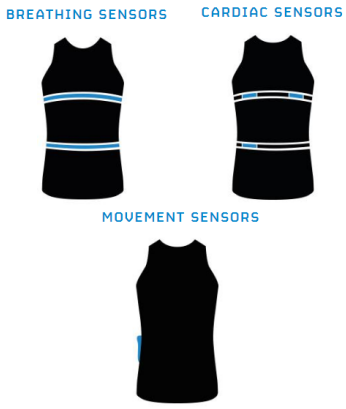


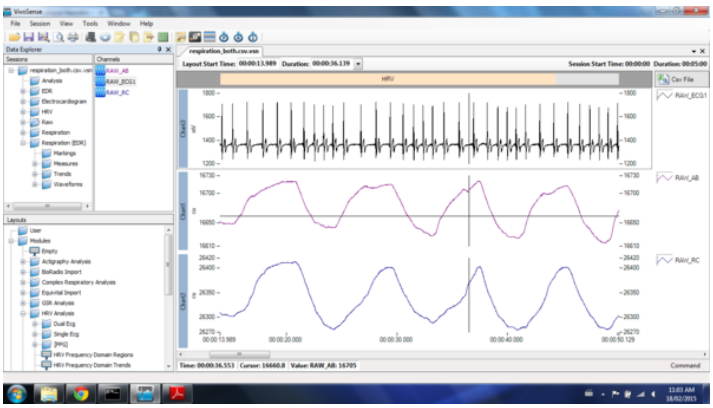
Figure 6.: XSens MVN Analyse Software [40]

XSens's solution integrates **17 inertial motion trackers** allowing for a complete approach to movement tracking. Bundled with its proprietary software it offers multiple data presentation options. These options range from the **graphical representation of movement velocity and force**, to the elaboration of motion through an animated **3d representation of the patient's movement patterns**, which could improve the detection of gait or even movement pathologies.

Some solutions are equipped with multiple systems for monitoring different sensory data. HexoSkin [4] provides a wearable solution for **analysis and monitoring of body metrics**, shown in Figure 7. Whilst also offering mobile and desktop applications for fitness tracking there is extensive information regarding the usage of the system within the clinical field along with **multiple scientific publications**[5] hence its inclusion in the current section.



(a) Hexoskin Sensors [4]



(b) VivoSense Analysis Software for Researchers [5]

Figure 7.: Hexoskin professional solutions

The present system makes use of a **smart shirt** incorporating **multiple sensory devices** shown in Figure 7a, permitting the capture of ECG and heart rate data, along with breathing and cadence data. It is therefore adapted to different fields of medical practice claiming to be integrated in multiple therapeutic areas[5].

As previously introduced proprietary software can be used to establish real time monitoring of the system in a fitness context. However, for clinical and research purposes, **integration with VivoSense** [8] [9] is more adequate for processing and visualizing collected data from the Hexoskin sensory devices.Figure 7b illustrates the **module for ECG and respiratory data developed by Hexoskin for the VivoSense**.

Still on the subject of monitoring technology, **BITalino**[11] offers one of the widest ranges of options. Developing both their own hardware and software suite, the **OpenSignals (r)evolution**[13], illustrated in Figure 8. Providing products with intent of helping researchers and programmers develop projects focused on sensory data acquisition and representation. Multi-platform software solutions are available to allow for a highly **customizable data plotting environment as well as automatized creation of data report sheets**.

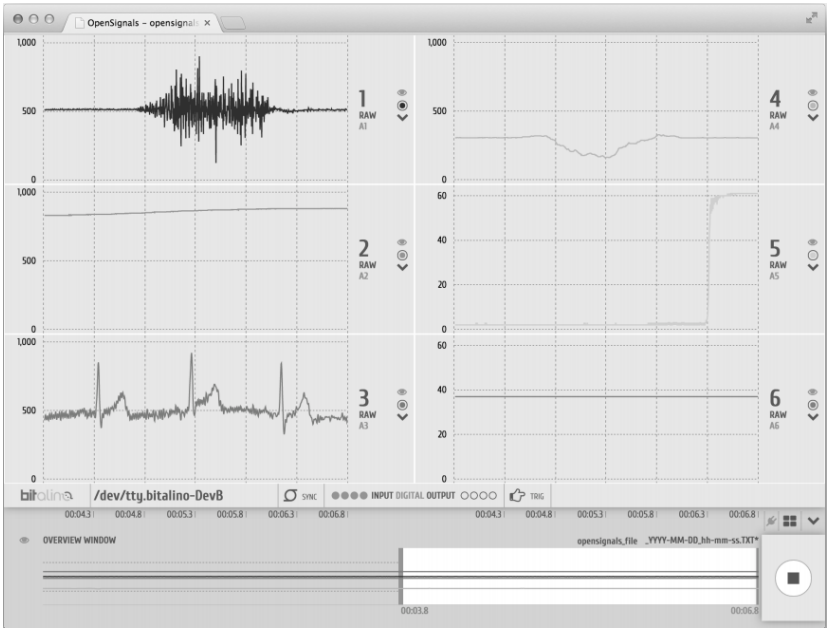


Figure 8.: OpenSignals (r)evolution [13]

In addition to all these features their hardware can be integrated in any third party application as an Application Programming Interface (API) is provided for most modern languages albeit with different compatibility factors[12].

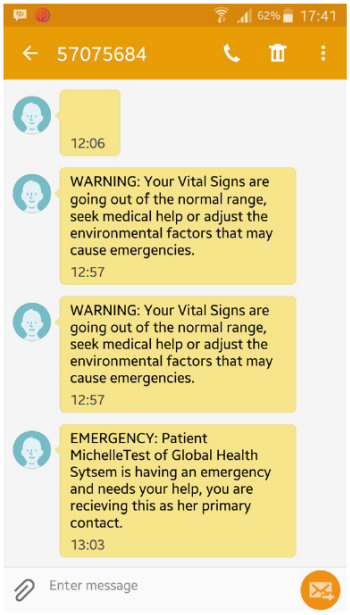
Another study to take into consideration when analyzing the current state of eHealth technology is ***Critical Patient eHealth Monitoring System using Wearable Sensors*** [29]

which presents the development process of a vital signs monitoring system and its comparison to other systems in the market.

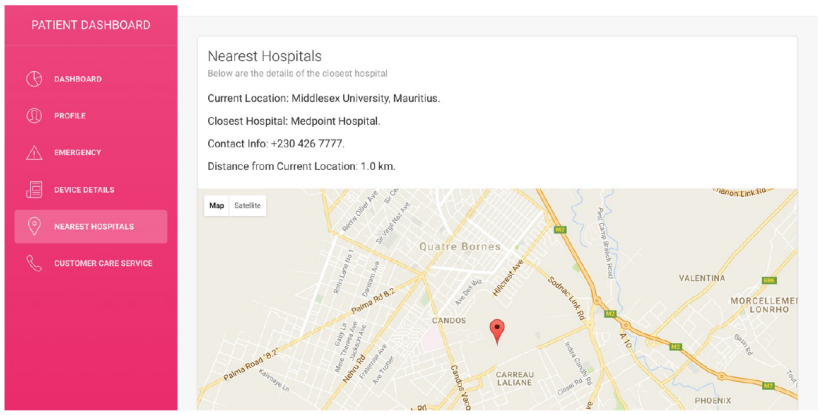
Although this product has no focus on physical rehabilitation or movement analysis the solution presented in this article proposes a modern approach to the usage of smart wearable devices with an effective system architecture. It focuses on the monitoring of risk patients, any cardiac irregularity prompts the notification of the responsible entities and the patient over the short message service as seen in Figure 9a .

Another particularity of this system is the usage of its cellular module to log patient information in a remote database. This allows to **track the user's vital signs throughout the usage of the system by using a web application**, shown in Figure 9b. Which can be especially useful as a diagnosing tool as a link to the patient's vitals from the last hour before an emergency is **automatically sent to the health-care provider**.

Analogously, the remote logging of information could be of particular interest in a rehabilitation context as a therapist would be able to record sessions and analyze progress throughout the entire treatment optimizing the process of recovery.



(a) SMS warnings regarding the vitals of the patient [29]



(b) Web Graphical User Interface (GUI) [29]

Figure 9.: eHealth Monitoring System

So far the presented solutions have only been capable of aiding clinicians establish a diagnosis. Following diagnosis it is necessary to proceed with the treatment of the diagnosed disorder. To that end, the merger of monitoring systems with physical feedback devices/systems can be instrumental in achieving optimal recovery results.

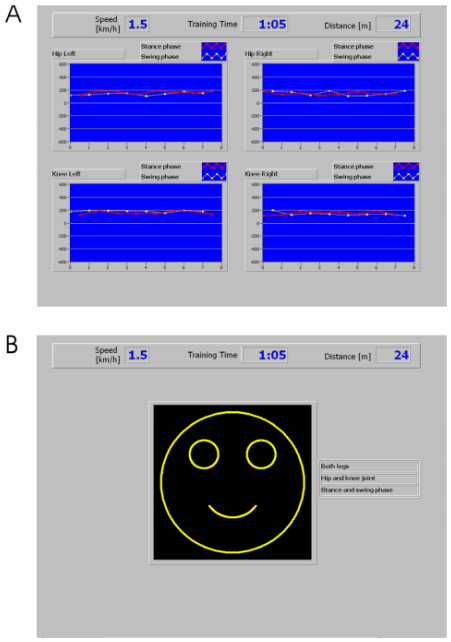
An example of robotic assisted therapy is **Lokomat** [25] , shown in Figure 10a. This solution integrates **force monitoring sensors to evaluate patient's gait patterns** which would in turn be analyzed by the operator and consequentially used to chose a treatment. The system can detect deficiencies in posture and stride, by comparing sensor data and treatment gait parameters, and ultimately assist the patient to walk within previously mentioned gait parameters.

The benefit of such a solution is the diminished variance of quality that inherently comes from multiple factors when being treated by a human professional such as fatigue and the limitations that incur from real time visual assessment of gait deficiencies.

This product **could also maximize the patient's recovery performance through proportional maximization of effort**. It is able to exert precisely the right amount force with which to help the treatment subjects on any given day, disassociating itself from emotional constraints.

Another important factor of this implementation is that the quality of the training is presented to the user, having been developed a software which provides **real time visual feedback on the stride**. A smile is drawn on screen for optimal behavior, or a frown whenever effort is evaluated to being less than maximum, this behavior is illustrated in Figure 10b.

(a) Lokomat Rehabilitation System [25]



(b) Lokomat's Visual Feedback [25]

Figure 10.: Lokomat System

Technical details of the introduced clinical solutions are also covered further in section 2.4 and presented in aforementioned table 1.

2.3 Application in the Academic Community

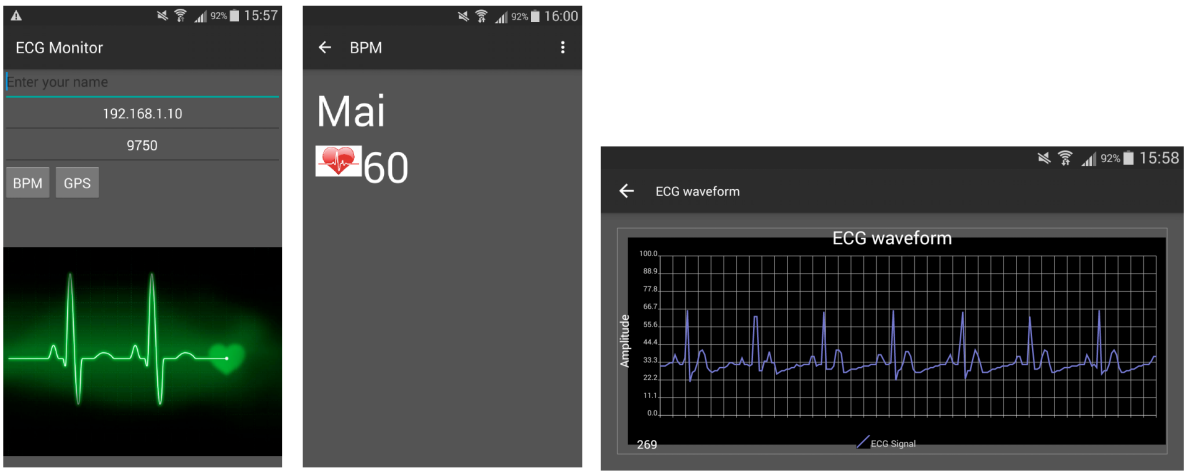
This section will cover projects that pertain to a similar objective to the present dissertation amidst the academic community, i.e. academic papers or articles regarding the creation of monitoring or control systems for interaction with sensory devices as well as the development or lack thereof of software to allow for the usage of referred tools by non-technical users, be these clinicians or patients.

An **architecture** for a potential eHealth system is contemplated in ***Insole optical fiber sensor architecture for remote gait analysis - an eHealth Solution***[16] where it is **proposed** the acquisition of data from a pressure sensitive insole, which would in turn be sent to a data processing and local storage unit such as a smartphone and consequently backed up over wireless communication to the cloud for further analysis.

Also on the realm of gait analysis, one other study regarding the ***Design and Evaluation of a Low-cost Mechatronic System to Study Upper and Lower Limbs Biomechanics***[24] shows a similar take on the design of a monitoring system using a arduino as a bridge between the sensors and the data acquisition device, in this particular paper there is no mention of remote storage of data and the arduino data is displayed using **MyOpenLab**[26] platform, a tool for creating custom GUI through drag and drop widgets, focused on the establishment of graphical widgets for academic or research purposes being particularly useful for the establishment of a generic prototype of a potential GUI.

Some academical articles also focus on the creation of practical solutions for the monitoring of health systems.

For instance, ***Wireless Body Sensor Network and ECG Android Application for eHealth*** [23] presented in Figures 11a and 11b.



(a) Configuration Screen and Current Beats per Minute (BPM) [23]

(b) Electrocardiogram (ECG) Waveform plotted [23]

Figure 11.: eHealth Monitoring System

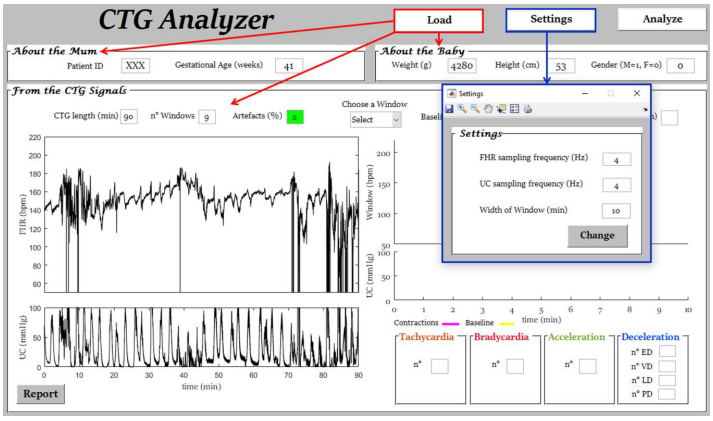
Figures 11a and 11b illustrate the implementation of a system in which an arduino is used to acquire data from a sensor which is in turn transfered to a smartphone where real-time monitoring of the user's heart rate is achieved through the creation of an ECG plot, this application is

Also within the academical field **CTG Analyzer: a Graphical User Interface for Cardiotocography** [32] covers the development of a graphical interface for the improval of quality in cardiotocography, i.e. the recording of fetal heartbeat and uterine contractions during pregnancy.

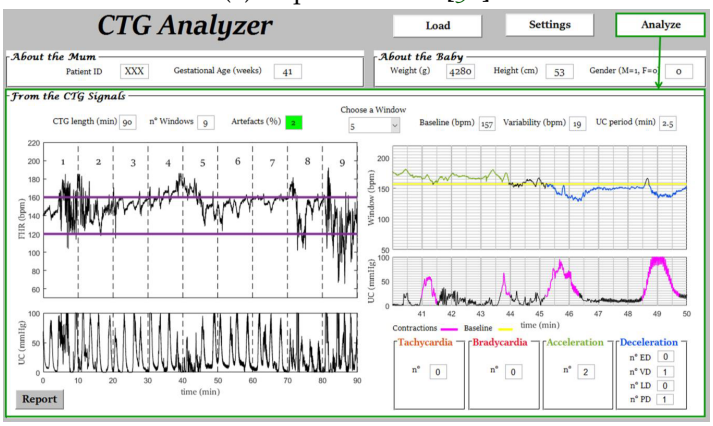
The current solution uses the capabilities of **MATLAB GUI** [6] to create a self-contained program illustrated in Figure 12.

Contrary to previous solutions, CTG Analyzer **imports previously recorded data inputs**, and allows the specification of various settings that pertain to the sampling of the imported signal, seen in Figure 12a.

The program automatizes the process of **data analysis and provides information that might be of use for the clinician**, this process is illustrated in Figure 12b.



(a) Import Session [32]



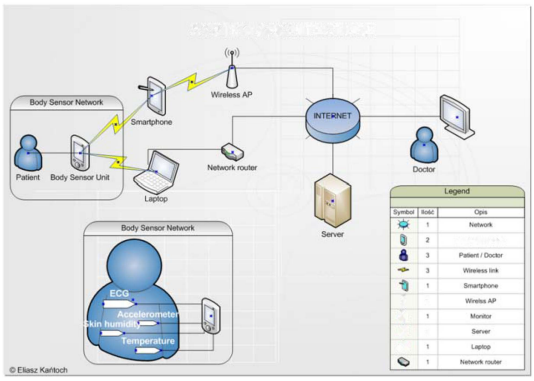
(b) Analysis Execution [32]

Figure 12.: CTG Analyser

Another study on the possibility of remote monitoring of physiological data, **Technical Verification of applying Wearable Physiological Sensors Ubiquitous Health Monitoring** [22].

The study consists on the testing of a potential **architecture for the transmission of physiological data from sensors to a central server** which should endow authorized users with the ability of remotely accessing monitoring data, this architecture is shown in Figure 13a.

Applications for monitoring of data were developed, an **interface of desktop utilization and visualization of recorded data**, as well as a smartphone application for the transmission of data to a server and a web application for the remote features of the architecture, establishing a means for interaction with a non technical user. These applications can be seen in Figure 13b



(a) Architecture [22]

(b) Graphical Interface for PC, Smartphone Application and Web Application [22]

Figure 13.: Wearable Sensors System

Section 2.4 details further the technical aspects of solutions covered in the current section. These solutions were also added to table 1 to facilitate the differentiation of different technical aspects.

2.4 Critical Overview

The intent of this section is to provide some critical analysis regarding the features of the existing products from all fields of application introduced in previous sections of this chapter.

Table 1 establishes a term of comparison of **features present in the previously described solutions** deemed important when evaluating the implementation of eHealth products **in the scope of this dissertation**, the main objective of this comparison is to distinguish main

takeaways when developing a related solution but also determine possible avenues for improvement from deficiencies these solutions might have overlooked.

Table 1.: Systems Overview

Solution \ Features	Platform(s)	Web Platform / Remote Data Logging	Focus	Purpose	Portability
Sensoria Fitness [33]	Smartphone(iOs)	✓	Fitness	Monitoring (HeartRate, Feet Pressure)	Seamless Usage (insole)
FeetMe Diagnosis [18]	Smartphone(Undisclosed Operating System (OS))/ Computer Application (Undisclosed Platform)	✓	Fitness/ Clinical	Monitoring (Foot Pressure)	Seamless Usage (insole)
Fitbit [3]	Smartphone(iOs and Android OS)/ Computer Application (Windows and Mac OS)	✓	Fitness/ Clinical	Monitoring (HeartRate and Distance)	Seamless Usage (Bracelet)
Tekscan [35]	Computer Application (Windows)	✗	Clinical	Monitoring (Foot Pressure)	Highly Portable (Insole Variation + Pressure Sensitive Mat)
XSens MVN Analyse [39]	Computer Application (Windows)	✗	Clinical	Monitoring (Movement Analysis)	Portable (Uses 17 Sensors)
HexoSkin [4]	Smartphone(iOs and Android OS)/ Computer Application (MacOs and Windows + VivoSense [9] for research)	✓	Clinical/ Fitness	Monitoring (Cardic, Breathing and Movement Sensors)	Seamless Usage (Under-garments)
BITalino [11]	Computer Application (Windows + Mac OS+ Linux)/ Smartphone (Android)	✗/✓ (The software is Web-Based/ API allows for the creation of such application)	Clinical	Monitoring (Imense variety of Sensors)	Portable (Portability depends on Sensor)
Critical Patient Monitoring System [29]	None (It has a messaging system)	✓ (Web Application for interaction with personel + Logging of ECG)	Clinical	Monitoring (ECG Sensoring)	Portable (The prototype appears to be cumbersome)
Lokomat [25]	Custom System	✗	Clinical	Monitoring + Active Rehabilitation	Not Portable
Insole Architecture for Gait Analysis [16]	None	None	Academic	Monitoring	Conceptually Highly Portable
Mechatronic System for Biomechanics Study [24]	None	None	Academic	Monitoring	Conceptually Portable (It predicts the usage of 6 sensors)
ECG Android Application [23]	Smartphone (Android)	✗	Academic	Monitoring	Non existent system (Used ECG Simulator)
CTG Analyzer: a Graphical User Interface for Cardiotocography [32]	Computer Application (Windows)	✗	Academic	Monitoring (Software solution for imported data)	Variable (Depends on imported Cardiotocography data)
Technical Verification of applying Wearable Physiological Sensors [22]	Smartphone (Android) Computer Application (Windows)	✓	Academic	Monitoring (ECG,Motion and Humidity)	Highly Portable (Small Block Under the shirt)

Intuitiveness and simplicity appear to be at the base Applications/Products launched for the general public, especially the fitness focused solutions. There are numerous fitness solutions in the market, section 2.1 includes relevant products which focus on gait analysis, the Sensoria and Fitness products, and one of the most popular solutions in regards to eHealth monitoring, the Fitbit.

The academic field encompasses numerous projects with various degrees of development, some of the found related articles procure to establish an architecture for an eHealth solution. These systems propose a similar design to the ones found in Fitness applications with the establishment of a mobile based connection for data acquisition and consequent backup to the cloud.

Solutions within the clinical environment are **less focused on the establishment of portability** and present greater focus on the aid of the diagnosis and rehabilitation.

Lokomatis the sole inclusion of an active rehabilitation system to the state of the art representing the **only found study for powered orthotics and assisted gait therapy that featured a software** counterpart that could **bridge the technical gap** in regards to usability. This system uses a large treadmill and patient support system for gait training, it is also not platform independent being that all of its components part of a **monolithic, non-modular, package**.

Much like the **lokomat** the SmartOs system also provides control over monitoring information and is equipped with an orthotic system wrapped into small, portable solution.

Another seemingly significant element of any popular system seems to be its portability [7], particularly when motor health is decreased, as the ability to setup the rehabilitation system independently of location makes for a more pragmatic and practical solution.

With the exception of lokomat all presented products are portable, this could imply that the lack of this feature would be a great factor in terms of adherence.

In regards to interfacing, **the found graphical applications do not usually posses control over settings of the sensory devices**. This would be a main objective of the present dissertation for the SmartOs system.

Within the Clinical field, BITalino, XSense, F-Scan and HexaSkin provide standalone applications to interact with their hardware, shown in Figures 5, 6,8,7b respectively. F-Scan, Bitalino and HexaSkin both provide native applications with **real-time motion analysis with varying polling rates dependent on the sensors**.

XSens provides a modern solution for monitoring sensory input that is complex in configuration, it offers a great amount of options within multiple nested menus, being likely pursued by a steep learning curve. The minimization of this effect would be important in the scope of this dissertation, when introducing the SmartOs to non-technical users.

In this sense, F-Scan Solution is much more usable to a non-technical user as it displays **automatically all information regarding the feet pressure in an organized manner**. This solution does not, however, encompass multi-modular system, also being limited in the ability to manage and review therapy sessions.

Paid addons add to the features of the BITalino software allowing to preform analysis on different types of sensory inputs. In spite of its modularity it uses a web-based application to display sensory input to visualize and record data, being potentially consuming for lower end systems.

Within the Fitness Industry, most products follow similar design patterns, making use of a mobile application **to connect to and receive data from the sensory data emitters**. This information would then be sent to an online platform in which data could be analyzed posterior to the exercise session shown in Figures 1c and 2.

The rehabilitative solution developed by FEETME also uses a bluetooth connection to record data from the insoles with the application shown in Figure 3a which would then be connected to a desktop application, seen in Figure 3b.

Also important to take into account would be the recurrence of multiple **design patterns in regards to the visualization of sensory data**, both in real-time or at a further review.

Most solutions use a **dashboard design** to present the multiple fields to the user, this allows to organize received data in a stacked and organized manner.

Dashboards are organized in multiple tiles, containing a graphical representation to a specific input or set of inputs. **Charts are also widely used** in most, if not all, monitoring applications to provide an easy to read interpretation of received data over time.

Integration with the cloud and web solutions is also a focus of many of the systems, this is found on many of the portable and commercial solutions as it offers the potential of accessing recorded data from any place with internet connection by an increasing variety of devices.

This feature would help a system such as the SmartOs achieve a level of differentiation allowing the potential clinician to track all therapy sessions and possibly exploring new solutions based on the cross-referencing of different treatment histories.

3

SMARTOS

SmartOs is a modular system whose main goal is to help people with physical, or more specifically, gait impairments. Incorporating an array of sensors as well as powered orthotic devices for both for the knee and ankle joints the SmartOs's objective is not only to provide the means for real-time analysis and gathering of gait patterns and walking data with intent of promoting optimal rehabilitation treatment. Additionally, the usage of the aforementioned data gathering capabilities to provide malleable gait therapy techniques granting active assistance adequate to patients' needs.

All these systems integrate fully with a single point of control, the central control unit of the SmartOs. This module communicates with the multiple appendages/sub-modules of the system being therefore the point of communication for any external applications.

3.1 DESCRIPTION

The previously introduced SmartOs is a portable rehabilitation device, as such, the development of this system has been designed from its conceptual stages to be as **intuitive and modular as possible**. These are some of the pillars of the current project, and **important precursors** to other characteristics of the systems such as the ability to cooperate seamlessly with the human being. The system is easily portable and movable which should in turn facilitate the treatment process as the main processing ability is not anchored to a specific location or environment.

Having portability and expandability as main principles, SmartOs is partitioned into a **plethora of submodules** that were also designed to be able to operate as **standalone** devices whilst retaining the ability to be integrated within the overall system, or more specifically assimilated by the central controller of the system.

In spite of having different specifications and functionality all modules integrated within the SmartOs system should have a common set of rules and global direction, which would be the objective of rehabilitating human function and biomechanics.

In order to achieve this goal, SmartOs was planned to be composed of the **conjoint works** developed by an aggregate of Master's students from **different academical backgrounds**.

The way by which multiple modules or appendages can be integrated in the system will be detailed further in the next subsection, as well as some system design decisions that pertain or affect the work being developed in the current dissertation.

3.1.1 *SmartOs Architecture and Submodule Integration*

As described earlier, the SmartOs system was designed from its conceptualization to be an ever evolving piece of technology which follows a **non-centralized foundation**, being that **not** all data is **acquired and processed** by a **single component**.

Instead each addition to the system should be self-sufficient in terms of data retrieval and consequent treatment.**Processed data** should in turn be passed through the Central Control Unit (C.C.U) of the SmartOs, this particular module is composed, or rather ran, by a **Raspberry Pi 3 (RPI3)** and is therefore responsible for running the system's main loop and consequently the **activation and coordination of the multiple appendages** that are added to the system.

Additionally the Control Unit is the **main hub of communication** between the different modules, designated **External Devices**. The system was developed in such a way that the addition of a new module is maximally, even if not completely, abstracted.

This abstraction requires the creation of threaded communication queues guaranteeing that synchrony is achieved through the usage of locking mechanisms. Ultimately allowing information to flow easily between the central controller and the multiple modules. Understanding the basic concepts of the communication process and architecture is important to the creation of the applications that will be developed throughout this document. The need to receive and send information to the central controller implies that these should be developed as a **couple of distinguished external devices** thus being granted their dedicated communication channels.

An overview of this system is presented in Figure 14.

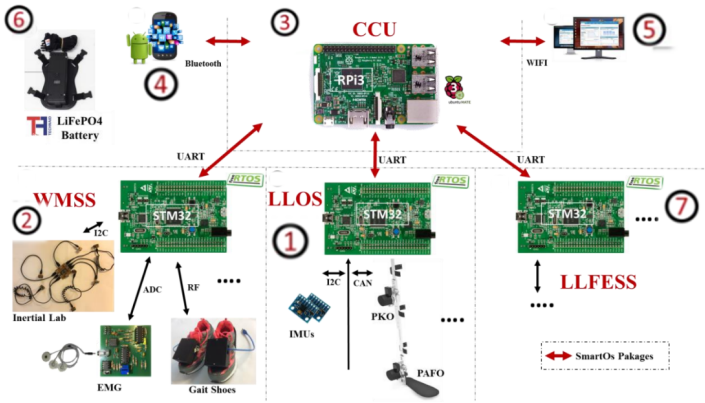


Figure 14.: External Module Communication proposed in [28]

Having covered the basis of the system operation it could be important, in order to fully grasp the integration of modules in the system, to divide it into major subsystems. Therefore specifying some of the abstractions that were previously made when explaining the communication system used in the SmartOs.

The SmartOs could be divided into four major subsystems which ultimately work as one.

Firstly the **Low-Level Orthoses System (LLOS)**,identified by 1 in Figure 14, is comprised by two wearable orthotic devices that allow for physical feedback and **actuation**. Consequentially this feedback requires some sensory input, and since any system introduced to the SmartOs environment should be self-sustaining the LLOS is equipped with inertial sensors, allowing itself to run **gait rehabilitation assistance** algorithms.

On another hand, one other major subsystem present on the SmartOs would be the **Wearable Multi-Modular Sensory System (WMSS)**, shown in 2, which is, as indicated by its name, comprised of multiple **sensory systems** with the exclusive purpose of data acquisition. One other major subsystem would be the **C.C.U**, seen in 3, which was detailed in the beginning of this subsection being a crucial element for data communication, **data management across the different subsystems as well as local data logging**.

The last major component of the system would be the **visual interfacing applications** whose development will be the main focus of this document, the role of these should be the **configuration of any alterable settings present on the first two subsystems** (WMSS, LLOS). As well as **visual feedback** regarding the signals outputted from the various sensory devices, these are identified by the numbers 4 and 5 in Figure 14.

This set of functionalities will not, however, be provided by an all encompassing application, having been defined by **the consortium of students responsible from the conceptual definition of the SmartOs system** that a mobile application would be responsible for the both the **configuration and start-up** of system's operations and monitoring, the system should then be able to operate and record to internal storage all data received from the treatment.

However, it could be important to **inspect all the generated data in real-time** as well as import previously recorded sessions, to this intent it was defined that another application should be developed in a desktop environment and would be optional in regards to the normal functioning of the overall system.

LLFES, identified in 7, is not yet created for the SmartOs. Red arrows indicate the method of communication of the multiple modules with the C.C.U of the SmartOs. The modules to be introduced throughout this dissertation, identified by 4 and 5, will communicate via Bluetooth and WiFi respectively.

3.2 CURRENT STANDING AND CRITICAL ANALYSIS

As previously explained, this dissertation is part of a multi-disciplinary project, as such, some of the previously mentioned features of the system represent the contributions of other members of the project. For instance, the subsystems present in the LLOS andWMSS modules.

In spite of the many contributions that have already been made throughout the development of this system it **could not yet be considered ready**, when beginning the current dissertation, for real world usage.

The previous section focused on the overview of major architecture decisions, existing modules and important principles of the project such as the modularity of the system. Whilst making for the best option to allow for integration with multiple devices also implies a higher level of complexity, and there is **not yet a way** in which the average, **non-technical user** would be able to benefit from this system.

System interaction and activation was limited to **the usage of hard-coded snippets in the C.C.U**. Whilst graphical interaction could be a critical aspect to the applicability of such a solution, a set of parameters/commands was not yet defined with which to interact with the system externally. More specifically, there was not yet a way in which the previously mentioned external controllers for system interaction and monitoring could integrate with SmartOs.

Although the system could allow, conceptually, for the integration of new external devices over Bluetooth and WiFi , modules provided by the central controller hardware(RPI3),i.e. the previously existing external devices, used serial ports for communication. As such, no preparation had yet been made to allow for wireless communication, both in terms of system configuration and the creation of communication classes to integrate within the currently existing C++ code-base.

Consequently, there was not yet a way with which to incorporate any graphical applications on the expected platforms and, as such, no way with which an average user would be able to operate the system.

Hereby lies the **necessity of creating intuitive graphical applications and the External Devices with which to integrate them**. This dissertation aims to address this problematic.

4

SMARTOS CONTROL INTERFACE

S.C.I was the selected name for the mobile application that would be responsible for the communication of all settings and control messages to the central controller.

These messages would be sent over a *Bluetooth* communication channel and allow for parameter configuration and session startup.

This application presents the main component of this Master's Thesis, at least in regards to its responsibility, as it should **"restrict"** the end-user from creating **invalid system configurations**.As such, the S.C.I was projected so that the end-user would be guided into creating a safe configuration of the system, hereby guaranteeing that the **physical integrity of both the system and the patient are preserved**.

The current tool also grants **autonomy to the system**, being that it can be operated without the use of a real-time monitoring application, whilst being a dependency and therefore a mandatory module to the function and usage of the system.

Another main goal of the application would be to control the flow of the configuration process, thus generating a straightforward experience, making **simple interaction** available to the user **without being simplistic and potentially excluding any of the accessible functionality** imbued in the multiple submodules that make up the system.

4.1 CENTRAL CONTROLLER INTEGRATION

In order to implement an interface with which to control and interact with the C.C.U it is necessary to add to the existing code-base classes with which to interact with other systems over *Bluetooth*. As explained in chapter 3, the central controller was not prepared to establish communication, in spite of having the hardware capability to communicate over this technology.

As such, the process of adapting the system to **Bluetooth** communication was realized over three main stages. The first of which being the C.C.U configuration. Secondly the creation of classes to be integrated with the previously existing code. Finally, the third stage would ensure the establishment of all the messages that would be sent to the system over the treatment process.

These three stages will be thoroughly explained in the following three subsections.

4.1.1 *Design and integration of Bluetooth communication*

The C.C.U is running on a Linux Based Distribution, more specifically the **Ubuntu Mate OS**.

Additionally a commonly used *Bluetooth* stack in the Linux community, more specifically, multiple open-source projects developed on the RPI3 would be the open-source *BlueZ* stack [20].

Moreover *Bluez* supports core *Bluetooth* layers and protocols, one of which would be the **Radio Frequency Communication (RFCOMM)** protocol, built upon another protocol of the *Bluetooth* stack (Logical link control and adaptation protocol (L2CAP)) the RFCOMM protocol allows the user to send data over a reliable data stream, being very comparable to the Transmission Control Protocol (TCP) protocol for network communication.

In spite of the reduced documentation from the *Bluez* stack development team, extensive documentation and exemplification was provided in **An Introduction to Bluetooth Programming**[21]. With the examples provided in the aforementioned document the process of creating a RFCOMM socket is parallel to that of the creation of a TCP socket.

A socket must be allocated and set to listen for a connection which will ultimately return the PID of the client. From then on the channel of communication is open until it is closed by one of the parts and read and write functions are used to transmit data over the channel.

The last challenge to the establishment of a *Bluetooth* communication channel would be the creation of a discovery service that would allow the server to be discovered by the client, documentation is also provided to this intent and a **service record** that would correspond to our **central control unit** was defined and created and consequently connected to the local Service Discovery Protocol (SDP) server allowing its discovery by any potential client.

Having installed the Bluez libraries the main configuration file used by the stack(located in the **/etc/bluetooth/main.conf** directory) would be altered to eliminate any discovery timeouts and enable *Bluetooth* from the system startup. This step is essential since we want the ability to always have the system be discovered even without the usage of a screen connected to the C.C.U which should be strapped to the individual undergoing the therapy.

Having installed and configured the **Bluez** stack, a generic class for handling *Bluetooth* communication which was developed using the **singleton** design pattern. This allows this class to be invoked and instanced in any other context of the C.C.U application without the need to pass the this class object to the constructor of the multiple classes of the system.

Moreover, the *Bluetooth* communication class (intuitively named Bluetooth Communication class (BC)) does not parse data and generates a generic message with the received data, which can then be parsed in any intended context. The BC is also responsible for establish-

ing and accepting RFCOMM connections using the processes mentioned in the beginning of this subsection and as functions for both input and output of data.

In chapter 3 the overall C.C.U architecture was explained, with this elaboration came about the notion of an external device as the generic unit of integration of an external dependency or appendage, basically acting as a message courier between the multiple submodules of the system.

The External Device is not a mere abstraction of any device to be added to the system but rather a base class that should be expanded by the multiple modules added to the system that will allow two-way communication(input/output) by defining a set of virtual classes that must be implemented.

MobileApp would be the extension, or rather the derived class of **ExternalDevice**. This class would use the generic BC to receive any message sent from the mobile application, which would then be parsed using a specified standard. Given that the message is valid it will be interpreted internally and used to activate lower level protocols developed or in the process of development by other members of the project.

The establishment of a standard would be instrumental in facilitating the parsing of all incoming messages, and is described in the following subsection.

4.1.2 *Establishing a Communication Standard*

The current subsection will address the challenges that followed the creation of a communication channel for the mobile application, namely the coordination of parallel development within a same project.

In spite of the addition of basic communication capabilities to the C.C.U an important challenge to the advancement of the project would still be missing. The definition of messages to be used for reporting warnings and errors and most importantly to allow for the control of the multiple appendages. This would have a major role in allowing the multiple participants of the SmartOs project to operate independently. Separating the development of the multiple modules such as the LLOS and WMSS from the graphical interfacing devices.

Moreover, this process should allow for **easier integration of any new module introduced to the system** as the development of standardized messages for system activation and configuration. As the need for developing hard-coded snippets mentioned in previous chapters as an unwieldy aspect of the SmartOs's current state would be eliminated.

The defined messages can be divided into **two major groups**. Messages to be communicated from the mobile application to the C.C.U could be generally defined as **command messages**, whilst messages reported back to the S.C.I could be defined as **report messages**.

1. **Command Messages** -

As previously introduced control messages are comprised by all the messages that are sent from the S.C.I to the C.C.U. Moreover, the denomination attributed to these messages arrives from the fact that these messages employ control over any configurable aspects of the system, thus representing the configuration of any therapy or monitoring session.

The established set of directives that would enable full interaction with the system could be described as follows:

a) **Global Configuration Message** - This message should precede a treatment initialization, its parameters include all possible configurations of the system.

   This message is comprised by 52 bytes, composed of multiple subfields with a fixed number of bytes for each of the possible configurable options that will be parsed by the C.C.U, the initial byte set to [$0x00$](hexadecimal) as it was decided to have the first byte of each message be correspondent to its ID.

   Furthermore, this message was generated in conjunction with the responsible members for the development of the C.C.U, and its parameters will be the **main driver** for the development of S.C.I application, as such their functionality and consequence to the final graphical design of the application.

b) **Start Message** - This 2 bytes message is responsible for starting the therapy process once the the configuration message has been duly parsed, its **first byte/ID** is set to [$0x02$]. [$0x02$] represents any message that **modifies the therapy state** - i.e., switching the internal status of the therapy from un-started to started and consequently bringing therapy to a halt or eventually, considering the possibility of pausing a therapy in progress it the byte could be valued [$0x02$]. The second byte will be the **subID** of the message identifying the **specific action** to be realized, for instance starting or stopping a therapy. For the start message the second byte's value is [$0x01$] representing the starting of a therapy.

c) **Stop Message** - This message is the counterpart of the Start Message, following a parallel structure, as explained in the previous paragraph the ID is set to [$0x02$] whilst the second byte is set to [$0x00$] as is contrary to that of the previous message, more specifically it is meant to safely stop a therapy.

d) **Emergency Stop Message** - In similarity to the previous Stop Message the objective of this message is to transition the therapy from a started state to a finished state, however it does so in an immediate manner, as such any orthotic devices will not return to a safe resting position, it takes the ID of [$0x02$] and the second byte or subID takes the value [$0x02$].

e) **Speed Configuration and Stiffness Configuration Messages** - In spite of having both speed and stiffness be configured upon receiving the Global Configuration

Message, both these parameters **should be alterable in real-time**, throughout the therapy progress. As such, these messages will allow individual configuration without resending a modified version of the Global Configuration Message. As represented in Figure 15 the Speed Configuration Message is composed of an ID valued [$0x03$] having the consequent 4 bytes be designated to store a float indicating speed in km/h. The Stiffness Configuration Message's ID is valued [$0x04$] followed by 10 bytes that can be divided into 10 individual bytes for each of the possible **configurable stiffnesses** that will be described further in this document.

The visual representation of these messages can be seen in Figure 15.

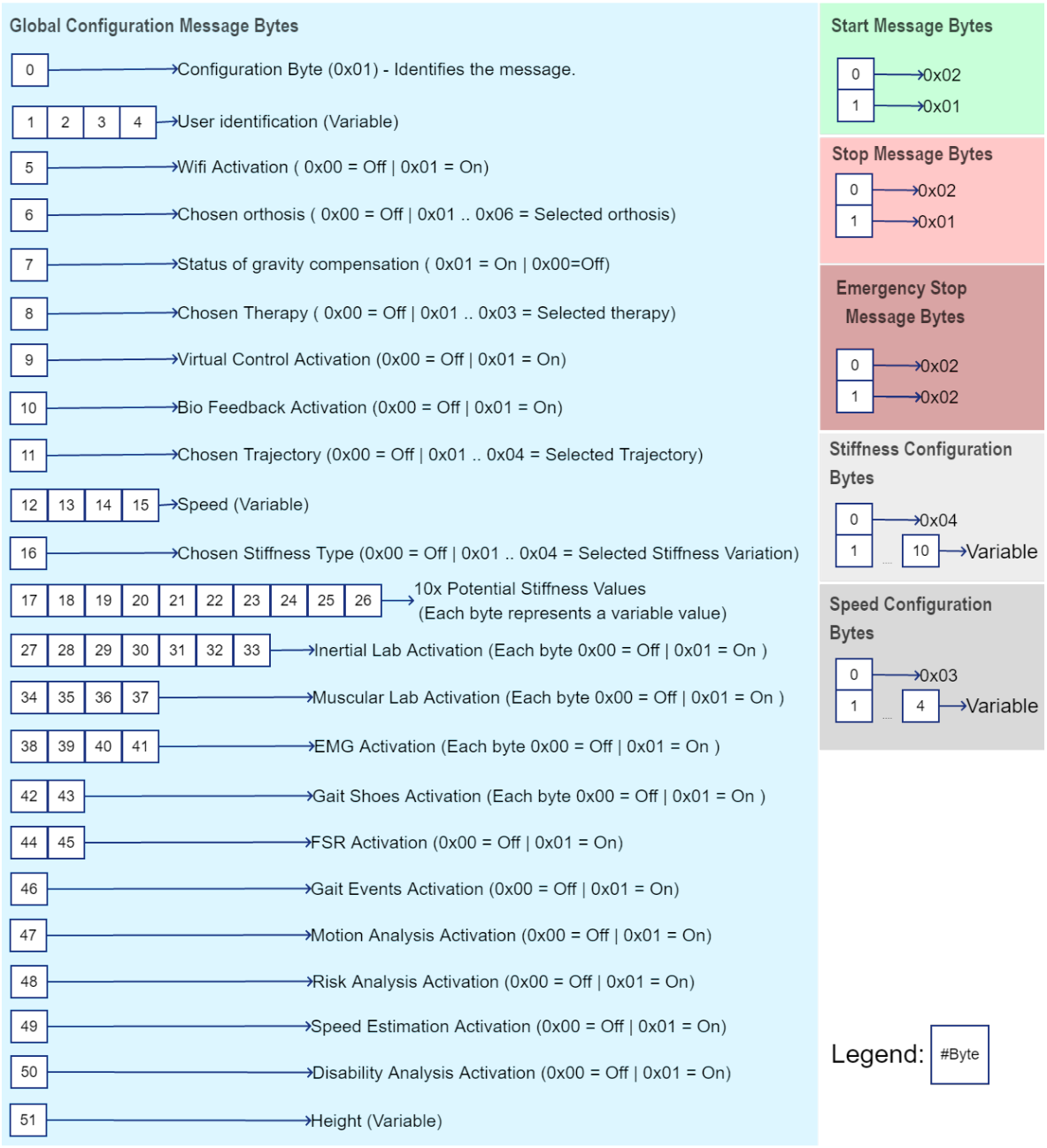


Figure 15.: Labeled S.C.I command messages

2. **Report Messages -**

   Report messages are the messages sent from the C.C.U to the S.C.I, these messages provide information regarding the current status of the SmartOs , ranging from battery information on the orthotic/monitoring devices to the failure or success of the previously defined behaviors such as start therapy, stop therapy, and so on.

a) **Success Messages** - Array of messages that represent a successful action. If no errors occur a command message should always be met with a successful response. All report messages follow a similar pattern, Success Messages are composed of 2 bytes being that the ID is set to [$0x00$] and the second byte categorizes the success.

b) **Error Messages** - This set of messages are sent to the S.C.I whenever an internal error occurs, may this be related to the application of a command or a spontaneous error that might arise amidst the initialization of the systems. The first byte is valued [$0x01$], however these messages make use of 2 additional bytes, having the second byte represent the submodule/sensor that presents the conflict leaving the third and last byte for specification within the submodule.

c) **Warning Messages** - So far only one warning message has been specified regarding the potential fail to initialize the desktop application, which does not pose a threat to the functioning of the system. It uses 2 bytes being its ID set to [$0x02$] and second byte to [$0x01$].

d) **Status Messages** - Lastly, some status messages have been conceptualized, these should give real time information on the system, however only one is yet supported by the low level systems, which would be the battery level. The size of these messages presents a higher degree of variability, the principles remain the same, having a set ID of [$0x03$] and varying subIDs for differentiation.

Report messages are illustrated in Figure 16.

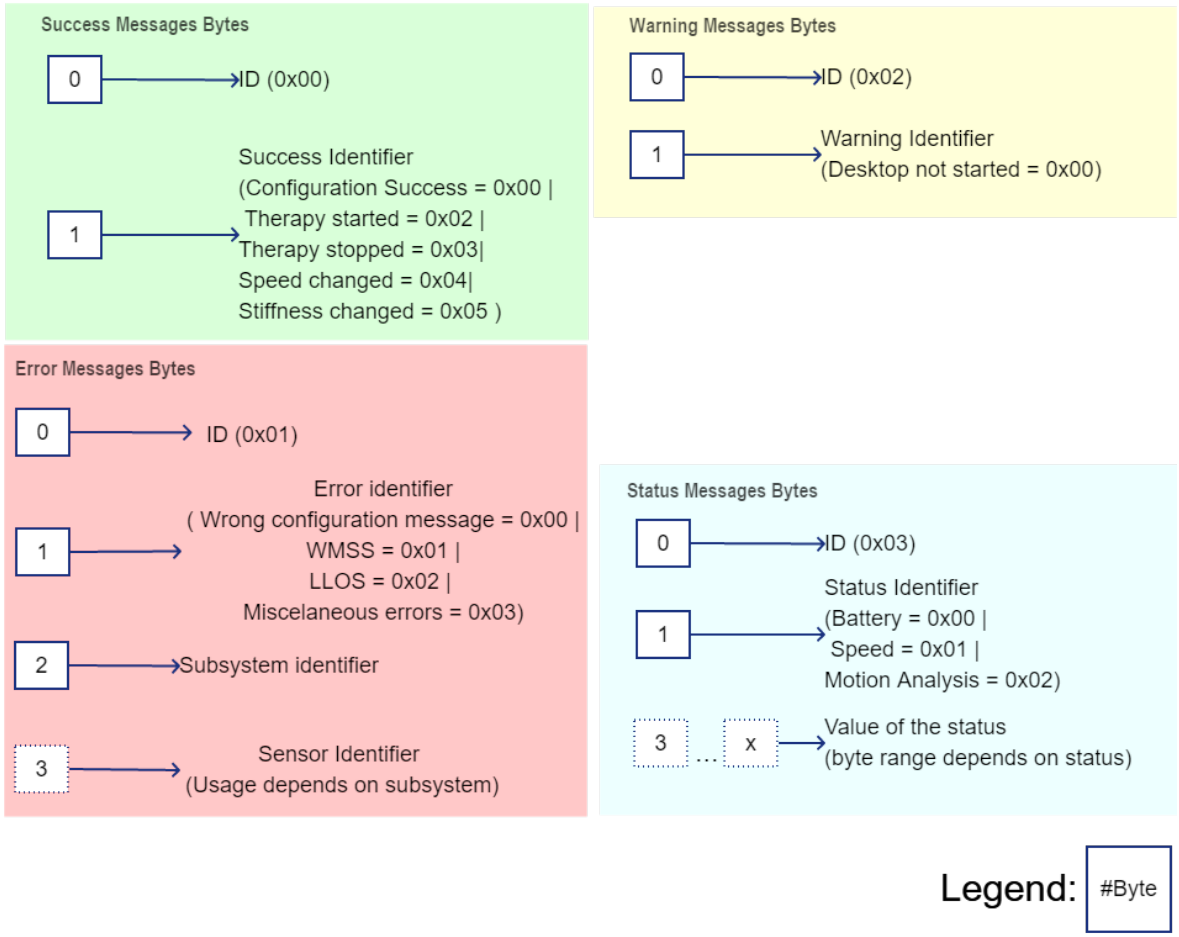


Figure 16.: Labeled S.C.I report messages

4.2 DESIGN AND IMPLEMENTATION

This section is focused on the process of creating the application which will communicate with the C.C.U by addressing the **steps taken to guarantee that the application provides ease of usage whilst enabling to bring out the potential of the system**. Moreover this section will cover the establishment of some guidelines that could help further the development of the ever-evolving SmartOs system, facilitating the addition of new features should these arise in the future.

4.2.1 *Early considerations and technology choices*

As mentioned previously throughout this document, this thesis is an addition to the work of multiple collaborators, as such some development choices can be dictated by input provided from previous work, it is important to review and provide some context on work

regarding the original development choices of the C.C.U and early interfacing concepts of the mobile illustrated in Figure 17.

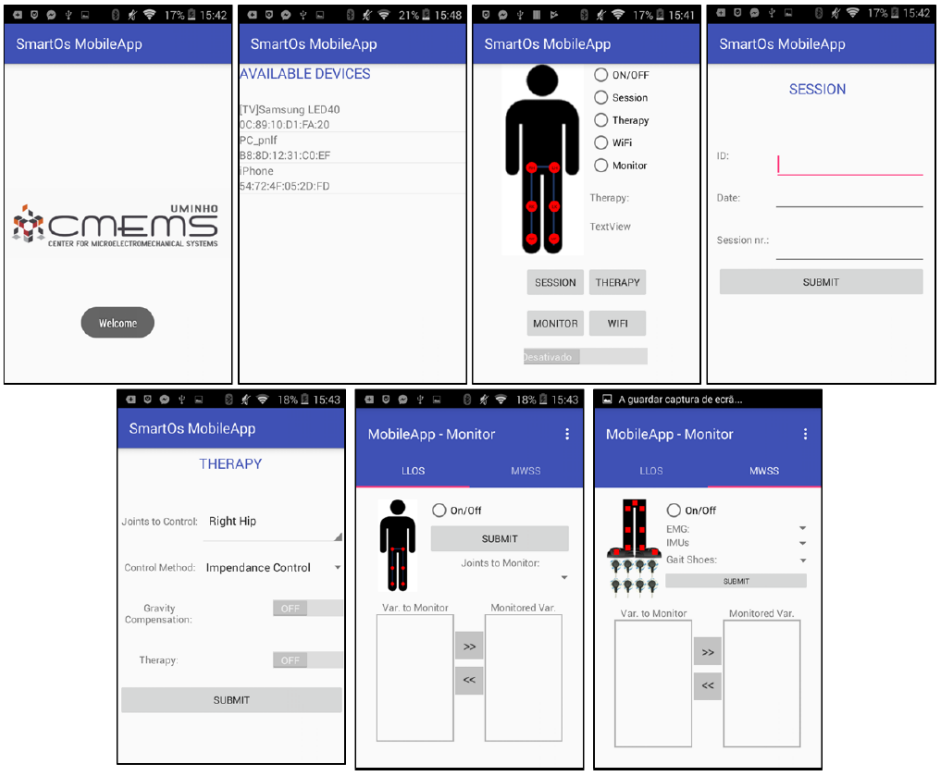


Figure 17.: Early mobile application concepts seen in [28]

Figure 17 shows an **early concept** of a potential mobile application, displaying the capability to configure the whole system throughout the span of six menus.

Whilst these concepts provide valuable insight into the understanding of intended capabilities of the application these settings are presented in a manner which could be much to confusing to be easily usable by someone with little experience with the system, featuring poor guidance whilst interacting with the system.

Additionally, the document that features Figure 17 defines the target platform for the mobile application as being an Android device, this factor was taken into consideration as it influences environment in which to develop the S.C.I.

Multiple frameworks were taken into consideration previous to the development of the software solution, initially React Native[27] was considered as a potential development framework with which to develop the application. Providing the a malleable solution, that allows the targeting of a greater range of hardware as the code produced can be compiled for either Android or iOS. However, S.C.I **relies heavily on the usage of *Bluetooth*** communication which is not officially supported by React Native.

Furthermore, SmartOs uses a dedicated companion mobile device which runs Android, as such the application would be developed in **Java**, the native programming language of the android system, using the **Android Studio IDE**. Thus allowing for the creation of an optimized application whilst granting the best interaction with the *Bluetooth* API.

4.2.2 *Application principles*

There are multiple factors to take into account when developing an application for non-personal usage. More specifically factors that influence development decisions to be made in an interactive and graphical context, both in terms of usability and updatability of the aforementioned application.

The development of graphical interfaces must take into consideration the users of the application. In this particular instance there are **two types of end-users**, that should be addressed with similar degrees of importance, the first being **the clinicians** that could eventually use the SmartOs system as a means for rehabilitation and recovery. Additionally, since the SmartOs **is currently still in development stages** the other group would be the **remaining collaborators** working on the project, who shall be the initial users of the system. Having established the users of the system it is now possible to guide the development of the application as a whole.

Clinicians are non-technical users, possessing little knowledge regarding the operation of the system, it is therefore important that when using the application they are faced with a linear experience. Interacting with the different components of the application should not guide them towards nested menus and hidden options.

Additionally, the application **should not allow its user to perform invalid operations**. This might not be always possible as detailed further in the following subsection but should be prevented to the fullest extent.

Finally, good design should be always **prevent the need for help guides**, however in the case of a **medical application using redundancy to provide extra information** on the usage of the system could be helpful to ensure that the software is correctly used by the most amount of its users.

On the other hand, collaborators are experienced and technical users who might display other needs in regards to interfacing, being especially interested on incoming report messages.

Additionally, visual design and intuitiveness is not the sole concern of this project as the ability to easily update the graphical interfaces and maintain the application when faced with new demands, or even the integration of new submodules might call for the formulation of an architecture that can easily adapt to possible additions of an evolving system.

Ultimately the requirements for the S.C.I are:

- **Streamlined and Guided Interaction -** The application should ensure that the configuration of a therapy session is **fast and uses natural queues** for navigating the menus that might compose it, whilst **permitting all valid forms of configuration**.
- **Guarantee of Safety -** The configuration generated by the application should **never be invalid or a threat to the user or the system**, dependencies should be enforced and signalized and completing a full configuration must require that these are met, invalid configurations should be restricted to the fullest possible extent.
- **Explicitness -** Graphical components used to set configurations should be **well labeled** and use design patterns that are **easily understandable**. Moreover menus should be simple and uncluttered of unnecessary visual elements.
- **Abstraction from Low Level Implementation -** The application should provide **full abstraction** to the technical aspects of the system. The submodules that make up the system and their capabilities must still be known by the clinicians, this is not however the objective of the current dissertation.

4.2.3 *Basic Application Structure and Mockups*

Preceding dependencies and necessities of the system were covered throughout previous subsections, these tasks were elaborated with the sole objective of preparing the development of S.C.I. The steps that would lead up to the creation of mentioned application will be the main focus of this subsection whilst bringing the elaboration of the application development process to a conclusion.

The S.C.I could be conceptually split into three major stages or actions, being these:

1. **Connection Stage -** The first stage of the application, happens at system startup, as previously described the C.C.U of the SmartOs is configured to be able to handle Bluetooth **establishing a server socket to be paired with an external entity**, it is at this time that the S.C.I should try to pair with the C.C.U.

   Besides system startup this process should be repeated whenever a device is disconnected from the C.C.U.

2. **Motorization and Assistance Stage-** In this stage all the possible configurations of the multiple submodules are set.

   This stage is segmented into two components, a **monitoring** component, which targets **activation of the WMSS** subsystem it should be possible to activate this component

without the initialization of any of the **assistance settings** of the SmartOs. The same would not happen in case of the activation of the **Assistance** component, which depends on the activation of the **Monitoring** ability of the system, being that it would be fruitless to start a treatment without recording gait data and the orthoses need sensory data input.

It is also set at the beginning of this stage whether data data should be sent over WiFi to the desktop counterpart for a real-time monitoring experience.

3. **Communication Stage -** It should be at this stage that the parameters set in the previous stage are **transformed into the configuration message** defined previously in 4.1.2 and sent to the C.C.U alongside all the other control messages.

Having defined the multiple stages to be addressed in the application, a navigation guide was created alongside rough mockups in order to establish the main components of the application. These mockups would then guide the establishment of the graphical layouts of the android application.

Figure 18 represents the global navigation of the application, more specifically the multiple Activities the user has to navigate in order to configure and initialize a therapy.

An **Activity** is the **class responsible for user interaction** in Android development it allows the user to **inflate**. I.e. to draw a specific set of graphical components to appear on screen, whilst dealing with any interaction(press, hold, drag) from users and navigation to subsequent Activities.

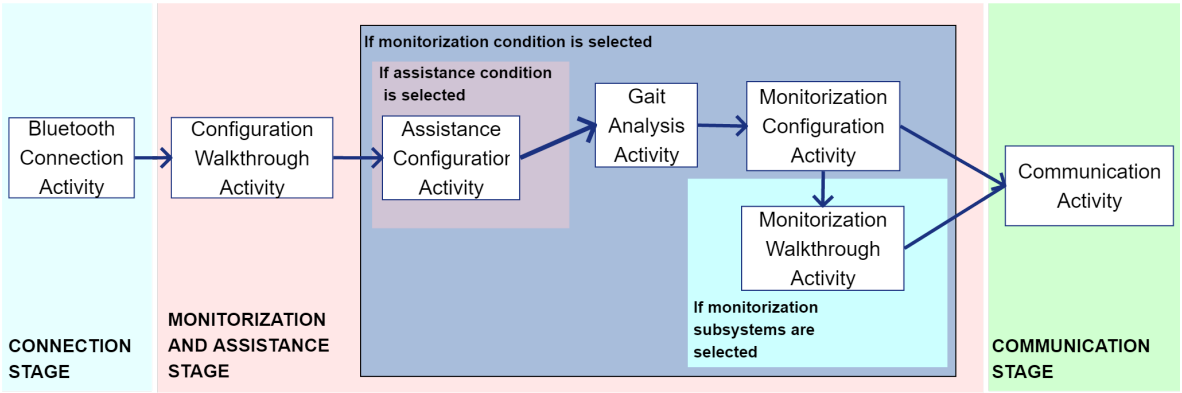


Figure 18.: Application Activities

The categorization of Activities shown in Figure 18 comes in direct correlation to the stages that were previously defined. The first activity should be responsible for the **initialization of a connection** between the S.C.I and C.C.U, followed by **global application configurations** such as monitoring and assistance activation. Should these be set in the Configuration Walkthrough Activity, navigation would be influenced as the user can either

be **allowed** to access **Monitorization and Assistance** Activities alike or proceed **only** to **Monitorization** Activities.

The final Activity would be responsible for dealing with all the Communication UI allowing the user to interact with multiple graphical components in order to alter the state of a therapy by sending the previously defined control messages to the C.C.U.

When programming for Android each of the multiple Activities that make up an application can be responsible for multiple full screen layouts, i.e. each Activity can deal with information originated from multiple UIs(layouts).

Moreover, each layout is made up of a set of other layouts and Widgets declared using specific **Android XML vocabulary**, to guide the process of creation of the multiple layouts the mockups shown in Figure 19 were created.

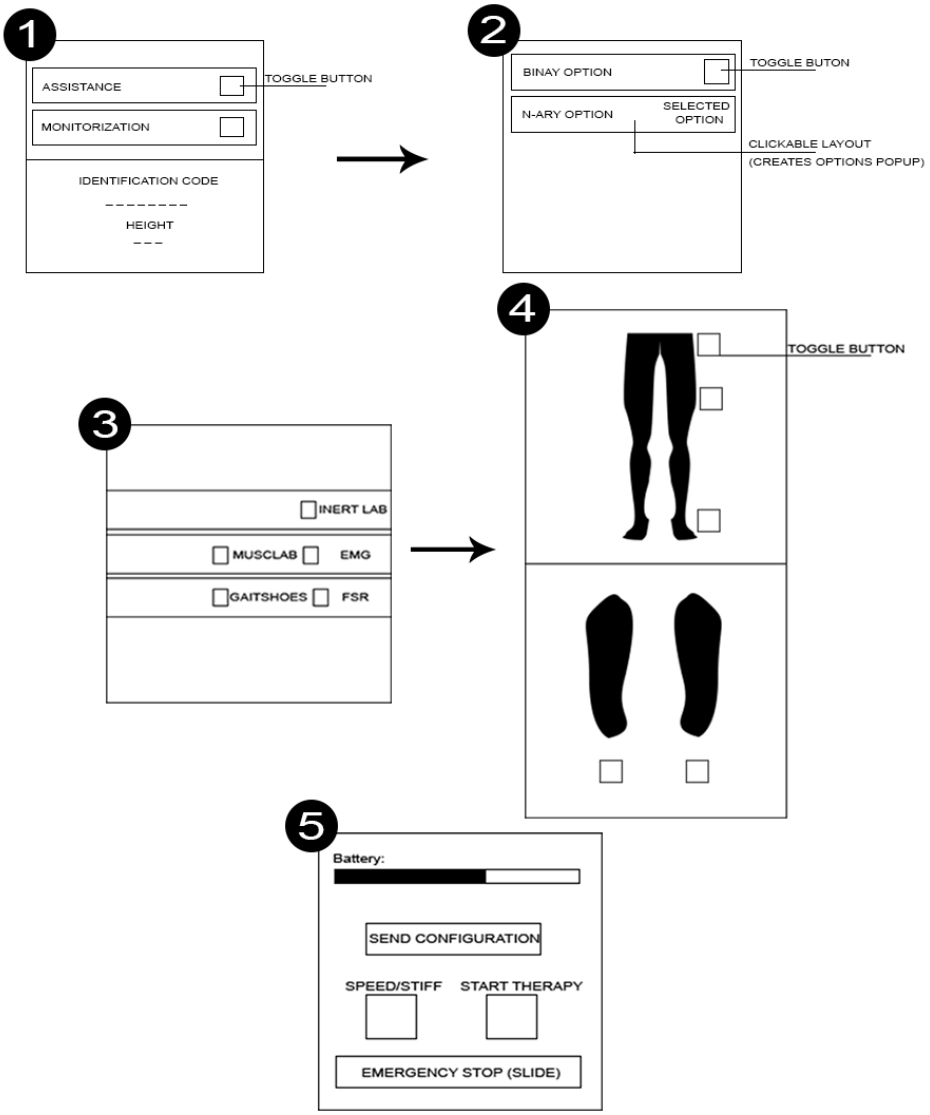


Figure 19.: Applicattion Mockups

These mockups do not represent the end product being instead rough concepts that represent the most important components of the system.

In spite of the low fidelity the images 1 to 5 presented in Figure 19, this illustration can portray the major part of all possible interaction patterns to happen in the finalized version of the application.

For instance image 1 represents the **first two menus** that the user is faced with after connection to the C.C.U. In this menu global system settings such as user identification and height will be set. Additionally, it is at this point that the user **selects the objective of the current therapy** by the **activation of two toggle buttons** representing the selection or de-selection of the **assistance and motoring capabilities** provided by the SmartOs respectively.

With the aid of Figure 18, some additional behavior can now be extracted from these menus, for instance the motorization toggle should be **independent** of the assistance toggle. However, the same should not be true in regards to the assistance toggle, as its selection should **imply the activation** of the **monitoring** toggle.

Furthermore, both the identification code and height configuration are instrumental to the recording and functioning of a therapy. As such, these should be identified as mandatory fields in the final solution and proceeding to the next activity or interface should only be possible if all mandatory conditions are met.

The second menu of Figure 19 presents us with the basic model for setting options. These could either be **binary** and represented in terms of truth and false by a **toggle button** or **n-ary** in which case a custom **pop-up** with varying options should appear upon clicking the second rectangle(internal layout) acting as a regular button.

Once a choice is selected it should be **reflected on the right-hand side of mentioned layout**, having the selected choice written. Otherwise it should be **defaulted with "OFF"**, again, should any field be **mandatory** it should be duly indicated and proceeding should be **interdicted** if conditions are not met.

The third menu aims to establish a relationship with the subsequent menus. There are multiple monitoring subsystems integrated in the SmartOs, represented in the third menu. These should be selected through the usage of toggle buttons, which will in turn dictate the interfaces that will be shown in menu 4.

The options represented in 3 are selected through the usage of toggle buttons, being that the **MuscLab** and **EMG** subsystems are mutually exclusive, and so are the **GaitShoes** and **FSR** subsystems, the **InertialLab** subsystem should not have any relation of exclusivity.

These subsystems will require the elaboration of the **customized interfaces** in menu 4, these five different subsystems are relatively similar, **targeting either the lower body** or the **feet**, in which sensors located in different portions of these body regions should be activated or deactivated with a toggle button.

The fifth and last menu will be dedicated to all communication function, it will receive the status updates such as battery represented on the top of the layout, and the command messages, essential to the activation of the system, should be available with the touch of a button.

Speed and Stiffness control messages, will be managed by a **custom pop-up** that will allow the changing of their **absolute values**.

In regards to the emergency stop message, this will be sent with a slider button as to prevent the sudden accidental stoppage of the system.

4.2.4 *Application Architecture and Navigation*

Up until this point, throughout the chapter, multiple steps were taken in regards to the **preparation and guidelines** to follow with the intent of achieving an intuitive and straightforward experience. These preparations are put into practice through the establishment of a robust architectural foundation.

The development of this dissertation entails the creation of intuitive interfacing for the SmartOs system, but contextually, the work detailed throughout this document is only a part of a bigger system which is, as previously mentioned, **continuously evolving**.

It was therefore necessary to create an architecture that would allow it to be **maintainable** by other individuals, and facilitate navigation between the multiple UI elements. The architecture, depicted in Figure 20, came as a direct result of this necessity.

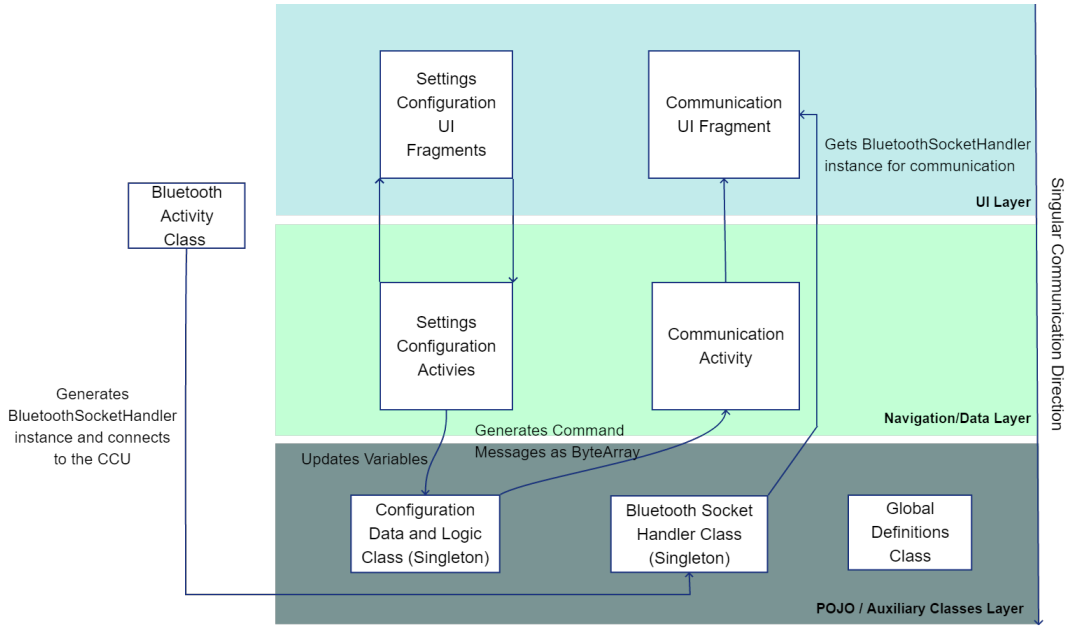


Figure 20.: System Architecture

Figure 20 displays the major class structure and division used in the S.C.I application, it comes in direct correlation to the basic application principles and structure detailed earlier.

Its intent is to provide more information in regards to the concrete organization of the application. Both by disclosing the messaging or interaction that happens between classes as well as defining the auxiliary objects that support the entire logic, navigation and communication of the application.

As it was previously introduced the structure represented in Figure 20 comes as a solution to the **maintainability and refactoring problematic** that arises when developing for an **evolving environment**, to this intent the current project is divided into three major layers.

The first of which would be the Auxiliary Classes Layer, the objective of this layer is essentially to **help navigation and retention** of session Data.

Each of the Classes that make up this layer comes as an aid to other components of the application, the simpler of the three classes that make up the layer would be the *GlobalDefinitions* Class, the objective of this class is to hold the possible options correspondent to binary or *n-ary* settings in the form of **intuitively named variables**.

This class is especially useful to **sanitize the code** of the application by improving its readability, more specifically it **improves the readability** of conditionals present in the system by preventing the usage of literal values.

Using the selection of a therapy, one of the n-ary parameters of the configuration message, as an example, this configurable field can now be represented by its variable instead of its literal value.

This avoids the appearance of undocumented loose values amidst the code-base as these would have self explanatory variable names, using therapy choice as a particular example it could be a selection of one of the following values: **TherapyDisabledFlag = 0 , TherapyPassiveFlag = 1 , TherapyPositionFlag = 2, TherapyImpendanceFlag = 3**.

The *BluetoothSocketHandler* is, much like its name indicates, responsible for holding the *Bluetooth* socket that will be initially instanced in the Bluetooth Activity Class, and ultimately used to send the C.C.U the command messages responsible for the management of a session. This class is created as a singleton to ensure that the communication socket is not closed throughout the operation of the application. It also prevents the passage of the socket variable along the multiple activities that make up the application, living in heap memory. As such, it can be instanced at any point of point of the application without generating a new Object.

The last of the auxiliary classes would be the *ConfigurationDataAndLogic* Class, also to avoid the need of being passed along the multiple Activities of the S.C.I this class is a singleton which **holds all the options of the system** that will make up the configuration message. Moreover, it is essential to the logic of multiple layouts, as some configurations

are dependent on settings defined in previous layouts. This occurrence will be illustrated in the next section when presenting the final application and its interfacing abilities.

In addition of being responsible for holding the application's current configurations the *ConfigurationDataAndLogic* Class also possesses functions to generate the ***bytearrray* messages** that will ultimately be sent over the Bluetooth socket from the Communication UI Fragment.

The last two layers of the application would be the UI and Navigation layers. In spite of their separation in terms of function, these are more easily interpreted in conjunction as they have a great amount of correlation.

In order to achieve a greater degree of malleability when developing the interfaces for Android , the capability of creating Fragments is available from early versions of the Android OS. A fragment is usually part of an Activity and **holds its own layout**, this creates the possibility of, for example, creating an UI made up of **multiple fragments**, **belonging to a single activity**.

As such it is possible to take advantage of this ability not only to diminish the number of activities that make up the application, but also to **increase modularization**, making **portability to screens of different dimensions easier**. This would also influence navigation, allowing the usage of a **walk-trough pattern to navigate** between the multiple Fragments of an activity and divide the application interface for better user experience.

The inputs received by the multiple components of the Fragments are then interpreted and a final answer is sent to the Activity that holds the fragment which is in turn responsible by updating it in the *ConfigurationDataAndLogic* Class. This behavior also allows for the an easy **addition or removal of fragments** and respective interfaces, as the activities possess fewer code.

The process of removing a fragment only requires the deleting of said fragment and consequent removal of code responsible for the Fragment inflation on the Activity that holds it. It should however noted that the messages received from this fragment should also stop being parsed in the respective Activity in favor of code readability.

The process of adding a Fragment is by consequence the inverse, one must create a Fragment and respective layout and user interaction, information messages should be generated and passed on to the Activity that inflates it. This programming practice also **furthers our quest to write maintainable code**.

The *Communication* Fragment and its *Communication* Activity were singled out in Figure 20 as these does not possess the objective of configuration. Moreover, these are an exception to the singular communication direction depicted on the right extreme of the diagram, this Fragment does not have the need for interaction with the Communication Activity as there are no more configurations to be altered.

Furthermore communication is realized **directly from the fragment** as it is easier to send messages directly from the buttons actions.

It should perhaps have been developed in the image of the *BluetoothActivity* Class, where the layout belongs directly to the Activity and no Fragments are used, however there is little probability of alteration of the *BluetoothActivity*'s layout, the same might not be truth for the Communication interfacing, as such preserving the **Fragment** $\leftrightarrow$ **Activity** division might be advantageous should the need to extend the communication functionality ever arise.

4.3 Final Application and Critical Analysis

To conclude the current chapter this section will cover the most important patterns implemented in terms of UI design and their addition to the system, **navigation will also be discussed** in the following paragraphs and finally connected with **previous development decisions**, some other topics mentioned throughout this chapter such as the **settings dependencies**, custom layouts and warnings will also be elaborated further in this section with the help of visual aids from the finalized application.

The materialization of the application and its development should culminate on a critical analysis of its operation and developmental process in regards to the achievement of planed goals(Subsection 1.2) and defined principles(Subsection 4.2.2).

As explained in previous sections, the process of running a therapy is comprised by a **unidirectional flow of navigation**, this simple flow allows the user to avoid the occurrence of mistakes, as the start of a therapy session is comprised by a set of steps or menus, each of which **having consequence in the creation or omission** of future menus and features, going back on a given menu implies that forward steps are reseted as to not compromise the state of the control message.

Figure 21 illustrates the the process of preparing and initializing an entire session for monitoring and assistance, as it has been reiterated throughout this document the process is linear. **Navigating between activities is not bound to the user but rather the application itself**, the navigation to different Android Activities is realized by tapping the multiple proceed Buttons like the one found on the bottom right corner of the third menu shown in Figure 21.

Furthermore, going back on an activity will automatically discard any configuration up until that point in order to prevent the user to break configuration dependencies, these dependencies are shown to the user using various methods.

Items marked with an asterisk are mandatory, meaning that items such as the "**Orthosis**" in menu 4 **must** be set, i.e. must be different than "Off", in order to proceed.

In fact the ”proceed” button is disabled until all mandatory conditions, represented in the interfaces by an asterisk, are met. This action is displayed by transforming the colors on the letters of the button **from grey to white** henceforth **representing its transition from a non-clickable to a clickable button**.

Other dependencies are implemented by **hiding non-relevant** items, for example the Trajectory item is only represented in the fourth menu if the Therapy is set to a mode named ”Position” or ”Impendance”, if Therapy is set to ”Passive” the Trajectory item **would not show up** as it is irrelevant **and setting it would break proper configuration message logic** for that scenario.

By doing this we are effectively limiting the user's ability to commit errors whilst allowing all valid combinations of the configuration message, guaranteeing the security of the system and the patient when establishing a therapy.

An example of a basic therapy configuration is presented in appendix A with the aid of a video, detailing the communication procedure between S.C.I, C.C.U and the S.M.I.

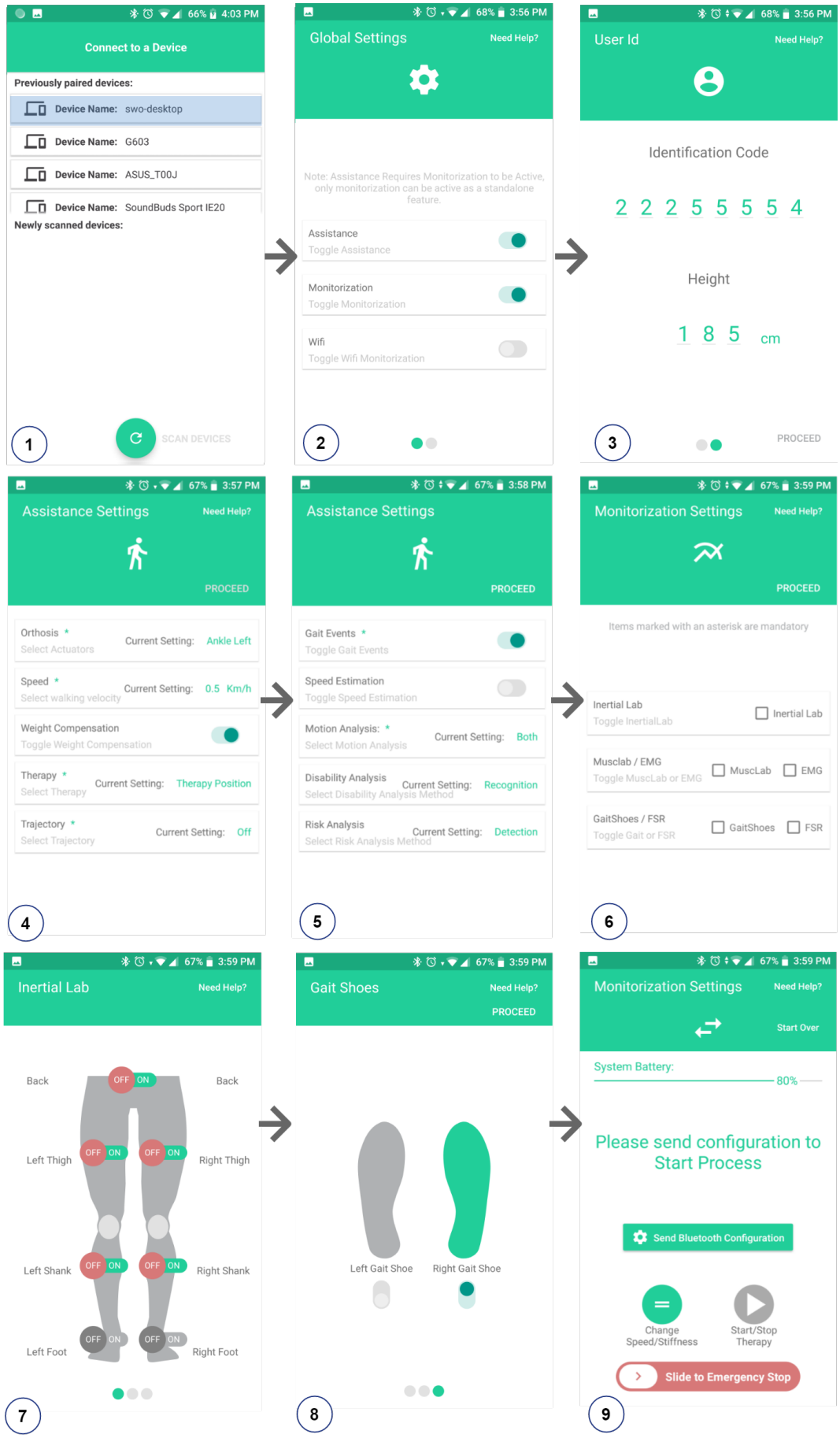


Figure 21.: Therapy Process Menus from beginning to end

Another topic mentioned previously in this chapter would be the existence of **different types of options** for the clickable items showed in Figure 21 figure, more specifically in menus 4 and 5. Whilst **binary items** can be activated or deactivated with the click of a **standard toggle button**, not all therapy settings can be set with binary logic.

The first menu of Figure 22 represents the **custom popup** generated to handle **multiple/*n-ary*** settings, in this specific case it is employed in the selection of the Orthosis. However, not all menus can be represented with a limited set of options, such is the case of both the **Speed and Stiffness items**, represented in the second and third menus, these options are represented by **absolute values**.

In the case of speed option the value is represented as a text label being alterable via the minus and plus buttons by subtracting or adding offsets of 0.1km/h respectively whilst being limited to a range of values(0.5km/h - 1.6km/h).

The third menu represents the alterable stiffnesses of the mechanical joints, ranging from 5 to 10 customizable stiffness values. Sliders were used to allow the user to promptly set the variables, additionally, stiffness values range from 20% to 100% and default Android sliders do not display absolute values. To ensure good user experience third-party, open-source sliders were employed in order to allow the **tear-drop effect seen displaying the current value in the last menu**.

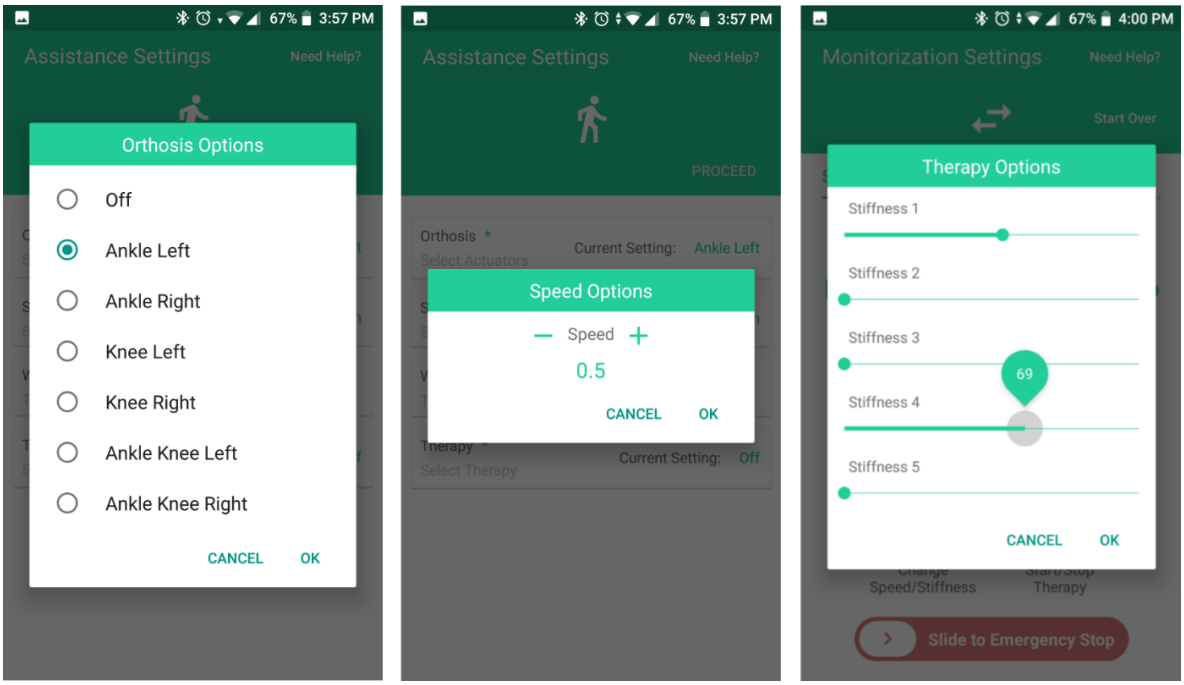


Figure 22.: N-ary options and Absolute Value options

Falling back on the topic of navigation and user experience in Figure 23 the **walk-through pattern** is used to navigate between multiple fragments of a same activity, this facilitates

the usage of Activities as a means to achieve code organization as an Activity can represent a global thematic.

In this case the Global Settings of the S.C.I, being composed of multiple individual fragments this allows for a **natural navigation** with a swipe much like turning a page. The "current page" is **represented by the green dot on the bottom**, navigating back and forth between these fragments **does not reset variables of any kind as these pertain to a singular Activity** although they represent different menus.

Once all necessary conditions are met the hidden button will show up on the bottom right corner of the last "page", **this enforces the visualization of all necessary settings before proceeding** as well as natural feeling of the standard left to right navigation seen on most applications.

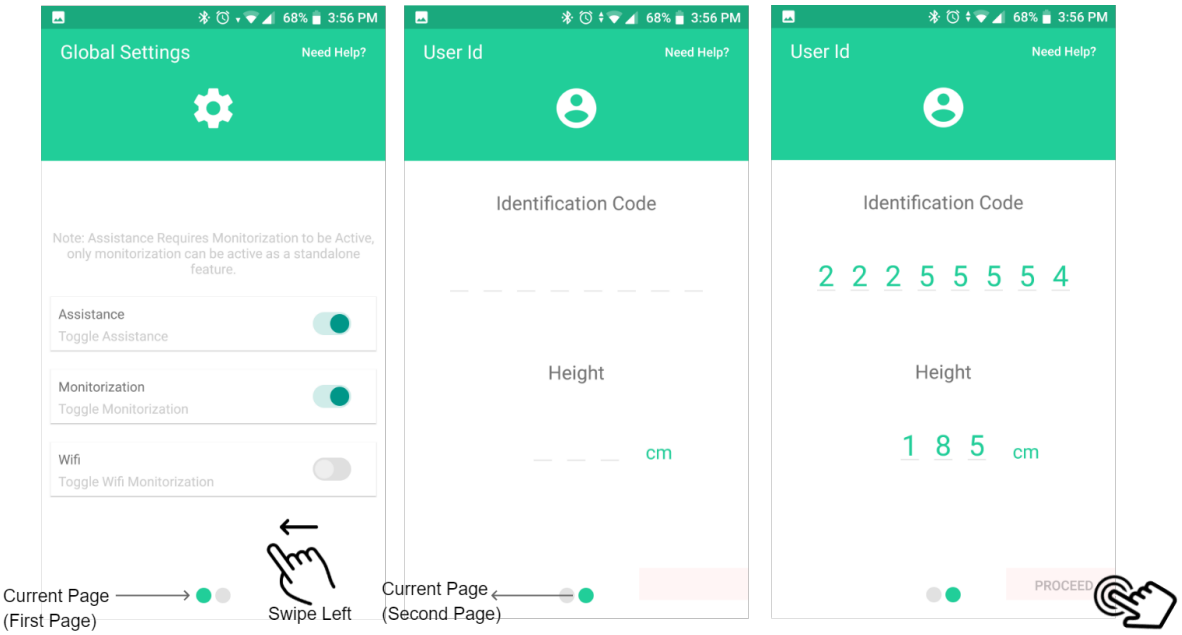


Figure 23.: Walkthrough pattern and guidance tactics

Another challenge in regards to the development of the current application would be setting the monitoring process of the SmartOs. The aforementioned process is comprised of 5 subsystems, **the InertialLab, MuscLab, EMG, GaitShoes and FSR**.

To bootstrap the process the systems to be activated should be chosen in an **overall configuration step**, the first menu in figure 24.

The toggle button for the InertialLab stands alone while all other toggles are coupled, this layout structure comes as a direct consequence of the fact that MuscLab and EMG are alternatives for a same process, much like the GaitShoes and FSR, in the case of the InertialLab there is currently no alternative to this system. Moreover, **systems with alternatives**

are mutually exclusive, as such the application will only allow one active toggle for each of the alternatives.

The choice of the subsystems will ultimately dictate the fragments to be displayed in the following activity in the form of yet another Walkthrough design pattern. These Fragments hold custom layouts that are either represented by a set off toggles in the lower body region or in the case of the GaitShoes and FSR, these sensors target the feet, and can be activated by either by clicking either of the feet images or their respective toggle button.

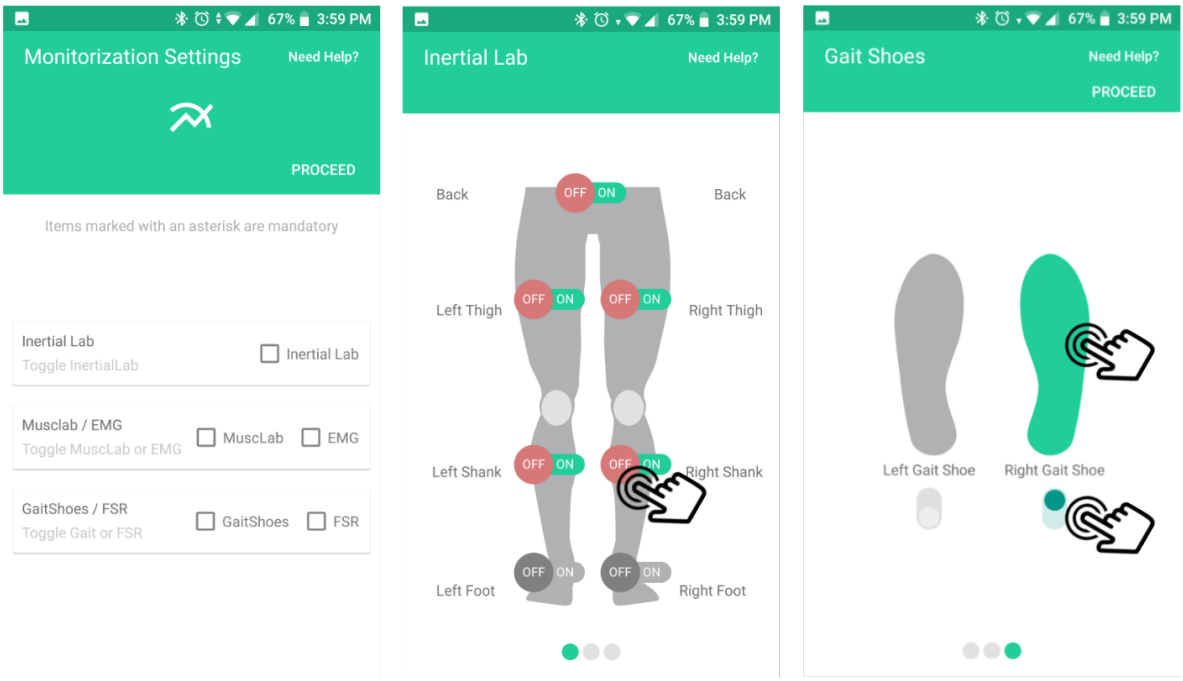


Figure 24.: Custom Monitoring Layouts

So far all components of the application were established having in mind the necessity to create an experience that is straightforward and intuitive for the user, however given that this application is meant to be used in a clinical context and should any doubt be had, the menus of the system explain interaction with the help of a third party library, Spotlight, shown in Figure 25, **prompted each time the always present "Help" button is clicked**.

It should also be noted that the help given in the Spotlight pertain to the graphical components and usability of the system as the functionality and repercussion of each option is related to the medical field and should be explained when introducing the system to the clinician, not being the focus of this Thesis.

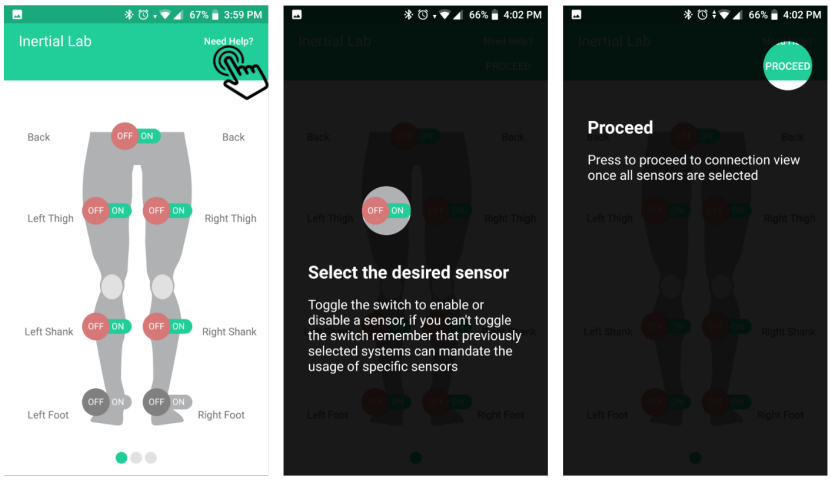


Figure 25.: Help Button and Spotlight

The last main process to have into account in regards of user experience would be the process of altering the therapy status, this process is brief and straightforward. Initially the user should click the "Send Bluetooth Configuration" button as illustrated in the first menu of Figure 26, the C.C.U will then process the message and send **either a success or error message**, in case of success the "Start Therapy" button **is enabled** and clicking it will start the therapy process. This process can then be stopped by pressing the button again or if need be an Emergency Stop message should be sent by using the slider shown in the bottom of the third menu. This message comes as a redundancy given that the system possesses a physical failsafe and having a software based emergency stop could be fruitless in case of system malfunction.

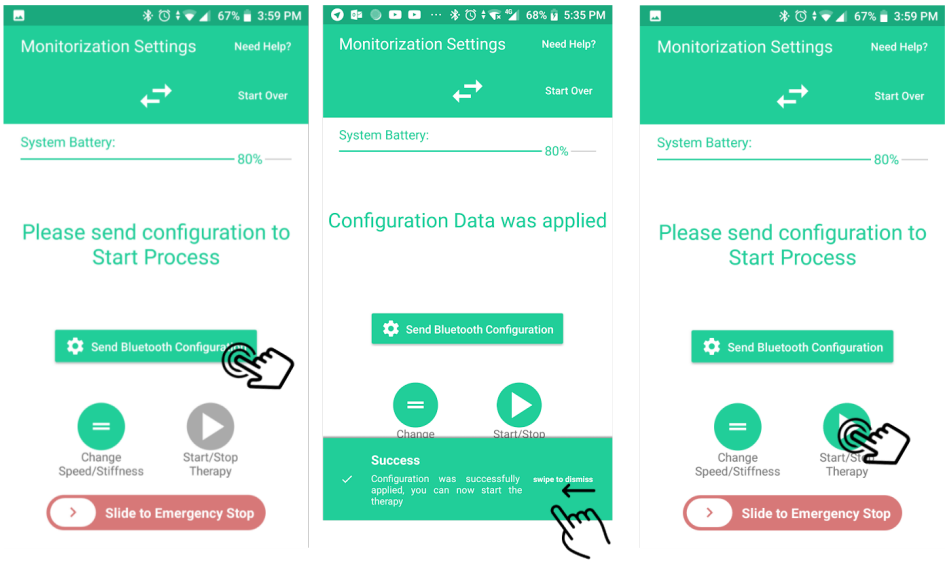


Figure 26.: Communication Process

Summing up, the S.C.I achieves **full control over the SmartOs system** whilst retaining a set of interfaces that should be **easy to grasp and follow by any user due to its linear nature**. Having already been used with success by multiple other members that participate in the SmartOs project in order to test the multiple components of the evolving SmartOs environment.

The messages elaborated previous to the development of the application have been up to this point established well enough to ensure that modification of any kind was not needed. In regards to **code application its maintainability or upgrade** by other members of the project remains to be tested, but the architecture of the system was designed so that effort in removal or addition of interfaces is minimized, the only compromise to this rule could be the logic segments that enforce mandatory options through the usage of previously defined variables.

Lastly, some of the choices regarding the C.C.U architecture can limit the ease of use of the interfaces developed, one of such examples would be **the need for selection of monitoring systems for activation**. This feature could be hidden to the user if connected systems were announced from the C.C.U to the S.C.I removing the need for manual selection of subsystems.

Another shortcoming of the system is the reliance on manual input of the user identification which could be addressed by introducing **online features to the system**. This would, however, breach the objective of the system as a portable unit that is usable under off-line conditions as well as the potential to have a direct connection to the real-time monitoring application that, as will be shown in the next chapter, makes use of the WiFi chip on the C.C.U to establish a communication channel.

4.3.1 *Application Validation and Testing*

The objective of this subsection is to provide some investigate the robustness of the application in regards of dependency enforcement and error prevention using technical users.

Additionally, **a questionnaire to demonstrate the intuitiveness and ease of navigation** within the application was taken by non-technical users to ensure that the proposed principles of the application are met, and usability in a real world scenario would be facilitated by the introduction of the present tool.

Figure 27 introduces the process of navigation within an application. This procedure ensures that the logic of navigation is correct and implements the streamlined and safety primitives that were set as requirements of the S.C.I.

As introduced in the legend of the Figure 27, blue squares indicate system actions, whilst pink squares indicate input from the user. The linearity of the navigation within the ap-

plication is reached by establishing the circular pattern of interaction, **User Interaction** $\leftrightarrow$ **System Actions**, depicted in the Figure 27.

Whenever a new android Activity is started the system checks for mandatory dependencies, whenever the user sets a configuration from one of the activities, the system will then **check if all mandatory dependencies are met**. Only then is the proceed button presented or activated to allow navigation to a different Activity. This process is **repeated** up until the last Activity of the android application, the Communication Activity.

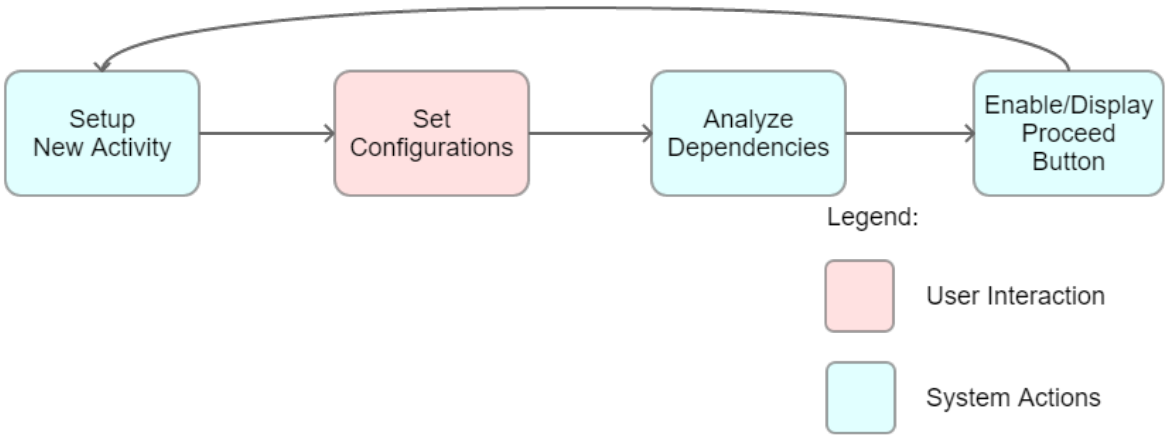


Figure 27.: S.C.I. interaction Flowchart

While it is important to streamline navigation by enforcing dependencies and preventing errors. The process of developing this logic required great input from peers with profound knowledge of the low level systems required for a therapy.

As introduced in subsection 4.2.4, whilst an android application is composed of multiple Activities, Fragments were used to implement the GUI to be presented to the user.

To accelerate the process of development all Fragments of the system were developed based on rough mock-ups, input from peers, and configurations available on the previously defined configuration message.

Once **all possible configurations were set and verified**, a mechanism used to ensure quality of the configuration procedure was started. This mechanism is represented in Figure 28.

After coordinating with knowledgeable peers, the dependencies that should be generated for configured options would be collected.

These dependencies would then be enforced in the application. Leading to the creation of **custom interfaces** in the application or guide the user towards the selection of a **mandatory choice** based on previous selections.

After each modification the system was thoroughly debugged in an emulator and ultimately **tested in a real-case scenario** by connecting the application to the C.C.U of the SmartOs system and executing a therapy in a controlled environment.

This would also allow to **guarantee the validity of the multiple messages** sent to the C.C.U, and to expeditiously correct errors that might have been overlooked in the application.

This process was repeated until the application was considered ready and safe to be used in any possible setup of the SmartOs system. Allowing the correction of details that might have been overlooked by peers or the addition of modules to the system contemporaneous to the elaboration of the application.

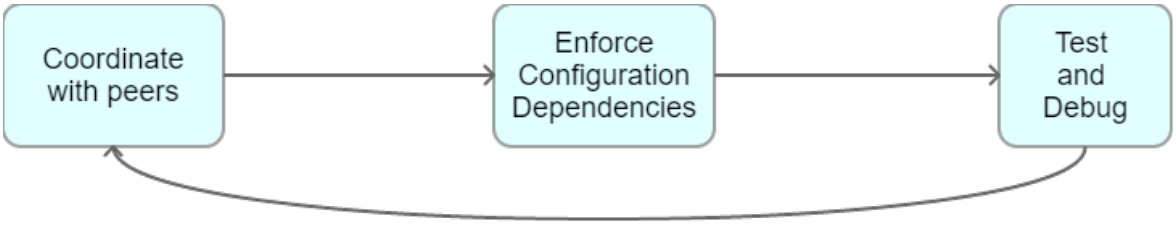


Figure 28.: S.C.I. development feedback loop

The previously presented process of error correction and dependency enforcement promoted the creation of an application that is safe to use and **does not incur in unwanted behavior**.

However, whilst interfacing patterns were used with the intention of encouraging intuitive navigation of the application, a standardized questionnaire, System Usability Scale (SUS)[1], was used to assess the usability of the system. This questionnaire is commonly used on the domain of application testing.

The SUS presents 10 questions to an user and employs an agreement scale. The possible values of the scale are 1 (Strongly Disagree), 2 (Disagree), 3 (Neutral), 4(Agree) and 5 (Strongly Agree) for each of the questions.The items of the questionnaire can be presented as follows:

1) I think that I would like to use this system frequently.

2) I found the system unnecessarily complex.

3) I thought the system was easy to use.

4) I think that I would need the support of a technical person to be able to use this system.

5) I found the various functions in this system were well integrated.

6) I thought there was too much inconsistency in this system.

7) I would imagine that most people would learn to use this system very quickly.

8) I found the system very cumbersome to use.

9) I felt very confident using the system.

10) I needed to learn a lot of things before I could get going with this system.

As can be observed **items 1, 3, 5, 7, 9** benefit from higher levels of agreement, whilst **items 2, 4, 6, 8, 10** should benefit from lower scores. A scoring system was also developed for establishing a 0-100 score for the SUS.

The answers to **each of the odd items should be subtracted by one**, for pair items the result should be the **subtraction of the answered value to the maximum value of the agreement scale (5 - Strongly Agree)**, results should then be summed and ultimately multiplied by 2.5 to obtain a 0-100 score. The formula for calculating the final usability score is shown in 1 ,in which Qx represents the result for item number x of the Questionnaire.

$$SUSScore = \Big(\big((Q1-1)+(Q3-1)+(Q5-1)+(Q7-1)+(Q9-1)\big)+ \big((5-Q2)+(5-Q2)+(5-Q4)+(5-Q6)+(5-Q8)+(5-Q10)\big)\Big) * 2.5 \quad (1)$$

The obtained score does not represent, however, a percentage, as such a study was developed [10] in which the scale shown in Figure 29 was created, allowing to interpret and grade the obtained 0-100 score.

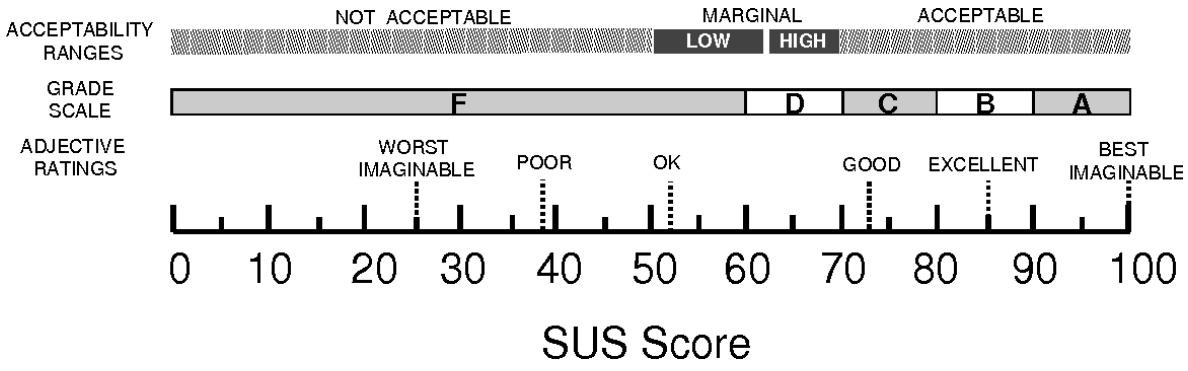


Figure 29.: Grading Scale for the SUS

Once the parameters of the SUS were established the application was presented to 8 individuals of different backgrounds or technical insight in regards to the SmartOs. The individuals were composed of **6 males and 2 females**. Five of the males belonging to the 20-25 age group, the remaining male being in the 40-45 age group. The female participants

were also integrated in the same age groups as the males, the first being aged in the 20-25 group and the second in the 40-45 age group.

Moreover it is important to reiterate that the subjects had **no relation to the clinical field** as such it was asked that aspects related to the technical meaning of each of the configurable options should be disregarded. These options were created and established by dialog with previously mentioned peers integrated in the project and are not the focus of this dissertation.

A brief explanation regarding the intended usage of the application was given to the users of the S.C.I, after which they were asked to carry out the configuration process for a therapy.

Table 2 shows the average individual scores of the SUS questionnaires along with the best and worst ranking items in average.

Table 2.: SUS scores for the S.C.I.

	Score
Subject 1	85
Subject 2	87.5
Subject 3	82.5
Subject 4	82.5
Subject 5	85
Subject 6	75
Subject 7	80
Subject 8	92.5
Average	83.75

Lowest Scoring Item(s)
Item 1: 2.75; Item 4: 2.875

Highest Scoring Item(s)
Item 2: 3.75 ; Item 6: 3.875

After applying the the formula for calculation of the SUS scores these were averaged. The **average score for the tests is 83.75**, this score implies that the degree of usability for the application is on average in the upper limits of the ”GOOD” to ”EXCELLENT” range shown in Figure 29.

These results display **great potential in regards to the usability of the application**. Moreover the worst evaluation still evaluated the application with a 75 in the usability scale, no other evaluation dipped under the 80 score.

It is also worthy of mention that one of the lowest scoring items would be **1)** scoring 2.75 out of 4 after value normalization. However, it could be argued that the dissociation from the individuals from the practice of therapy could be a **diminishing factor in regards to the frequent utilization of the system**.

It should also be noted that higher normalized values always represent better values, independently of the item in question.

Item number **4)** also produced at 2.875 out of 4, the only other item that scored under 3 after normalization. It is once again possible that the low score is related to the lack of

knowledge in regards to the medical field and the capabilities of the orthotics and sensors when presented with a great amount of configurable settings.

Moreover, **all users present in the test concluded the configuration of a therapy session with success**, having items **2), 6), 8), and 9)** scored values over 3.5 out of 4 after normalization. The high score in these particular items might demonstrate that the system is a facilitator to the operation of the SmartOs, indicating high levels of trust in the system and ease of operability, a possible consequence of the design choices and easy flow emphasized throughout this dissertation.

Each of the participants was also asked about possible alterations that could improve the operation of the S.M.I. The responses were valuable in preparing the application for a real world scenario as a couple of overlooked details were evidenced and ameliorated in the applications as follows:

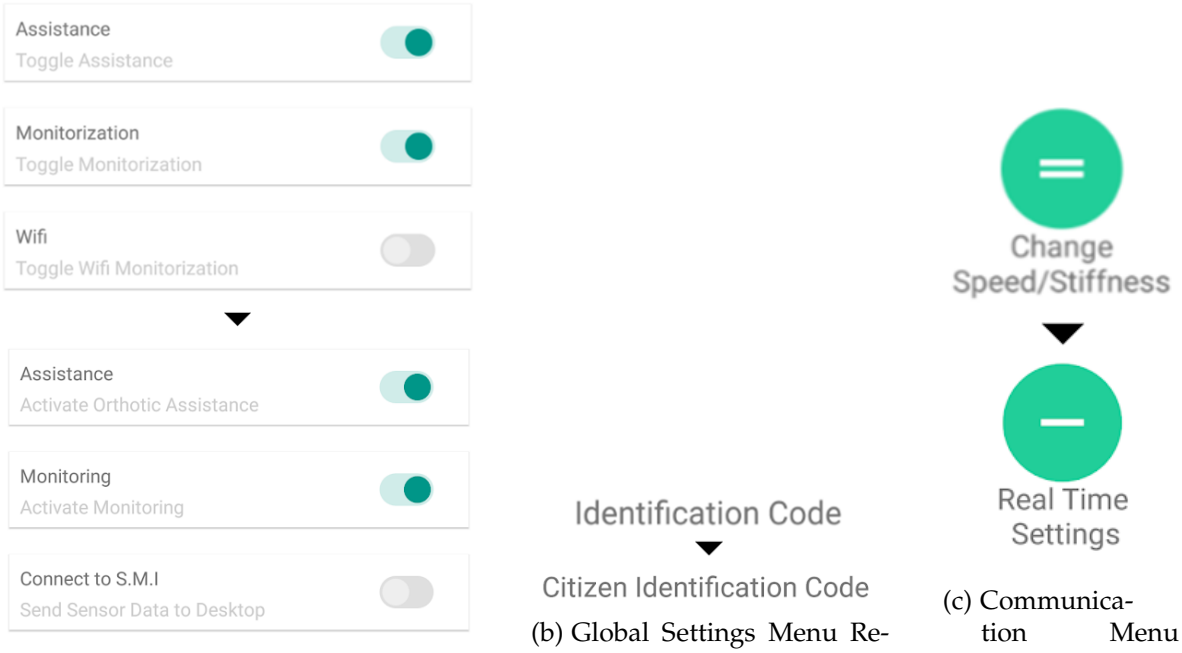
- **Mislabeled or poorly identified elements** - Figure 30 is composed of three subfigures with modification to text aids that generated confusion or entropy in the creation of a therapy session.

  Not all elements in Figure 30a were readily understood by individuals who underwent the process of therapy creation. The subtitle for the **Assistance component was relabeled in order to indicate it uses the powered orthosis**.

  The previously labeled ”WiFi” button was changed to indicate that **its function is to send the acquired monitoring data in real time to the desktop counterpart**.

  Previously identified with ”Identification Code”, Figure 30b displays the changed label to better specify the intended identification code in this case the ”Citizen ID”.

  The button shown in Figure 30c creates a component with all the settings that can be changed in real time, it was labeled ”speed/stiffness settings”, not providing further information on when it should be used. This relabeling is more **efficient in regards to horizontal space** and should be more intuitive for the users who should know the capabilities of the hardware and as such the intended functionality for the button.



(a) Global Settings Menu Relabeling

(b) Global Settings Menu Relabeling

(c) Communication Menu Relabeling

Figure 30.: Relabeled components

- **Poor dependency signalization** - Some menus **lacked the information in regards to the usage of an "*" to indicate its mandatory status**. It was therefore added to all menus that enforce dependencies.

  While this practice is fairly standard, the indication "Items marked with * are mandatory" shown in Figure 31 in bright green text should provide attention and aid users navigating the system more promptly.

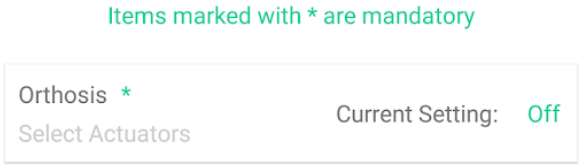


Figure 31.: Mandatory text update

Help buttons specify the existence of these dependencies for each menu, however these were not always clicked. As previously explained, these come as a redundancy and menus should be self-explanatory, these tests also imply that it is not likely that the users use these buttons even if faced with adversity.

The application was also tested in multiple emulated roms and devices, ranging from Android 5.0 to 8.1.

No inconsistencies were found over the multiple tested devices and layouts adapted **correctly to different screen resolutions**. The application **is not computationally heavy** performing adequately even when running on low-end devices.

Moreover, the companion mobile device for SmartOs runs Android 5.0, at a resolution of 720x1280 pixels with a 4.5 inches screen. Although it could eventually be upgraded to allow for better visualization of the multiple configurable options. These requirements would be the **minimum recommended for operating the S.C.I**. Furthermore, the minimum Software Development Kit (SDK) version set , i.e. the minimum Android version to be supported was set to be the same as the companion device.

5

SMARTOS MONITORING INTERFACE

This chapter will cover all developmental stages of the creation of the S.M.I, as its name indicates the intent of this application is the **visualization of data** generated from therapies ran on the SmartOs system acting as an information hub. This component of the SmartOs system does not possess any control over the C.C.U, as such the S.M.I could be considered a complement to the S.C.I, its main goal is to provide real-time information in regards to currently active therapy sessions through the usage of **human readable visualization models for the multiple sensors** imbued in the SmartOs.

The S.M.I should also be able to manage the multiple therapy sessions that make up the users therapy sessions, either by saving a full session transmitted by the C.C.U or by importing session data. Ultimately the process should be as autonomous as possible in order to reduce interfacing complexity without restraining usability.

5.1 CENTRAL CONTROLLER INTEGRATION

Similarly to the problematic faced in section 4.1 a necessity existed for the creation of a communication interface. However, due to the data rate in which the multiple systems are polled by the C.C.U the implementation the S.M.I communication channel has been planned from conceptual stages of the SmartOs system to be established over WiFi.

The RPI3 unit that makes up the C.C.U has the hardware capabilities to elaborate such a task, being equipped with a **wireless LAN chip**. However, the system has been designed to work independently of external hardware such as a router. Additionally, the connection to a wireless network may not be achievable as the C.C.U itself will be strapped to any individual who undertakes a therapy during which keyboard input and graphical output is not possible. The RPI3 has, however, the ability to act as a **portable hotspot**, generating its own wireless network that can be connected to by any device with wireless capabilities.

Moreover, there is no code implementation in the C.C.U for wireless communication in the SmartOs project and a message protocol that will enable the parsing of sensor data is not yet defined. These are the challenges that will be covered in the current section.

5.1.1 *Design and Integration of Wifi communication*

Contrary to the configuration of the system to allow the usage of Bluetooth protocols the current distribution already possesses the necessary libraries to allow for the establishment of a network protocol, which is independent of its method of wireless or wired transmission.

As such using once again socket programming standards it is possible to establish a channel using a network communication protocol. In the case of the SmartOs the protocol to be used would be the TCP protocol, and the process of socket communication would be identical to the one described in subsection 4.1.1. Entailing the creation of a TCP server socket that would listen for a client connection and consequently a communication channel is created. However, the establishment of a server socket is not enough as a known IP address must be given to the client in order to specify the server to connect to within a network.

Furthermore, this would require the usage of a static IP address. Another challenge of creating the wireless connection would be the aforementioned necessity for establishing a wireless hotspot on the C.C.U. The solution to the latter would consequently result in solving the server IP address problematic. Taking advantage of the popularity of the RPI3 and its community, a tool for creating a configurable portable access point, **rpi3-hotspot**, available in a public github repository[15], was used. This tool effectively transforms the C.C.U into a wireless hotspot. Additionally, a configuration file(located in **/boot/hotspot.txt**) **allows to set few, albeit useful networking settings** as it allows the setting of a custom name to the newly created network and corresponding password as well as the **static IP** address to be given to the hotspot server.

The process of wireless communication culminates in the establishment of classes parallel to those demonstrated in subsection 4.1.1. A generic class for WiFi Communication implemented with a singleton design pattern is generated allowing for the **input and output of data** over a TCP socket, being independent of context and abstract.

Once again an External Device derived class is implemented, in this case the *DesktopApp* Class which can in turn be added to the existing code and used in a dedicated thread.

Although parsing is not needed in the context of this module as messages are sent from a *Monitorization* class created by other members of the project. The data is then transformed into a valid format and sent over to the desktop application. Given the nature of the S.M.I, **no input is predicted** in regards to messages incoming from the desktop to the C.C.U.

5.1.2 *Establishing a Communication Standard and Data Marshaling*

In this subsection the conception of the multiple messages that make up a session will be addressed. In the case of the S.M.I there will not be multiple different classes of messages.

In fact all messages that are sent from the C.C.U to the desktop application could be classified as **Monitoring Messages**, as such the level of complexity of the communication standard is diminished, however the multiple messages sent over to the S.M.I pertain to different systems and and have different objectives.

Additionally there is a notion of order to the aforementioned messages. In fact, a full stream of communication could be divided into **three distinct sections**, a starting message, followed by all data messages from all the sensors and timer, and a message for finalizing the current session. The messages that compose the a session can be therefore described as follows:

- **Configuration + Start Message** - The first byte of this message corresponds to its ID, in this case $[0xfe]$, this message is comprised by **all the bytes present in the *Bluetooth* configuration message** established in the S.C.I chapter, prepended with the previously mentioned ID byte and appended with a separation byte,$[0xff]$, in order to aid the parsing of the messages stream.

  This acts as both a **configuration message and a starting message**, as it will help define the graphical components to be rendered on screen and is sent whenever a therapy starts, the reasoning for this behavior will be explained further throughout this section.

- **Sensory Messages** - These messages are comprised by 7 bytes, the last byte being the $[0xff]$ closing byte. The first two bytes are used as an ID and subID as each ID will represent a specific system from the SmartOs whilst the subID represents the specific sensor or event from that subsystem, **the remaining bytes will be used to identify the value collected from the active sensor or event**. The representation of the multiple sensory messages will be discussed in the last section of this chapter.

- **Timer Messages** - Timer messages possess the same structure as Sensory messages. Both the ID and subID of these messages is valued $[0xfd]$. The remaining 4 bytes are used to send a float representing **elapsed time**.

- **End of Session Message** - This message represents the **end of a therapy session**, for convenience its structure is the same as that of the sensory and timer messages, however only its ID byte is parsed, being valued $[0xfe]$.

Figure 32 was elaborated to ease the interpretation of monitoring messages.

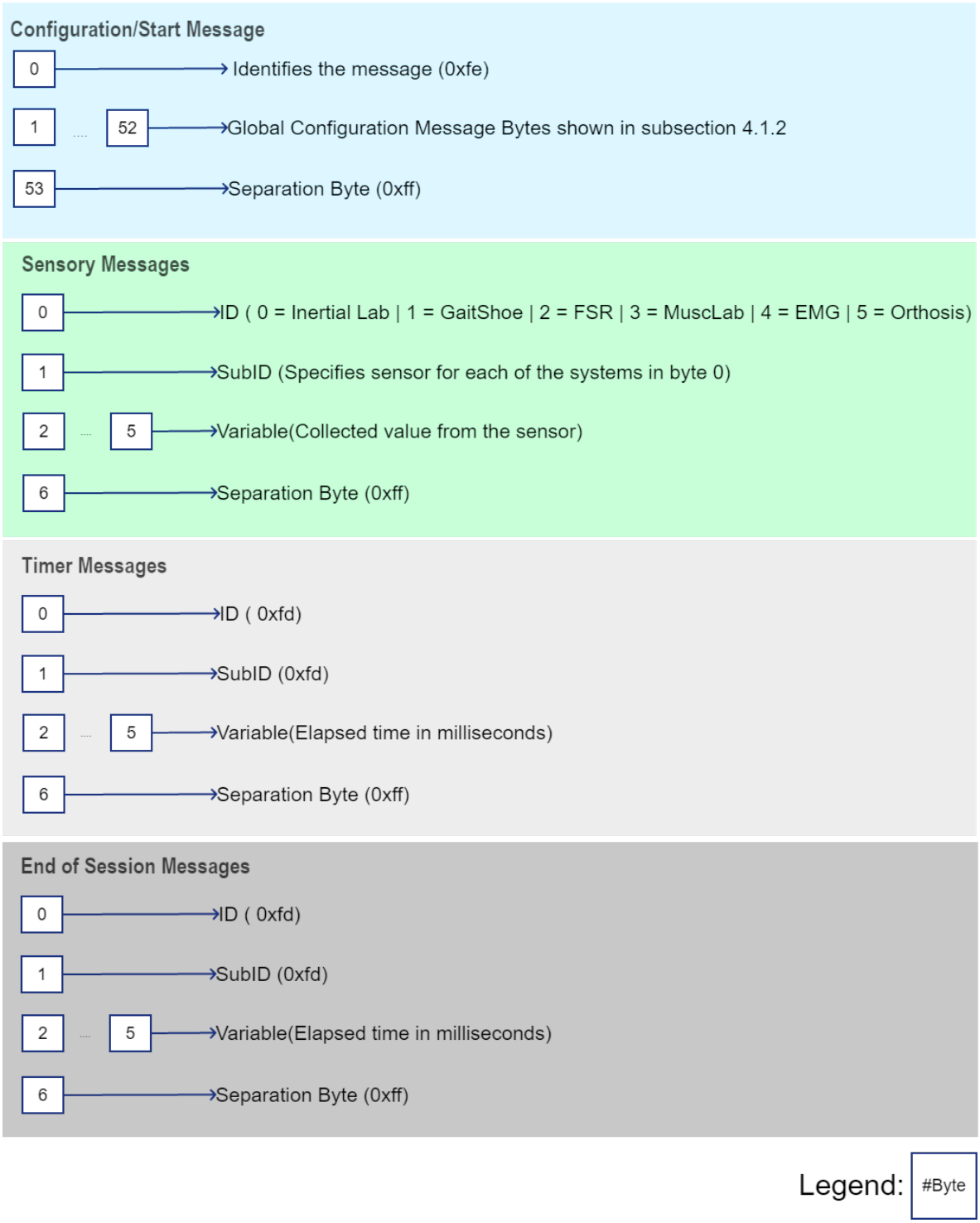


Figure 32.: Labeled S.M.I desktop messages

As mentioned earlier in this chapter, the S.M.I should have the ability to import recorded sessions, both as a means to allow for non-real time observation of session data that might

have been recorded by a session ran solely with the S.C.I as well as the management of session data.

This necessity requires the marshaling of data from a session to a file, as each message as high degree of identity and a set of messages is associated to a point in time from the starting of a therapy the **JSON format was chosen for this specific task**.

Figure 33 shows the representation of a session in a JSON file. As illustrated each field of the configuration + start message is parsed and saved, this will allow us to prepare the session for review when importing the file.

The configuration options are then followed by a field string named **"sessionValues"**, this field holds all the messages sent throughout the therapy session, being represented by a map of lists in which a tick(value that indicates the elapsed time since the beginning of a therapy) **is mapped to a list of sensory messages**.


```
{
   "currentUser": "xxxxxxxx",
   "emg": "bool",
   "emgLeftShank": "bool",
   "emgLeftThigh": "bool",
   "emgRightShank": "bool",
   "emgRightThigh": "bool",
   "fsr": "bool",
   "fsrLeft": "bool",
   "fsrRight": "bool",
   "gaitShoe": "bool",
   "gaitShoeLeft": "bool",
   "gaitShoeRight": "bool",
   "inertLab": "bool",
   "inertlabLeftFoot": "bool",
   "inertlabLeftShank": "bool",
   "inertlabLeftThigh": "bool",
   "inertlabRightFoot": "bool",
   "inertlabRightShank": "bool",
   "inertlabRightThigh": "bool",
   "inertlabTrunk": "bool",
   "muscLab": "bool",
   "orth": "bool",
   "selectedOrth": "#orthosis",
   "therapy": "#therapy",
   "sessionValues": {
      "#elapsedtime": [
         {
            "id": "#Message Id",
            "message": "#Message Value",
            "subID": "#Message SubId",
            "type": "Type of message (Float, Integer, ByteArray)"
         },
         {
            "id": "#Message Id",
            "message": "#Message Value",
            "subID": "#Message SubId",
            "type": "Type of message (Float, Integer, ByteArray)"
         }]
   }
```


Figure 33.: Abstract representation of JSON file content

5.2 DESIGN AND IMPLEMENTATION

This section will cover the developmental process and finalized implementation of the desktop graphical application S.M.I. The intent of the subsections that follow is to address the steps involved in the creation of an application. The application should enable to transform collected data from the various treatments and monitoring therapies carried out by the SmartOs into a form that is intelligible and advantageous for the clinician. Also allowing for the establishment of basic session management. Moreover navigation of the application should not be cumbersome relying on a minimal design and little manual input from the users.

5.2.1 *Early considerations and technology choices*

Homologous to the process presented in the previous chapter, more specifically subsection 4.2.1 some choices regarding the development of the S.M.I **had already been taken and conceptualized**, for instance, early mockups were already developed in previous work as shown in Figure 34

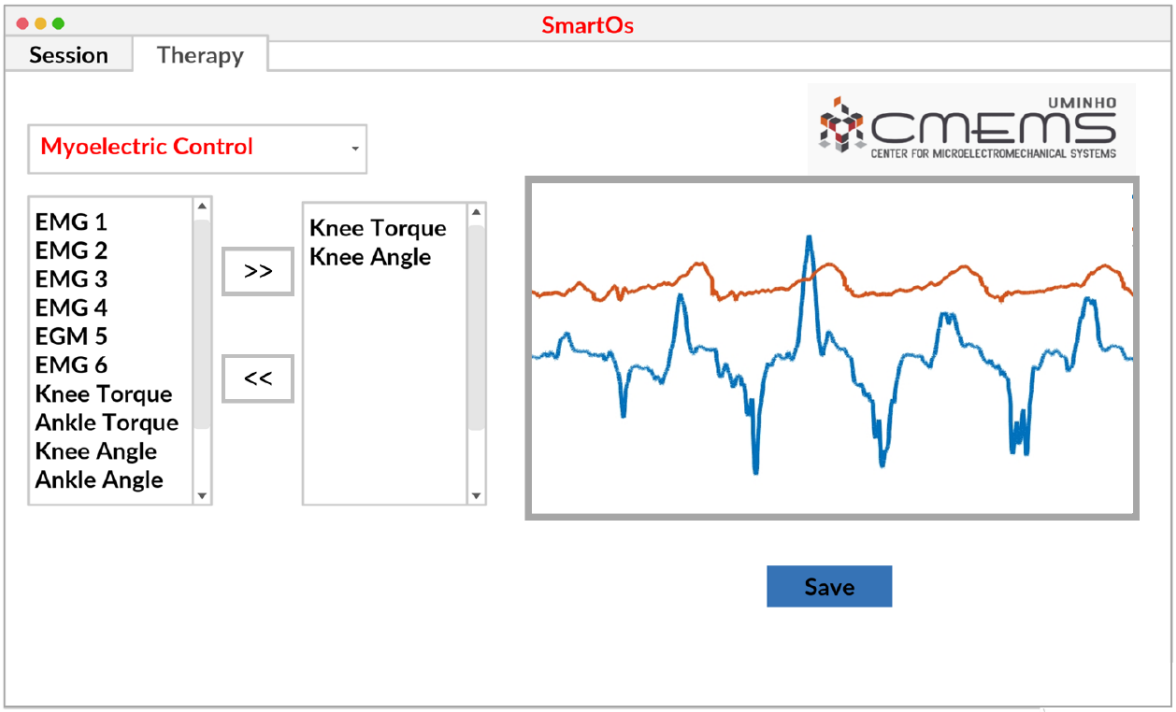


Figure 34.: Early desktop application concepts seen in "A real-time architecture for smart wearable orthoses system" [28]

Figure 34 displays a general direction to be taken in regards to the development of an application for the monitoring of sensory information output from the C.C.U, the preceding image displays a light and easy to use environment with little navigation diminishing usability confusion that might arise upon contact with a new application.

On the other hand using a manual selection menu for the visualization of real-time graphics is shown, the reliance on a component with a **high degree of user dependence** can be negative for the usability and interactivity of the S.M.I. As such, straying from the early concept of sensory data visualization could prove useful in terms of achieving a **more friendly experience** which is still able to support monitoring need to the fullest extent.

No platform target can be extrapolated from Figure 34, more over upon discussing the matter with other collaborators of the project, it is possible that the S.M.I will run on a Windows PC. Multiple options were considered when selecting the development environment for the application, however, since no target OS was specified and taking into account that running the application natively could be beneficial, as to create the lightest application possible, the **Qt platform** was eventually the used for the project.

Qt possesses a **vast number of libraries** allowing for the creation of high-performance UIs, moreover the platform allows for **cross compilation**, meaning that the application can be theoretically ported to any of the main operating systems with diminished effort. Finally the Qt platform is proven to be a viable option in multiple fields of work including the **medical field** [31].

5.2.2 *Application principles*

In regards to the establishment of principles of development for the S.M.I all values that were taken into consideration when developing the mobile application, in subsection 4.2.2, **remain suitable** in the current context.

The application's graphical design should take into consideration the different capabilities of the multiple users of the application. However, in the current scenario, the S.M.I should **not follow the principles of guided actions** that were presented in the mobile counterpart. In regards to the current application there are much less configurations, as such its complexity in regards to the establishment of UI dependencies and conditions is much lower.

This application also presents much higher value to non-technical users, the clinicians, being that technical users would prefer the usage of the marshaled therapies. The lower interaction complexity enables the creation of a system that is much more **automatized in nature and simple in navigation**. As such, it should be able to present a therapy to the user in a layout that fits the demands of existing context, **automatically displaying**

all the needed information that makes up the treatment in session by understanding the components that make up the currently running session.

The final factor to have into account would again be its **maintainability by other members of the project**, this would require the creation of a well segmented application, that provides tools for the multiple functionalities that the system might implement, so that the addition or removal of the various aspects to be contemplated in the current application might be easily executed.

Ultimately the requirements for the S.M.I are:

- **Minimal Input -** The application should not require much input from the user, all functionalities of the application should be **readily accessible and visible to the user**. Additionally, any action that could be effectuated inside the application should require minimum effort on the user's part.
- **Automatic Generation of Visualization Layouts -** Monitoring is first and foremost the main objective of this application as such it should not be hindered by **over-configuration or hidden functionality**.

  Visualization of chosen **sensory inputs should be automatically provided** by the application not requiring the user to navigate through graphical components for unset configurations.
- **Temporal Requirements -** The application should be able to **display sensory input in real time**, but also provide the means with which to review them at a later time.
- **Basic Session Management -** Managing basic aspects of the sessions and users should be provided as to facilitate the process of reviewing a recorded session.

  The management of sessions should require the **establishment of a local database**.

5.2.3 *Basic Application Structure and Mockups*

Having settled basic guidelines for the development of the application it is important to detail the main tasks to be achieved by the S.M.I as well as the process by which these tasks should be achieved. More specifically, it could be relevant to define the multiple components that will compose the application.

Contrary to the mobile application, the S.M.I should not encompass inter-dependent menus. In fact the desktop application should be comprised of the following components:

- **Patient and Session Management** - The **main menu** of the system should provide the user with the ability to search for patients and their consequent therapy sessions.

The application should not only be able to allow the **visualization of real time session but also help the clinician organize and review these sessions**.

This menu should also be **updated whenever a live therapy session ends** and update the Review Session component should a new session to a patient be selected.

- **Live Session Visualization and Session Recording** - This component should be responsible for both the reception and handling of all real-time events. Whenever data is being sent from the C.C.U to the desktop application this component should be able to **capture all incoming messages and automatically produce a customized menu layout for the visualization of all the activated sensors** of the SmartOs system.

- **Import Session** - The session importing component should allow the user to import one of the marshaled sessions that might have been recorded without the usage of the S.M.I, in a standalone therapy.

- **Review Session** - Sessions selected in the Patient and Session Management menu should be visualized in another as a dashboard, allowing the user to analyses data from previous experiences. This should in every way be similar to the menu presented in the Live Session with the exception of certain graphical event components that will be explained further ahead in this chapter.

Using the description of the above mentioned components as guidance rough mockups were elaborated for menus in the system with highest graphical complexity, in this case the Review and Live Session as well as the Patient and Session Management menus were drafted resulting the 2 windows presented in Figure 35:

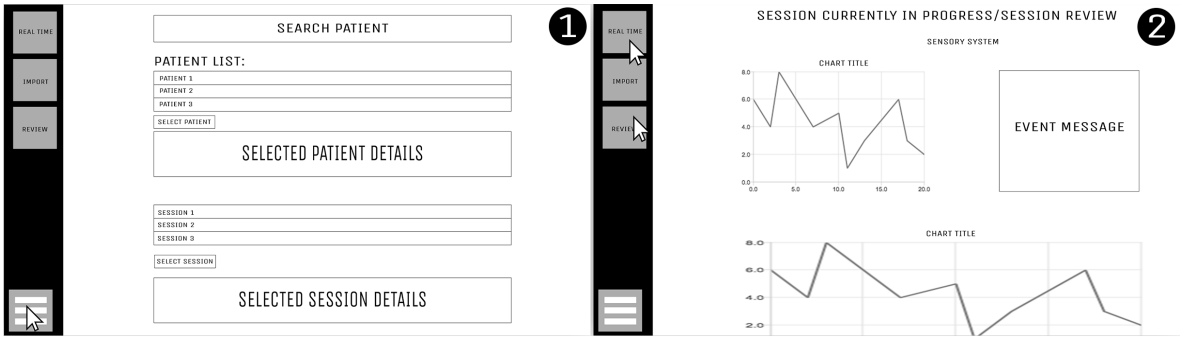


Figure 35.: S.M.I. mockups

Figure 35 represents **The Patient and Session Management** and **Live Session Visualization** respectively. The graphical interfaces to be developed on the desktop application should, as previously mentioned, be navigated with **minimal user interface**.

Moreover, the main objective of this application is the observation of a considerable amount of data. As such, a graphical design similar to that of a **dashboard was implemented**.

Navigation should never take more than a couple of clicks. To the left side of the windows in Figure 35 a side bar with multiple menus **should allow navigation between the main components** to be implemented on the application. The implementation of this style of navigation does not require the utilization of complex nested menus, ensuring that the user can **instantly navigate** between the multiple components with a single click.

Simplicity regarding the graphical design language of the application is kept along the multiple interfaces of the system. The first window represents the first menu to appear upon the start of the application, it allows the user to search for a patient, consequently presenting it with all the matches for any given search in the form of a list. This should in turn prompt the appearance of the patient details as well as a listing with all the sessions belonging to said patient. Once a session is selected the system should also **load the session on the review** menu where the user can see the data received from a completed therapy.

The second window illustrates the representation of the multiple graphical interfacing components that will be explained further ahead, each of the graphical representation widgets should be individually titled for ease of comprehension, additionally a block of graphical elements **should also be titled with the name of the sensory subsystem being analyzed**.

It should also be noted that at the end of therapy the session should be recorded to a patient, may this patient not exist in the database the patient should now be added, hence the **nonexistence of any buttons for adding a new patient in the main menu of the application**. The intention of this behavior is to once again diminish the necessity for manual control of the end-user.

The process of recording sessions would mean that a database would have to be introduced to the system so that users and their sessions can be linked, additionally it will facilitate the lookup of these entities. Sessions should also have a field for storing the full path of their marshalled representation.

5.2.4 *Application Architecture and Navigation*

This section will cover the most relevant classes used in the development of the application, their importance in the materialization of behavior described in the previous subsection such as the development of one the one click navigation system and patient and session management as well as **SIGNALS**, a Qt platform primitive which aids the **messaging between multiple menus and different contexts within the application**.

In previous sections the infrastructure for the development of a robust application was started, however, up until now the S.M.I was defined in relation to its capabilities for navigation and ease of interaction in conceptual terms, many objectives determined and abstract solutions defined. So that the application could evolve from its conceptual stages it was necessary to generate a concrete architecture with which to implement them.

Figure 36 represents the architecture that support the S.M.I.

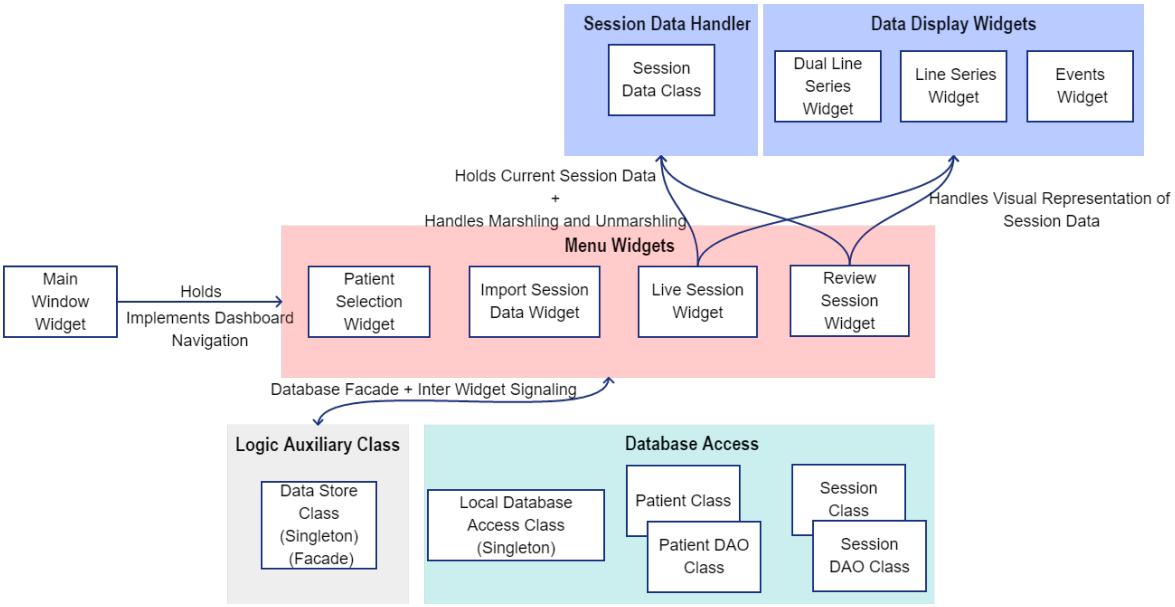


Figure 36.: Desktop Application Architecture

Figure 36 represents the division established when structuring the Qt project. It is composed by five major segments which are then composed by multiple classes. These will help implement all the objectives defined in previous sections whilst helping the application be easily upgradeable due to high modularity of the multiple Qt widgets.

To better understand the Figure 36, interaction between the multiple segments will be the focus of the current subsection, process which can be started with the Navigation Segment. The navigation segment is comprised of the **Main Window Widget** and the **Menu Widgets**, the main window class holds all four essential menus of the system introduced in the previous subsection. Moreover, the Main Window widget implements a **sidebar with four clickable buttons**, clicking these will result in the **activation of the four menu widgets held by the Main Window Widget Class** effectively rendering a layout written in an XML document, all the meanwhile the previous shown menu is hidden.

Each of Menu Widgets Classes instantiate the **Data Store Class shown in the Global Auxiliary Class Segment** holding it as a private variable, this will facilitate immensely the communication between the multiple menus that make up the application as Qt uses its own primitive, SIGNALS, for communication between the multiple widgets of the system.

In essence signals are an alternative to the usage of function pointers and callbacks, allowing for one class to **emit** a **signal**, and caught by another class implementing a **slot** which **captures said emission**. This slot is represented in code as a regular function accepting as a parameter **whichever object is sent by the emitte**r, this parameter can then also be processed as in a regular function.

This practice while very helpful still needs the usage of a ”parent” widget, this ”parent” can either connect itself to one of its Widgets or two of its Widgets. As such the Data Store Class comes as an **intermediary**, being implemented as a QObject (the basic object from the Qt Platform). This class can be registered as a slot or a signal emitter and **connected to multiple objects** meaning that if one menu needs to interact with one another a signal sent by one of the **Menu Widgets** can be connected to a slot from the **Data Store Class**, this slot function can receive this parameter and **re-emit it to a slot on whichever other Menu Widget**.

Additionally, this class acts as a **Database Facade** implementing all the needed requests to perform the UI logic in all Widgets, this should reduce code complexity as functionalities that would need to modify or access both patient and session tables can now be represented in a single line of code.

Whilst the Data Store Class stands as a facade for the database access it should not venture over into the SQL realm, the SQL Layer should be completely abstracted through the usage of the Database Access Segment. Being that this system will be operated on a machine to machine basis and the database would for now be comprised of only two tables, it is not expected that the system would need a database with a high performance.

As such, and to facilitate the development off an application that should ultimately integrate a portable and offline system a **SQlite** database was chosen to handle the storage of all data. SQLite is **an embedded, self-contained database engine** which is fully supported by the Qt platform, its integration with the application would therefore be practical and trivial and without shortcomings in the current context.

The database would be comprised of two tables, the Patient Table and Session Table holding all the necessary details for the management of sessions. Using the identification inputed in the mobile application as the identifier for the patients to which correspond multiple sessions. Each of these sessions holding the path for a marshaled therapy file as shown in figure 37.

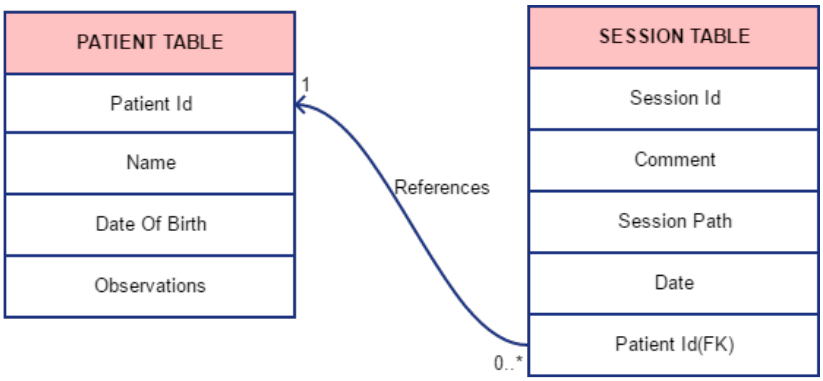


Figure 37.: Database Tables

The SQL Code responsible for addition to each of these tables is implemented in the Patient and Session DAO Classes, **these are data access objects** which permit a complete abstraction of the usage of SQL code in the remaining classes establishing all the needed functions to modify Patient and Session values whilst returning objects of the Patient and Session Classes, which are **Object representations** of the multiple entries found in the Patient and Session database tables.

Lastly, Data Representation and Storage Segment of the application. This segment is pertains to all the widgets that in any capacity handle the sensory data collected from the SmartOs system, being divided into two sub-segments. The Session Data Handler is responsible for holding all of the sensory data that makes up a session in its object form. Moreover, it provides methods for both the **importation** of JSON data files into its own variable as well as the **marshaling** of the data after a session has been ended and added to the system.

In addition to this class multiple widgets had to be created to enable the elaboration of customized line charts with **one or two vertical axis**, **Line Series and Dual Line Series** Classes respectively, in a matter that is reproducible for the multiple sensory inputs without resorting to the repetition of QtChart libraries code. Additionally, some sensory information takes the form of a set of constants, each with a specific textual representation and only relevant to a specific moment in time, this information should be represented by its textual correspondence implemented in the Events Widget.

5.3 Final Applications and Critical Analysis

Similarly to the closing section of the previous chapter, the current section will link the preparation described in previous sections to the final application that would be named S.C.I. Using images from the final application elations should be taken and comprehen-

sive detailing of the implemented methods of interfacing and consequent comparison with established goals for the project made.

Contrary to the S.C.I, the S.M.I should not impose, as explained in earlier chapters, a fixed navigation pattern, as such the steps for interaction with the multiple components are not particularly guided, nonetheless these should be easy enough for anyone to use. The **first step** when using the S.M.I should **always be the connection to the WiFi** network generated by the C.C.U, this step can be seen in Figure 38. The process by which the user connects to the newly generated network is similar to that of the connection to any other network, ultimately generating the conditions for establishing the communication channel described in previous sections.

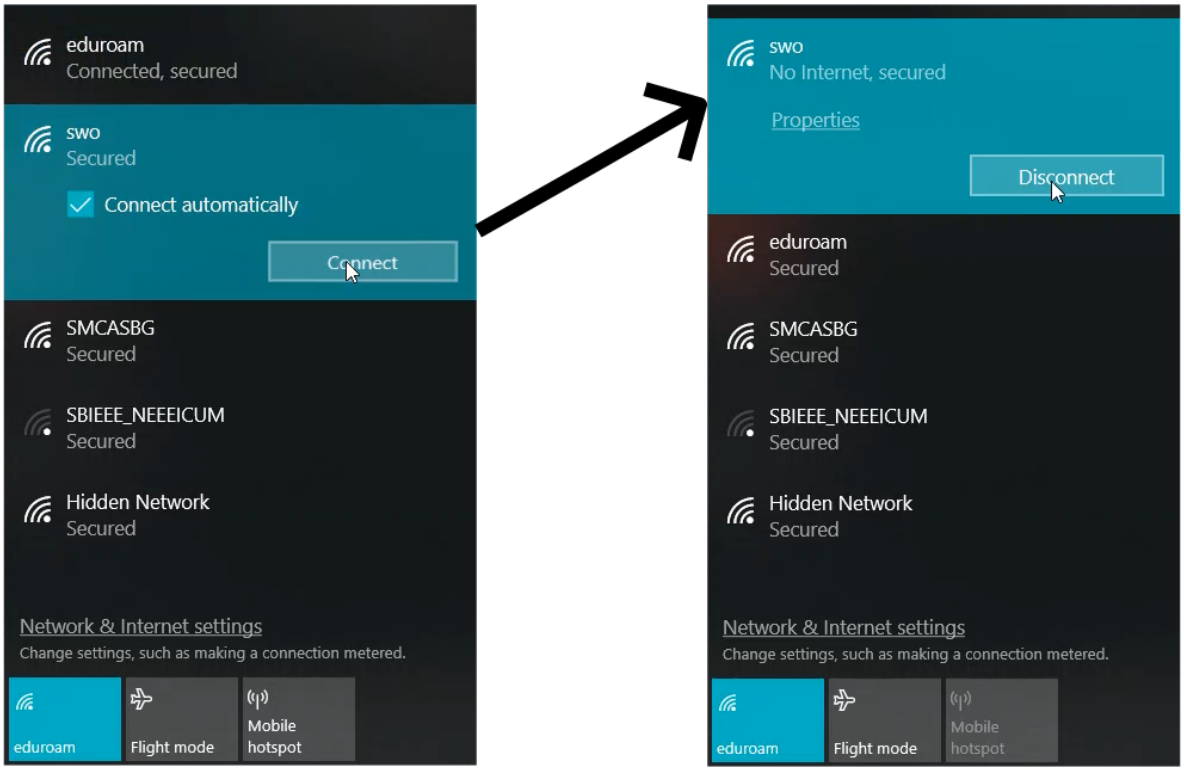


Figure 38.: Establishing connection to the C.C.U

Upon starting the application, a connection to the C.C.U will be established automatically, as such the user should now be able to interact fully with the various menus and options offered by the application. The first menu presented to the user upon starting the application would be the **Patient and Session Management Menu** illustrated in Figure 39. From this menu the clinician can obtain essential information regarding a patient, by either **selecting a patient** from the patient list indicated with the number 2, to ease the search process the number of entries can be **filtered with the aid of the search box** identified with 1.

Once the patient is found he should be selected with a click on the button presented bellow the list, this should in turn **reveal all the recorded patient information** comprised by its identification, name, date of birth and general observations taken by the clinician whenever it was first added to the system. The general observations field should, in particular, facilitate the potential review of a patient's condition by multiple clinicians, and consequent visualization of specific therapy sessions.

In addition to the user details, upon patient selection a list of therapy sessions is filled in the component identified by the number 4, once again using the same process of selection a therapy session can be selected thus displaying its details such as its date and observations where **updates regarding the evolution of the patient** as well as session parameters should be given.

This selection also prepares the review menu for the loading of the selected session. Once selected the user can navigate the multiple menus of the system using the sidebar on the left and view all the sensory information regarding the recorded session.

Up until this point all these actions were realized without leaving the main menu, using **minimal amount of user input**, in fact as it is observable no buttons for user or session addition are present on the Patient and Session Management widget.

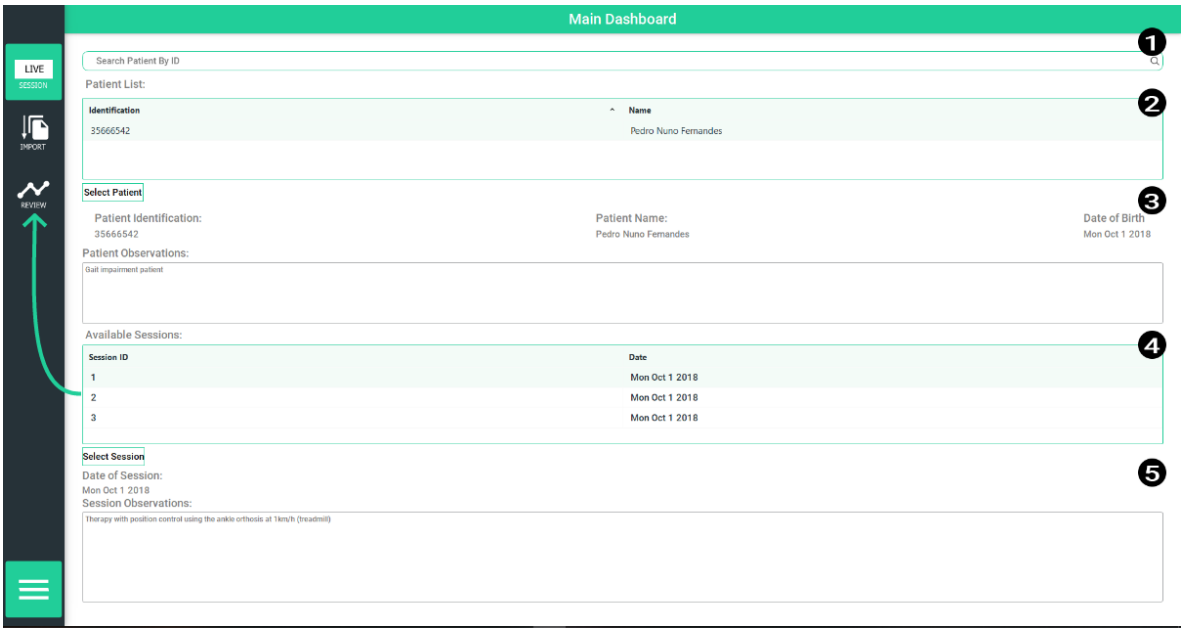


Figure 39.: Patient and Session Management

The process by which an **user or session is added** is represented in Figure 40, in the current scenario the process is being applied in the **context of a session importation**. On top of the menu a button imprinted with a folder is used to bring up a system file explorer, which will allow navigation to the folder of the recorded session. Upon selecting the session

the menu shown bellow the button will appear, the same menu will also appear once a real-time therapy stops.

User identification is automatically filled in from either the marshaled session or the configuration message, as such the user does not have to manually research for a patient. If the patient is present in the database all other fields related to the patient will be **filled and unalterable**, otherwise these will be blanked out and editable, if the user identification is incorrectly inserted in the mobile application **it can still be altered** in the current menu.

A session description can now be inputed in the Session Data section and upon clicking the add button the Session will be added to the displayed patient or in the case of a new patient both the patient and session will be created. The usage of a single menu for both session and patient addition in multiple contexts of S.M.I should grant the user familiarity with the system by **diminishing the amount of interfaces that compose it**.

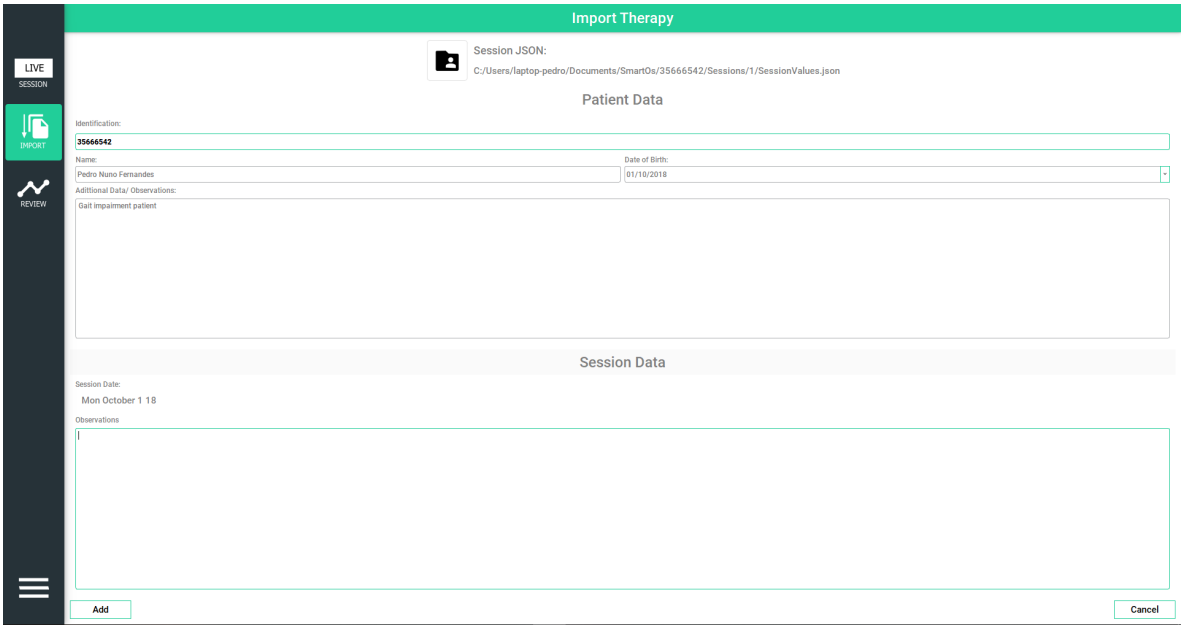


Figure 40.: Importing a previously recorded session

By now, most of the menus that compose the system have been presented, to the exception of the data representation menus, the **core** of the S.M.I. The menus shown on Figure 41 and Figure 42 represent the real-time observation of a therapy and review of a therapy respectively.

These will be explained and compared as a whole, due to their similar nature, both in terms of **implementation and functionality**. Both these menus are represented through **dynamic layouts** rendered with the invocation of the Qt primitives in their respective classes, rather than through the usage of **conventional static XML files**. As previously explained, initial concepts of the application required a great amount of user input in regards to the

observation of collected data. Not only would this require the user to know what sensors were being polled but also would only allow to visualize a single sensory system at a time.

In the current scenario, and as planned earlier in subsection 5.2.3, these layouts should be tailored to the user needs granting the best possible experience. As such both these menus display all the collected sensory information through **multiple graphical and event widgets** (numbered 1, 2 and 3) in a scrollable pane, ensuring that the user can rapidly scroll through the **inputs from the multiple active subsystems of the SmartOs**.

Additionally, the Widgets used for displaying data shown in Figures 41 and 42 were introduced earlier in subsection 5.2.4. The widget labeled with the number 2 is the Event widget, as such it should only appear in the real-time session menu, these widgets are used for specific actions in time representing a numeric value or its textual representation that is only useful for a clinician in the context of the current movement. In this particular example it uses the number "155" to specify current stiffness of the orthosis. However it could also represent a movement pattern, for instance one other event present in the system would be the FSR event, the text for this event could be "Heel Off", representing the removal of the heel from the floor, which would be a more typical usage of this type of widgets.

Lastly, two other widgets were developed both for the real-time data collection and session review, these are represented by **Line Series Plots** which were developed specifically for this intent. The simpler of which being the LineSeries Widget shown in 3 it represents a plot with a single axis and **no optional toggles**, on the other hand the plot shown in 1 represents a plot with **two vertical axis and a toggle for hiding and showing another series to the graphic**. This addition came from the needs of **other collaborators** of the project when testing the system. The complexity of the latter is therefore much higher.

Additionally logic had to be developed both for **navigation and zooming** on both graphical representation showed on the upper left corner of each of the plots as it is not standard in the QtCharts library, moreover the establishment of real-time graphics logic also had to be developed as the **Qt API is focused on the creation of static charts**. This logic allows the plot to follow the currently received messages readjusting both vertical and horizontal axes. To allow for more versatility, the user can navigate backwards even on a currently running therapy whilst always being able to go to real-time by pressing the last of the previously mentioned navigation buttons(labeled "LIVE").

The process of establishing real-time graphics has however some shortcomings, as the API's provided by Qt for rendering graphics do not perform well in terms of rendering, being that the polling rate for each of the sensory subsystems is 100Hz a process of decimation had to be partaken, **thus reducing the signal quality in real-time data display**. Data is, however, still recorded at the original sampling rate. Additionally pre-loading a QChart with all session data **has lower performance impact**. As such it is possible to see the difference in data sampling from the real-time graphics illustrated in Figure 41, shown

at a frequency of 10Hz after decimation process, and the plots drawn in Figure 42 which manage to display data in its full sample rate.

Still on the topic of real-time session monitoring, both connection and message reception is handled by a Qt class, QTcpSocket, that provides an extensive API for handling network connectivity, furthermore it offers **asynchronous non-blocking IO functionality** by using QThreads and the Signaling primitive, this prevents the existence of UI for connecting to the server by polling the server IP address for a connection, additionally allows data to be received on the background and consequently parsed and displayed on the various Data Representation Widgets, offering yet another layer of abstraction to the system in regards to the end-user.

Figure 41.: Real Time Therapy Observation

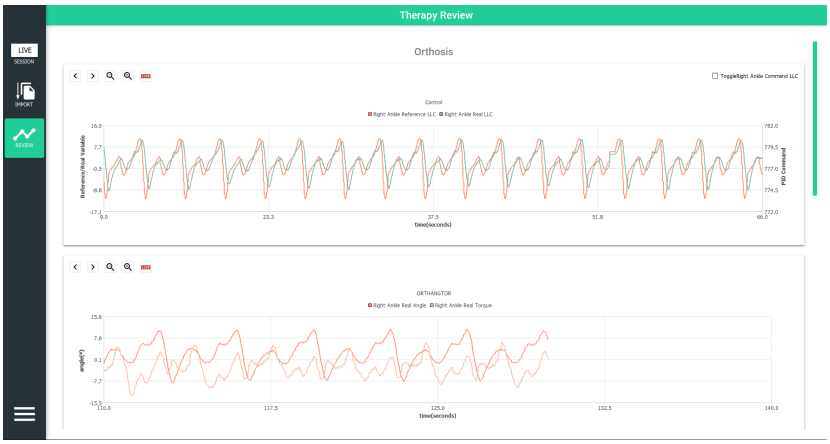


Figure 42.: Review Therapy Session

The process of real time therapy establishment is also further detailed in appendix A, linking a recording of this process and corresponding visual aid detailing the communication procedure between S.C.I, C.C.U and S.M.I over time.

Ultimately, an application that fulfills all the envisioned requirements was completed throughout the developmental process of this dissertation, comprised by a minimal number of menus that use textual and visual aids so that the **adaption period** to the usage of a new tool such as the S.M.I is **minimized**.

This design minimality **was not**, however, **achieved through the diminishing of functionality** but rather through the usage of modern UI design patterns such as the dashboard representation of visual graphics.

While the application is comprised by a modular code structure to maximize maintainability and updatability, menus that display widgets dynamically still possess a high degree of complexity, such is the case for the **Review and Recording Session Widget**, both in terms of layout definition and inter-widget signaling introduced in earlier sections of this chapter.

5.3.1 *Application Validation and Testing*

Much like the validation provided for the mobile counterpart, this subsection aims to validate application **functionality and usability** in regards to implementation and user experience.

The present application does not have the need to enforce dependencies, however, it is focused on the automation of functionality and therefore minimization of user input. The process of interaction is therefore illustrated in Figure 43.

Figure 43 displays all major interaction with the system in regards to the functionality of the application from its initialization. As can be seen only four of the systems actions require the input from the user.

The transition between different states is represented with arrows, additionally arrows labelled with the numbers 1 to 5 represent different phases or patterns of the interaction.

Arrow 1 represents the asynchronous connection to the C.C.U followed by the **automatic configuration** of the live therapy layout once monitoring messages start reaching the S.M.I.

Arrow 2 on the other hand represents the manual selection of a marshaled session by an user.

Both the real-time session and session importing culminate in the presentation of an interface created for the session addition or if need be user addition as well, shown in arrows 5. The usage of the same interface in two different tasks with the same final objective of adding a session to a user should diminish the presence of unfamiliar or different graphical components to the user. Hence decreasing the learning curve faced when interacting with new software.

Arrow 3 includes user input from the main menu of the system. It is at this instant that the user can select an user and session and see its details. Culminating in the automatic loading of a therapy for review presented in arrow 4.

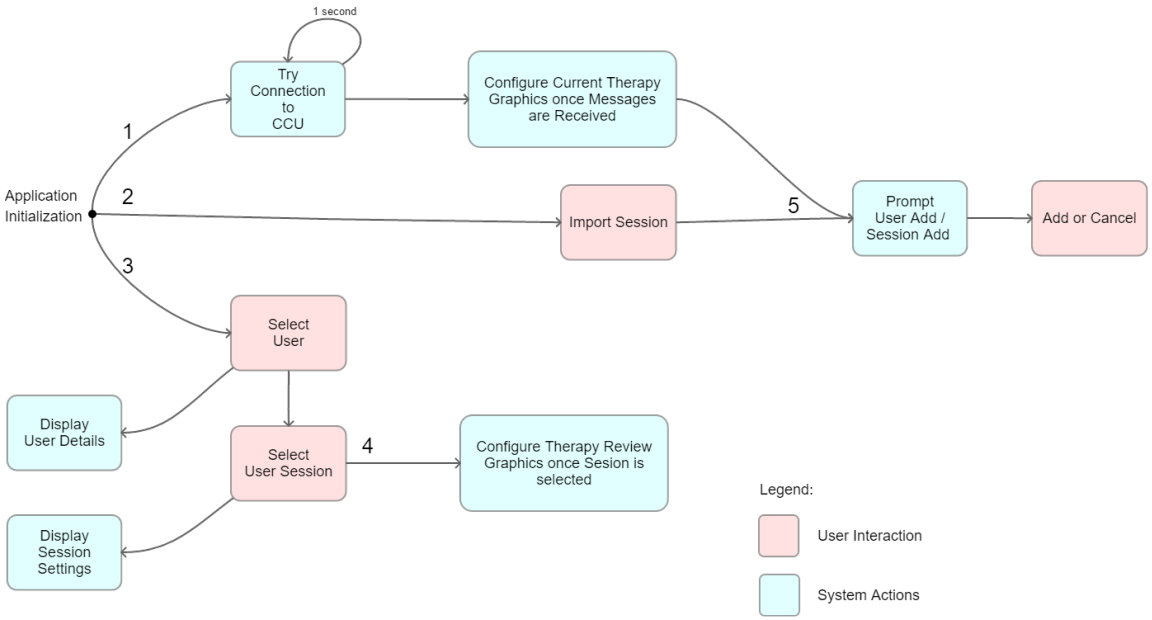


Figure 43.: S.M.I. interaction Flowchart

Not much data is created by the present application, however live data incoming from the C.C.U is recorded by the S.M.I. This **data was validated by using the data recorded locally in the C.C.U and corresponding file in the S.M.I** after recording a live therapy. Ensuring that the system is able to handle the data throughput without missing messages and ensuring that the data recorded in the S.M.I matches the data recorded locally on the C.C.U.

Validation in regards to the usability of the S.M.I **followed the same process as the one described in the S.C.I** in subsection 4.3.1, being introduced to the 8 individuals introduced in subsection 4.3.1 with a brief explanation of the intended objective of the application. As the application is dependent on monitoring the users were asked to use the multiple menus and try to understand their usage without external observations.

The SUS was also applied to the current application as it is agnostic of the system it evaluates.

The answers to the SUS questionnaires will also be analyzed further in this subsection (please refer to subsection 4.3.1 for more information on the SUS evaluation).

Table 3 presents the average scores of the SUS questionnaires along with the best and worst ranking items and respective individual scores.

Table 3.: SUS scores for the S.M.I.

	Score
Subject 1	80
Subject 2	85
Subject 3	90
Subject 4	75
Subject 5	82.5
Subject 6	77.5
Subject 7	85
Subject 8	85
Average	82.5

Lowest Scoring Item(s)
Item 1: 2.625

Highest Scoring Item(s)
Item 2: 3.75 ; Item 6: 4.0

In parallel to the evaluation effectuated for the S.C.I the mean score obtained from the 8 individuals who partook the SUS is **82.5**, the **lowest individual score being 75**, which represents an evaluation in the upper limits of the **"GOOD" to "EXCELLENT" range**. This value indicates that the application is effective in respect to the objectives that is set to achieve, being likely that the employed design practices resulted in a facilitated and **straightforward interaction with the system as a whole**.

The normalization of the mean values for each of the items in the questionnaire also provides some important insight into the multiple aspects of the application.

Item **1)** resulted in the lowest score, 2.625 out of 4, enhancing the hypothesis that item **1)** is not well adapted to the test subjects used for the current application as the **distance to the field of utilization potentially isolates the subjects from the importance of the application**.

No other item scored under 3 out of 4. Moreover, the highest scoring items once again point to the simplicity and instinctual usage of the system.Item **6)** received a perfect score, 4 out of 4, as none of the participants found any inconsistency with the system.

Items **2) and 8)** both scored over 3.5 out of 4, these items are directly related to the complexity of the application and limiting factors, **suggesting that the principles of minimization of input and automation of the monitoring layouts defined earlier were successful in abstracting the user from the inherent complexity** of the system.

All users were able to navigate the menus to the **fullest extent and understand the functionalities and interaction between them**, choosing a patient and respective session and proceeding to the review menu. Users also imported therapy session files from a thumb drive to a chosen patient and waited for a real time therapy which was simulated to facilitate the testing process.

Again users were asked about any possible changes to be committed to the application. Contrary to the S.C.I no modification was pointed out, possibly to the simple nature of the application both in terms of menu quantity and intended functionality as graphically

monitoring data should be more familiar to the average user than the control of a foreign, multi-modular system as is the case with the S.C.I.

One **gripe** users pointed out was the small delay that occurs when using the review menu, this is once again product of the **poor optimization of the QtChart library** and could not be, as such, avoided.

The application was also tested under several PCs. Whilst Qt permits development of cross-platform software the S.M.I was developed and compiled for running on Windows OS.

The minimum tested settings for the present application were a dual core Intel Core i5-3337U with the integrated graphics card and a 1280*728 pixels resolution running Windows 7. The multiple functionalities of the application are serviceable, however, taking into account the low performance of the QChart library the S.M.I performance is improved when using the application in PCs with greater computational power, especially in regards to the real time rendering of charts.

6

CONCLUSIONS

This chapter presents the conclusions to be taken in regards to the work that has been developed, and introduce some potential areas for future improvement that were evidenced throughout its development.

Introduced in the earlier chapters of this document the objective of this dissertation was the creation of an intuitive set of tools with which to interact with the existing SmartOs system, these tools would target both mobile and desktop platforms and would culminate in the development of S.C.I and S.M.I, respectively.

The S.C.I would have the ability control the aforementioned system, whilst the S.M.I would **allow the motorization** of the variables collected by the C.C.U of the system throughout, whilst providing some management capabilities through the introduction of an embedded database.

Both applications possess at core the same pillars, as such they were created to be as simple as possible and unobtrusive to the functioning of the system as a whole.

Being more specific, the development of S.C.I provided a way to visually command the system, this would culminate in a tool that would **facilitate** the testing of the system and aid demonstrate its potential.

Using a linear navigation in which subsequent menus are influenced by the choices realized in previous menus and visually outlining mandatory fields throughout the configuration of a therapy allows the user to navigate to following menus with ease.

Moreover, the navigation's linearity also permits to impose restrictions in regards to the users' choices, as such an user can only navigate into further menus of the application if the current conditions meet the mandatory fields. This ensures that the end configuration message is always valid guaranteeing the safety of both the system and patient.

Each **configurable field was discussed with remaining members of the SmartOs project** so that the options for each of the systems are well categorized and identified. This would be very important, indicating the necessity for creating more complex custom layouts for the activation of sensors and setting N-ary options or absolute values as is the case for speed and stiffness.

Each Activity relates to a different set of configurations, to ensure that each Activity does not show to much information some activities can encompass multiple Fragments within a same context, allowing for a fluid sliding navigation between the multiple Fragments.

The user is completely separated from the implementation of the multiple modules, each of the possible configurations uses custom made layouts intuitive to the choice of each option.

The S.M.I presented fewer interfacing restrictions, being implemented with the main goal of aiding real time observation of therapies, it evolved due to necessity into a system capable of managing these sessions in an organized manner as well as a tool for its revision.

The desktop application is composed of four major menus, ensuring that adaptation to the system is rapid and effective as no other hidden menus are present.

Navigation within the S.M.I is realized with minimal amount of user input and menus are tailored to each specific task of the menu. This is especially important for the monitoring menus of the system as it allows the user to focus on the task at hands and does not require any additional manual configuration in regards to the presented graphical components.

The S.M.I **presented some shortcomings** in regards to the performance of the selected development framework, more concretely the chart libraries provided by Qt, these would however only affect the real-time plotting slightly, resulting in a reduced data display rate when compared to the polling rate of the systems. This delay was also noticed by users testing the application, being a potential aspect for improvement.

Using an embedded local database, it was possible to establish a basis system for managing sessions. Logging patients basic information and therapy sessions with brief descriptions and marshaled data, ensuring that all aspects of reviewing a treatment can be realized from within the application without resorting to external tools.

The validation of the applications through the involvement of peers was valuable in the effort of **establishing robust applications that behaved in the intended way**, guaranteeing that security requirements are met.

The testing of the applications by non technical individuals was instrumental for the improvement of overlooked details and additionally guaranteeing that the work developed in this dissertation is aligned with the needs with the final user and a positive addition to the system as a whole.

Moreover, the positive results obtained from the SUS questionnaires indicate that both **applications implement an effective abstraction of the low level primitives of the SmartOs system,** as both were evaluated in an acceptable range of usability, more specifically the ”GOOD” to ”EXCELENT” range.

As such the developed applications show **great potentiality** in terms of expediting the process of configuration and monitoring of the multiple modules of the system through the development of simple yet complete interfaces that allow to complete therapy sessions with

minimal input whilst ensuring safety and organic usage of the system, demonstrating their value, especially in a real world scenario.

Additionally the remaining members of the project are now equipped tooling to better test the multiple modules of the SmartOs, both in terms of activation and monitoring. This would be important as **not all members would have the familiarity** with the C.C.U of the system, facilitating the integration of new modules, or alterations and further development of previously existing modules of the SmartOs.

Also regarding the constant development of the project, the work realized throughout this dissertation was built with the intent of having a sensible and modular structure to each of the applications with an easily alterable and intuitive code-base. Whilst efforts towards this end have been made as explained throughout the the developmental chapters of both the applications, **this would only be tested when members** of the project need to update application parameters based on changing dependencies or additions of the SmartOs.

Ultimately, having in mind the developed work and the feedback of multiple peers and users the tools developed throughout this dissertation have, by abstracting the multiple components of the SmartOs into a visual package, the potential of making what would be an **unusable system due to its a complexity** and technicality **into a potentially beneficial system for gait analysis and treatment within a clinical field**. Providing a **small insight and testing grounds for a system that could eventually be a commercial tool for human rehabilitation**.

6.1 Future Work

In spite of having achieved the main goals for the dissertation, some aspects of its development possibly warrant the definition of additions to the currently existing system.

A single device approach to the system could be beneficial to the SmartOs as a whole. For instance, the usage of a dedicated tablet could decouple the need of using a personal desktop and allow all interaction to be made on a **touch-screen potentiating natural usability and motion with the monitoring elements**, especially in regards to the navigation of the multiple charts that make part of a therapy session.

The introduction of the SmartOs to the world of Internet of Things (IoT) would be crucial in heightening its applicability in a real world scenario, namely through **the development of a web platform for online session storage and access**.

Web based applications are employed by some of the commercial fitness and clinical solutions and gaining popularity in academic research papers as introduced in chapter 2, the establishment of a Web based solution could ultimately facilitate the **usage of collected data in any device** not requiring the installation of a dedicated application to do so.

Moreover it could expedite the process by which data from therapy sessions is shared, potentiating the categorization and optimization of treatment through the creation and analysis of bigger data sets.

Additionally, the introduction of a C.C.U with **higher computational power could be paired with the web capabilities and proprietary communication protocol to create what would eventually be the best solution for the system as a whole**. For instance, introducing a stronger C.C.U should allow to port the logic from the current S.M.I and S.C.I to a local web application solution in the C.C.U of the SmartOs, such a solution could permit both control and monitoring of the SmartOs over a Web Browser from a device on a same local network, targeting the greatest number of platforms and entirely decoupling the SmartOs from external dependencies.

The introduction of an Long-Term Evolution (LTE) capable C.C.U could allow to store recorded sessions directly in the cloud based solution for data storage mentioned in earlier paragraphs of this section.

REFERENCES

[1] System usability scale (sus). URL https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Retrieved October 15, 2018.

[2] Fitbit app, . URL https://www.fitbit.com/eu/app. Retrieved January 29, 2018.

[3] Fitbit, . URL https://www.fitbit.com. Retrieved January 29, 2018.

[4] Hexoskin, . URL https://www.hexoskin.com. Retrieved January 30, 2018.

[5] https://www.hexoskin.com/pages/health-research, . URL https://www.hexoskin.com. Retrieved January 30, 2018.

[6] Matlab gui. URL https://www.mathworks.com/discovery/matlab-gui.html. Retrieved January 30, 2018.

[7] Demand for smaller, more portable medical devices is driving innovation in sensor technologies. URL http://www.newelectronics.co.uk/electronics-technology/demand-for-smaller-more-portable-medical-devices-is-driving-innovation-in-sensor-te147227/. Retrieved January 30, 2018.

[8] Vivosense, . URL https://www.vivosense.com. Retrieved January 30, 2018.

[9] Hexoskin and vivosense analysis software for researchers, . URL https://www.hexoskin.com/pages/vivosense-data-analysis-software-for-researchers. Retrieved January 30, 2018.

[10] A. Bangor. Determining what individual sus scores mean : Adding an adjective rating scale. 2009.

[11] BITalino. Bitalino homepage, 2017. URL http://bitalino.com/en/. Retrieved January 11, 2018.

[12] BITalino. Bitalino api, 2017. URL http://bitalino.com/en/development/apis. Retrieved January 11, 2018.

[13] BITalino. Bitalino software, 2017. URL http://bitalino.com/software/. Retrieved January 11, 2018.

[14] N. B. Bolus, C. N. Teague, O. T. Inan, and G. F. Kogler. Instrumented ankle 8211;foot orthosis: Toward a clinical assessment tool for patient-specific optimization of orthotic ankle stiffness. *IEEE/ASME Transactions on Mechatronics*, 22(6):2492–2501, Dec 2017. ISSN 1083-4435. doi: 10.1109/TMECH.2017.2761746.

[15] D. Caselli. rpi3-hotspot. URL https://github.com/damiencaselli/rpi3-hotspot. Retrieved January 29, 2018.

[16] M. F. Domingues, N. Alberto, C. Leitao, C. Tavares, E. R. de Lima, A. Radwan, V. Sucasas, J. Rodriguez, P. Andre, and P. Antunes. Insole optical fiber sensor architecturefor remote gait analysis - an ehealth solution. *IEEE Internet of Things Journal*, PP(99):1–1, 2017. ISSN 2327-4662. doi: 10.1109/JIOT.2017.2723263.

[17] FEETME. Feetme holter. URL http://www.feetme.fr/en/index.php#open-shadowholter. Retrieved December 28, 2017.

[18] FEETME. Feetme sport, 2017. URL https://feetmesport.com/en/index.php. Retrieved December 28, 2017.

[19] FEETME. Feetme podiatrist, 2017. URL http://www.feetme.fr/en/index.php#open-shadowpod. Retrieved December 28, 2017.

[20] M. Holtmann and J. Hedberg. Bluez. URL http://www.bluez.org/. Retrieved January 29, 2018.

[21] A. Huang. An introduction to bluetooth programming. URL https://people.csail.mit.edu/albert/bluez-intro/. Retrieved September 18, 2018.

[22] E. Kantoch. Technical verification of applying wearable physiological sensors in ubiquitous health monitoring. In *Computing in Cardiology 2013*, pages 269–272, Sept 2013.

[23] A. Khalaf and R. Abdoola. Wireless body sensor network and ecg android application for ehealth. In *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, pages 1–4, Oct 2017. doi: 10.1109/ICABME.2017.8167526.

[24] H. Lara-Padilla, X. S. Sanchez, and T. A. Paucar. Design and evaluation of a low-cost mechatronic system to study upper and lower limbs biomechanics. In *2017 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 1–5, Oct 2017. doi: 10.1109/GHTC.2017.8239307.

[25] G. C. Lars Lunenburger and R. Riener. Biofeedback for robotic gait rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, jan 2007.

[26] MyOpenLab. My open lab homepage, 2017. URL http://myopenlab.org/. Retrieved January 1, 2018.

[27] R. Native. React native. URL https://facebook.github.io/react-native/. Retrieved September 25, 2018.

[28] P. J. F. Oliveira. A real-time architecture for smart wearable orthoses system, 2017.

[29] M. Omoogun, V. Ramsurrun, S. Guness, P. Seeam, X. Bellekens, and A. Seeam. Critical patient ehealth monitoring system using wearable sensors. In *2017 1st International Conference on Next Generation Computing Applications (NextComp)*, pages 169–174, July 2017. doi: 10.1109/NEXTCOMP.2017.8016194.

[30] W. H. ORGANIZATION. Fifty-eighth world health assembly. page 109, 2005.

[31] Qt. Qt in medical. URL https://www.qt.io/qt-in-medical/. Retrieved January 29, 2018.

[32] A. Sbrollini, A. Agostinelli, L. Burattini, M. Morettini, F. D. Nardo, S. Fioretti, and L. Burattini. Ctg analyzer: A graphical user interface for cardiotocography. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2606–2609, July 2017. doi: 10.1109/EMBC.2017.8037391.

[33] Sensoria. Sensoria fitness homepage, 2017. URL http://www.sensoriafitness.com/. Retrieved January 1, 2018.

[34] Sensoria. Sensoria fitness technology, 2017. URL http://www.sensoriafitness.com/technology. Retrieved January 1, 2018.

[35] Tekscan. Tekscan homepage, 2017. URL https://www.tekscan.com. Retrieved December 23, 2017.

[36] Tekscan. F-scan, 2017. URL https://www.tekscan.com/products-solutions/systems/f-scan-system. Retrieved December 23, 2017.

[37] Tekscan. F-scan in practice, 2017. URL https://www.tekscan.com/applications/podiatrist-dr-debrule-uses-f-scan-in-practice. Retrieved December 23, 2017.

[38] Tekscan. Footmat software for clinicians, 2017. URL https://www.tekscan.com/products-solutions/software/footmat-software-clinicians. Retrieved December 23, 2017.

[39] XSens. Xsens homepage, 2017. URL https://www.xsens.com/products/xsens-mvn-analyze/. Retrieved December 20, 2017.

[40] XSens. Xsens mvn analyse, 2017. URL https://www.xsens.com/products/xsens-mvn-analyze/. Retrieved December 20, 2017.

A

S.C.I. AND S.M.I. REAL TIME SESSION

Figure 44 illustrates the actions taken over time to complete a full therapy session.

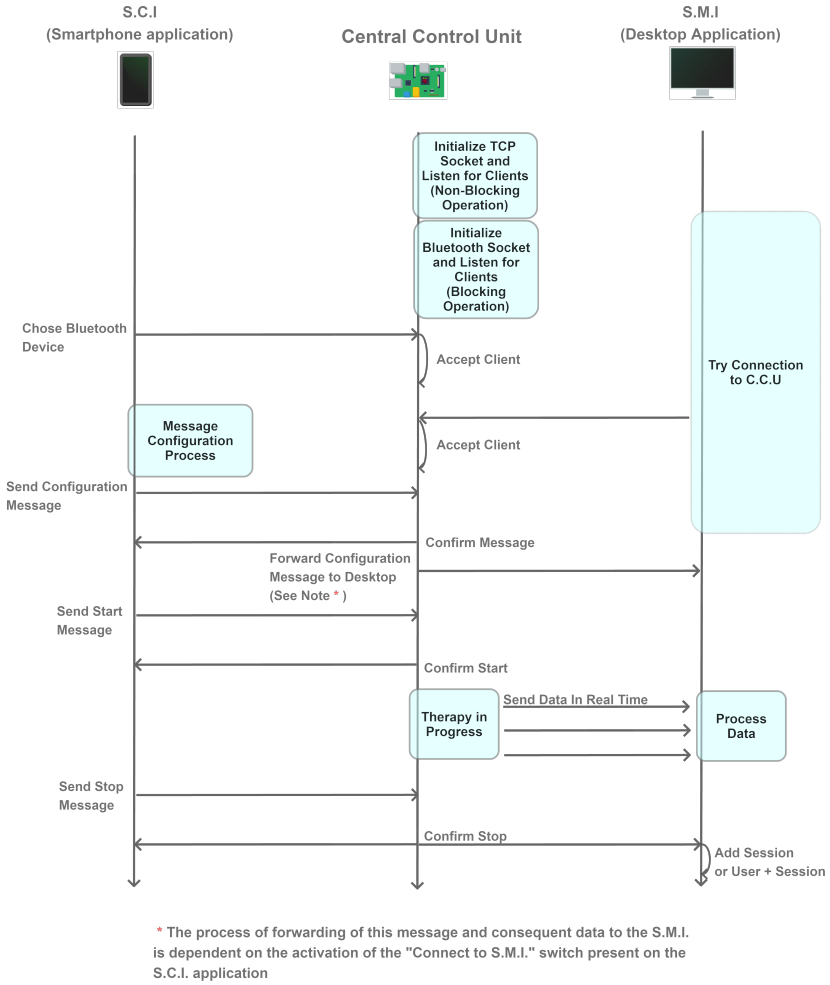


Figure 44.: Interaction between major SmartOs's components over time

The aim of the present appendix is to provide a visual representation of the execution of a therapy session in real time, this particular objective requires interaction between the major SmartOs's components in the scope of this dissertation, the S.C.I, S.M.I and C.C.U.

In this particular instance the therapy session encompasses the connection of both the mobile and desktop applications to the C.C.U, followed by the creation of a "configuration" message and consequent propagation to the C.C.U and S.M.I. Once started, session data is sent from the C.C.U to the S.M.I and processed, until the "stop therapy" message is sent from the S.C.I to the C.C.U.

A video with the execution of the actions shown in Figure 44 can also be found in the following link: https://youtu.be/5xWA0HUB2-o

NB: place here information about funding, FCT project, etc in which the work is framed. Leave empty otherwise.