



Departamento de Ciências e Tecnologias da Informação

Smart Object for Physical Rehabilitation Assessment

Pedro Martim Valente Lima Frango

A Dissertation presented in partial fulfillment of the Requirements for the Degree of Master

Engenharia Telecomunicações e Informática

Supervisor:

Dr. Octavian Postolache, Assistant Professor,
ISCTE-IUL

Co-Supervisor:

Dr. Vítor Viegas, Assistant Professor,
Escola Naval

October 2018

Acknowledgments

I would like to thank my supervisor and co-supervisor, Prof. Doctor Octavian Postolache and Prof. Vítor Viegas for all their availability, support and encouragement given during the development of my dissertation.

To my family, specially my parents, for giving me the opportunity and encouragement to finish my master's degree.

To my girlfriend, Sara Vieira Fernandes who never stopped supporting me and my friends that kept encouraging me.

Finally, I would like to thank Instituto de Telecomunicações (IT-IUL) and Mr. José Gouveia for providing all the material for the project necessary to complete the project.

To all those above named, I sincerely thank you.

Abstract

The technologies associated with smart healthcare are a reality nowadays, however in the physical therapy area there is still lack of patient monitoring during the physical rehabilitation and common usage of walking aids by the patients affected by lower limb impairments. Currently there are fewer systems that provide the patient monitoring during the rehabilitation process by physiotherapists, which may lead to less adequate diagnostic techniques for the patient's physical condition. The dissertation presents a solution to this problem by relying on smart equipment used in physical rehabilitation, more precisely a crutch.

By embedding multiple smart sensors on crutches, the physiotherapist will be provided appropriate information regarding the interaction between the patient and the walking aids through a mobile application, developed for Android systems, which will receive data from the sensors via Bluetooth. All the data collected will be stored in a local database located on the physiotherapist's mobile device and also on a remote server, giving the possibility of having a full offline application.

This system allows for any session previously done to be consulted, which results in the possibility of visualizing historical values and comparing them with different sessions, allowing the physiotherapist to analyze the evolution of the patients.

Keywords: Physical Therapy, Wireless Sensor Network, Internet of Things, Mobile Application, Monitoring.

Resumo

As tecnologias associadas à saúde são uma realidade na atualidade, porém na área de fisioterapia ainda há falta de monitorização dos pacientes durante a fisioterapia e o uso de objetos que auxiliam o movimento pelos pacientes afetados por deficiências nos membros inferiores. Atualmente, existem poucos sistemas que proporcionam a monitorização do paciente durante o processo de reabilitação por fisioterapeutas, o que pode levar a técnicas de diagnóstico menos adequadas para a condição física do paciente. A dissertação apresenta uma solução para este problema, contando com equipamentos inteligentes utilizados em fisioterapia, mais precisamente uma muleta.

Ao incorporar vários sensores inteligentes em muletas, o fisioterapeuta receberá informações adequadas sobre a interação entre o paciente e as muletas através de uma aplicação móvel, desenvolvida para sistemas Android, que receberá dados dos sensores via Bluetooth. Todos os dados recebidos serão armazenados numa base de dados local localizada no dispositivo móvel do fisioterapeuta e também num servidor remoto para fins de sincronização, dando a possibilidade de ter um uma aplicação completamente offline.

Palavras-Chave: Fisioterapia, Rede de Sensores Wireless, Internet das Coisas, Aplicação Móvel, Monitorização.

Contents

Acknowledgments	i
Abstract	ii
Keywords	ii
Resumo	iii
Contents	iv
Tables	vi
Figures	vii
List of Acronyms	ix
Chapter 1 – Introduction	1
1.1. Motivation and Overview	1
1.2. Objectives and Research Questions	2
1.3. Research and Planning Method	3
1.4. Structure of the Dissertation	4
Chapter 2 – State of Art	5
2.1. Existing Projects	5
2.1.1. Real time patient monitoring system based on Internet of Things	5
2.1.2. Development of a System for Monitoring and Tracking of Physiotherapeutic Movements in Patients with Neurological Diseases	6
2.1.3. Assisting Physical Therapy with Wireless Sensor Networks	8
2.1.4. The Pediatric SmartShoe: Wearable Sensor System for Ambulatory Monitoring of Physical Activity and Gait	8
2.1.5. Physical Rehabilitation Assessment based on Smart Training Equipment and Mobile Apps	9
2.1.6. Smart Cane: Instrumentation of a Quad Cane with Audio-feedback Monitoring System for Partial Weight-Bearing Support.....	10
2.2. Android Mobile Applications	12
2.2.1. 9zest Stroke Rehab & Recovery	12
2.2.2. Shoulder Rehabilitation Exercises.....	12
2.2.3. MEHAB – Mobile Physical Therapy	13
2.2.4. PhysioU: Complete Rehab Guide.....	13
2.2.5. Healure – Physical Therapy.....	14
Chapter 3 – System Description	15
3.1. Overview.....	15
3.2. Users and Applications	16
3.3. SmartCrutch System	17

Chapter 4 – Hardware	19
4.1. Crutch.....	19
4.2. Force Sensor	20
4.3. Gait Sensor.....	24
4.4. Load Cell sensor	26
4.5. Microcontroller	29
4.6. Bluetooth Adapter.....	31
4.7. PCB Board	33
4.8. Battery.....	34
4.9. Mobile devices	35
4.10. Electrical Circuit	37
4.11. Prototype	38
Chapter 5 – Software.....	41
5.1. Arduino IDE	41
5.2. Android Studio.....	44
5.3. Database and Server.....	49
5.3.1. Local Database - SQLite.....	49
5.3.2. Remote Database - MySQL.....	50
5.4. Server	53
5.5. PHP Scripts	55
Chapter 6 – Mobile Application	58
6.1. Main Features	58
6.2. Design and Implementation	62
Chapter 7 – Results	83
7.1. Individual sensor testing	83
7.2. Patient tests results.....	87
Chapter 8 – Conclusions and Future Work	92
8.1. Conclusions.....	92
8.2. Future Work.....	93
References.....	95
Annex A – Articles	101
Annex B – User Manual	114
Annex C – Technical Manual	147

Tables

Table 1 – Threshold levels of sensor readings	6
Table 2 – FlexiForce A201 performance specifications.....	21
Table 3 – FlexiForce calibration values	23
Table 4 – SNC2C6 load cell sensor performance specifications.....	26
Table 5 – Load cell calibration values.....	28
Table 6 – Raspberry Pi Zero and Arduino Nano specifications	31
Table 7 – Samsung Galaxy J5 and Samsung Galaxy Tab S2 specifications.....	36
Table 8 – Patient information for testing purposes.....	87
Table 9 – Extra FSR sensor data calculated	88
Table 10 – Extra load cell sensor data calculated.....	89
Table 11 – Extra IMU sensor data calculated.....	90

Figures

Figure 2.1 – Modern ICU unit	5
Figure 2.2 – Mobile application developed for ICU based on IoT system	6
Figure 2.3 – Radio humeral joint flexion exercise	7
Figure 2.4 – WSN architecture	8
Figure 2.5 – SmartShoe	9
Figure 2.6 – SmartWalker	10
Figure 2.7 – Heart ratio evolution displayed by ‘m-health’ application	10
Figure 2.8 – SmartCane prototype and hardware used.....	11
Figure 2.9 – 9zest Stroke Rehab & Recovery APP	12
Figure 2.10 – Shoulder Rehabilitation Exercises APP	13
Figure 2.11 – MEHAB – Mobile Physical Therapy APP.....	13
Figure 2.12 – PhysioU: Complete Rehab Guide APP	14
Figure 2.13 – Healure – Physical Therapy APP	14
Figure 3.1 – SmartCrutch System	15
Figure 3.2 – SmartCrutch system application’s languages	16
Figure 3.3 – Time flow diagram of SmartCrutch system	17
Figure 4.1 – Crutch handling	20
Figure 4.2 – FSR’s load-resistance and load-conductance plot	20
Figure 4.3 – FSR 406.....	21
Figure 4.4 – FlexiForce A201.....	22
Figure 4.5 – FlexiForce Adapter.....	22
Figure 4.6 – Calibration curves and linear equations for FlexiForce sensors	23
Figure 4.7 – AltIMU-10 v4	24
Figure 4.8 – Crutch representing Euler angles	25
Figure 4.9 – SNC2C6 Load Cell	27
Figure 4.10 – Load cell conditioning circuit	27
Figure 4.11 – Load cell calibration curve.....	29
Figure 4.12 – Raspberry Pi Zero	30
Figure 4.13 – Arduino Nano.....	30
Figure 4.14 – Bluetooth SPI friend.....	31
Figure 4.15 – Bluetooth module HC-06	32
Figure 4.16 – Schematic for Bluetooth HC-06 module configuration	33
Figure 4.17 – PCB board	33
Figure 4.18 – PCB board connections	34
Figure 4.19 – Power bank.....	34
Figure 4.20 – Samsung Galaxy J5	35
Figure 4.21 – Samsung Galaxy Tab S2	36
Figure 4.22 – SmartCrutch system electrical circuit	37
Figure 4.23 – SmartCrutch initial prototype.....	38
Figure 4.24 – SmartCrutch box components	39
Figure 4.25 – SmartCrutch final prototype.....	39
Figure 4.26 – SmartCrutch final prototype 2.....	40
Figure 5.1– Arduino IDE sketch example	42
Figure 5.2– Program uploading to Arduino boards	43
Figure 5.3 – Data on the relative number of devices with a certain version of Android platform collected during a 7-day period ending on July 23, 2018.....	44
Figure 5.4 – Android Studio IDE	45
Figure 5.5 – Build process of an Android Application.....	46

Figure 5.6 – Layouts of different activities	47
Figure 5.7 – JAVA class (physiotherapist register activity).....	48
Figure 5.8 – XML file (physiotherapist register layout)	49
Figure 5.9 – SQLite functions	50
Figure 5.10 – Android Debug Database example	50
Figure 5.11 – Example of query for patient SQL table	51
Figure 5.12 – Database diagram	52
Figure 5.13 – XAMPP software	54
Figure 5.14 – Main folder and files on the server	54
Figure 5.15 – HTTP request-response protocol	55
Figure 5.15 – PHP script example: adding physiotherapists to MySQL database	56
Figure 5.16– PHP script example: sending physiotherapists to the SQLite database	57
Figure 6.1 – Line chart and Bar chart example	59
Figure 6.2 – Life cycle of a thread.....	59
Figure 6.3 – Client-Server socket communication	61
Figure 6.4 – AsyncTask methods	61
Figure 6.5 – IoPhyr application logo	62
Figure 6.6 – Login activity of the IoPhyr application	63
Figure 6.7 – Register activity for physiotherapists.....	64
Figure 6.8 – User area activity.....	65
Figure 6.9 – Physiotherapist profile activity	66
Figure 6.10 – Patients activity	67
Figure 6.11 – New patient activity	68
Figure 6.12 – View patients’ activity	69
Figure 6.13 – Selecting patient for session activity.....	70
Figure 6.14 – Sessions activity	71
Figure 6.15 – New session activity.....	72
Figure 6.16 – Live FSR sensor data	73
Figure 6.17 – Live load cell sensor data.....	74
Figure 6.18 – Live IMU sensor data.....	75
Figure 6.19 – Session name and notes.....	76
Figure 6.20 – View session activity	77
Figure 6.21 – View single session	78
Figure 6.22 – FSR data analysis	79
Figure 6.23 – Load cell data analysis	80
Figure 6.24 – IMU data analysis	81
Figure 6.25 – Negative patient progress considering FSR data	82
Figure 7.1 – FSR sensor values test.....	83
Figure 7.2 – Load cell sensor values test.....	84
Figure 7.3 – IMU sensor front inclination	84
Figure 7.4 – IMU sensor back inclination	85
Figure 7.5 – IMU sensor right inclination	86
Figure 7.6 – IMU sensor left inclination	86
Figure 7.8 – FSR sensor data of a session for each patient	88
Figure 7.9 – Load cell sensor data of a session for each patient	89
Figure 7.10 – IMU sensor data of a session for each patient.....	90
Figure 8.1 – Body temperature sensor MAX30205 (left) and heart rate temperature sensor SEN0203 (right)	93
Figure 8.2 – MQTT protocol	94

List of Acronyms

3G – 3rd Generation

4G – 4th Generation

ADC – Analog to Digital Converter

API – Application Programming Interface

APK – Android Package

APP – Application

ASCII – American Standard Code for Information Interchange

BLE – Bluetooth Low Energy

COM – Communication port

CPU – Central Processing Unit

DEX – Dalvik Executables

ECG – Electrocardiography

FSR – Force Sensor Resistor

FTP – File Transfer Protocol

GPIO – General-Purpose Input/Output

GPS – Global Positioning System

HTTP – Hypertext Transfer Protocol

ICU – Intensive Care Unit

IMU – Inertial Measurement Unit

IOT – Internet of Things

MISO – Master Input/Slave Output

MOSI – Master Output/Slave Input

MQTT – Message Queuing Telemetry Transport

OS – Operating System

PCB – Printed Circuit Board

ISCTE-IUL
Smart Object for Physical Rehabilitation Assessment

PPI – Pixels Per Inch

SCK – Serial Clock

SFTP – SSH File Transfer Protocol

SPI – Serial Peripheral Interface

SQL – Structured Query Language

SS – Slave Select

SSH – Secure Shell

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver-Transmitter

UI – User Interface

USB – Universal Serial Bus

UUID – Universally Unique Identifier

WSN – Wireless Sensor Network

Chapter 1 – Introduction

1.1. Motivation and Overview

The health department area is one of the most important and essential area in our lives, providing new methods to treat diseases and impairments, developing new medicines, improving resources and equipment used to diagnose and treat patients, among others.

Assistive technologies became central to medicine during the 20th century. Advances in science, engineering and manufacturing were applied to medical problems. Technologies such as hearing aids, artificial limbs and mobility aids became more sophisticated while computerized machines in hospitals monitored patients continuously and performed diagnostic tests quicker and more accurately [1].

Physical rehabilitation is one of many areas in the health department that has had more demand recently due to population aging, car accidents, sport injuries, among other causes, which leads to a need to optimize the rehabilitation process according with patient evolution.

Improving the performance in the healthcare field is a big obstacle in today's era, where some of the main issues are dealing with nursing staff availability and treatment costs [2-3]. With the use of smart objects, the physiotherapists should be able to get a more appropriate reading of the patient's session which would result in a better and more appropriate treatment [4-5].

The incentive to work for a professional is vital in any profession and according to *Forbes* magazine, physiotherapists were ranked as having one of "The Ten Happiest Jobs" according to articles published in 2011 and 2013, while an expected job growth between 2010 and 2020 was an astonishing 39% [6].

Physical therapy's innovations in technology are a work in progress and they offer new opportunities for improved treatment and patient connection. Advances in robotics and bionics provide physiotherapists with better decision making, therefore leading to an improved outcome and patient experience [7]. Analyzing the data received from the system is challenging because different factors must be considered, like patient's age, sex, walking style, walking impairments, among others [8].

Internet of Things has become increasingly important in urban development, using a vast network of sensors covering a city or part of it. For example, in some cities where

the technology is already in place, it is possible on a day-to-day basis to remotely check the availability of several car parking lots through a mobile application [9] or even live in a smart-house, where everything from air conditioning, underfloor heating, security alarm, lightning, sprinklers, and many other house appliances can be easily controlled by the end user, also using a mobile application [10].

Recent advances in IoT technologies are providing a development of smart systems with the main purpose of improving health care. Some examples are automatic identification and tracking of people and biomedical devices, correct drug-patient associations, and patient monitoring [11]. By designing a portable low cost and non-intrusive IoT system capable of monitoring the patient's sessions, it is possible to analyze certain data that is invisible to the naked eye, like gait patterns.

The data needs to be visible to the physiotherapists, and the most common device to do so would be a mobile device, which are used on a daily basis. In 2016 an estimated 62,9% of the population worldwide owned a mobile phone [12]. In the 4th Quarter of 2016, the number of phones sold with Android system was 352 million reaching an 81,7% market share, while iOS system phones only sold 77 million phones which resulted in a 17,9% market share [13].

1.2. Objectives and Research Questions

The objective of this research is to develop a SmartCrutch system capable of retrieving and storing data received from sensors that are attached to a crutch used by patients that are in a physical therapy session. This system should allow the physiotherapists to visualize the patient's clinical status and overall progress. Based on those results, the physiotherapist should be able to make new decisions or try new approaches to the training sessions in case the progress is going slowly or in a wrong way.

The patient's data that will be visualized by physiotherapists will be extracted from sensors, which need to be chosen according to the object's structure, in this case, a crutch. Data from the sensors needs to be processed, so a microcontroller is needed in the system and depending on which one is chosen, an adapter may also be needed in order for the data to be sent from the microcontroller to the mobile device.

Visualizing the data obtained requires a mobile application developed solely for the physiotherapists that could support both tablet devices and mobile phones, ensuring that more than one type of device can be used in the system. The application should be entirely

offline based, meaning there is no need for an Internet connection for it to view old session data or start new sessions.

The results obtained from the sensors should be visible in real time and also should be stored in a local database in which they are statistically treated to elaborate metrics that will be used by physiotherapists to evaluate the evolution of the patient. Charts should be displayed to the physiotherapists which would indicate a single session's data and overall data, providing an idea of how the patient's progress is going.

To avoid the loss of local data it should be possible to synchronize the phone/tablet's application database, with another database located on a remote server.

This project aims to obtain an answer to certain questions, such as:

- Will the mobile application be user friendly, so the physiotherapist has no problems understanding the data received?
- Can the system be inserted on a crutch without obstructing the patient's movement?
- Which wireless technology would be more appropriate to the system for short ranged data transmission?

1.3. Research and Planning Method

The research method used in this project is divided in 6 different steps described below:

- **1st Step → Identifying the problem** – In this phase it is important to understand the lack of technology currently used in physical therapy so we can set objectives to improve overall treatment;
- **2nd Step → Definition of objectives** – This step involves studying and choosing different technologies and materials that can help with patients monitoring and development in their physical rehabilitation;
- **3rd Step → System prototype development and construction** – Several components of the project are developed at this stage, such as:
 - System description and architecture;
 - Development and creation of an electrical system based on a microcontroller to be attached to a crutch;

- Mobile device application designed for physiotherapist's use in order to visualize patient data;
- Creation of local and remote databases for data storage.
- **4th Step → System prototype implementation** – By implementing the components in the previous stage, a project prototype will be created;
- **5th Step → System prototype tests** – Series of tests, adjustments and modifications to the system prototype will be performed in order to obtain a final prototype;
- **6th Step → Evaluation** – The final prototype of the project obtained in the last step must be tested and evaluated.

1.4. Structure of the Dissertation

This dissertation is divided in seven chapters:

- Chapter 2 includes a literature review where some sensors can be applied in the physical therapy area.
- Chapter 3 describes this project's system description
- Chapter 4 contains all the hardware used in the project, including sensors, microcontrollers, mobile devices.
- Chapter 5 describes the software used, such as the microcontroller's firmware, local and remote database and mobile application development.
- Chapter 6 considers the experimental results and processed data of the application itself.
- Chapter 7 mentions the conclusions of the project and future work.
- Lastly, two accepted articles, a user manual regarding the mobile application usage and a technical manual explaining some of the main functions of the application can be seen in the annex.

Chapter 2 – State of Art

2.1. Existing Projects

This chapter presents the research on sensors used in the health department area, more specifically physical therapy. The existing projects researched and presented below provide new ways of monitoring patients to ensure that the undergoing treatment is done correctly.

2.1.1. Real time patient monitoring system based on Internet of Things [14]

The implementation of sensors is continually expanding in several different areas. Since the focus of our project is physical therapy, the area that fits the most for research on similar topics is the health department area.

A project developed by Mohammad S. Udin [14] focuses on an IoT system developed for patients in ICU care, providing the ability to detect the critical condition of patients and immediately notifying the doctors and nurses for a quick intervention.



Figure 2.1 – Modern ICU unit [15]

Data processing is done by an Arduino 101 and then sent wirelessly via an Ethernet Shield to an open source IoT platform, '*thinger.io*' [16] that would operate as a Cloud infrastructure, saving all the data collected from the sensors. Sensors used measured body and room temperature and humidity, pulse, ECG, unexpected patient movement and blood pressure. Table 2.1 indicates the thresholds levels of sensor readings applied to the system to indicate a patient emergency. If any condition in Table 2.1 was verified a member of the hospital in charge of that patient would be immediately notified.

Table 1 – Threshold levels of sensor readings

Sensor	Threshold level
Heart rate	Less than 50 and greater than 120
Temperature	Less than 35 and greater than 39 (Celsius)
Humidity	Less than 40% and greater than 55%
Blood oxygen level	Under 90%
Upper blood pressure	Less than 120 and greater than 180
Lower blood pressure	Less than 80 and greater than 110

Notification is done via a mobile application which is connected directly to the Cloud and therefore provides the possibility to view real time data using different types of charts and gauges. Figure 2.2 shows the layout of the application developed.



Figure 2.2 – Mobile application developed for ICU based on IoT system [14]

2.1.2. Development of a System for Monitoring and Tracking of Physiotherapeutic Movements in Patients with Neurological Diseases [17]

Another way of improving physical rehabilitation is through the use of serious games on a virtual environment, using a Microsoft Kinect API to do so. Many studies have proven that serious games that include a feedback system, such as scores, sounds, progress bars and vibration can encourage patients and motivate them to continue the treatment [18].

In [17], the author's objectives are to develop a monitoring application with the purpose of helping the treatment of patients whose mobility is affected because of neurological disorders, based on the use of the Microsoft Kinect API. Three types of exercises were created in order to test and monitor through image processing techniques and it is important to note that these exercises were not chosen at random. In fact, the methods used to obtain the three-dimensional position of the points corresponding to the joints that are moving as well as the distance between them and the angles between segments are the only ones that can be monitored. The physiotherapist is able to access all the data regarding any training exercise previously done, cross reference it and finally plot charts regarding patient overall evolution. Figure 2.3 shows an example of one exercise, the radio humeral joint flexion, as well as the resulting imaging processed by the application.

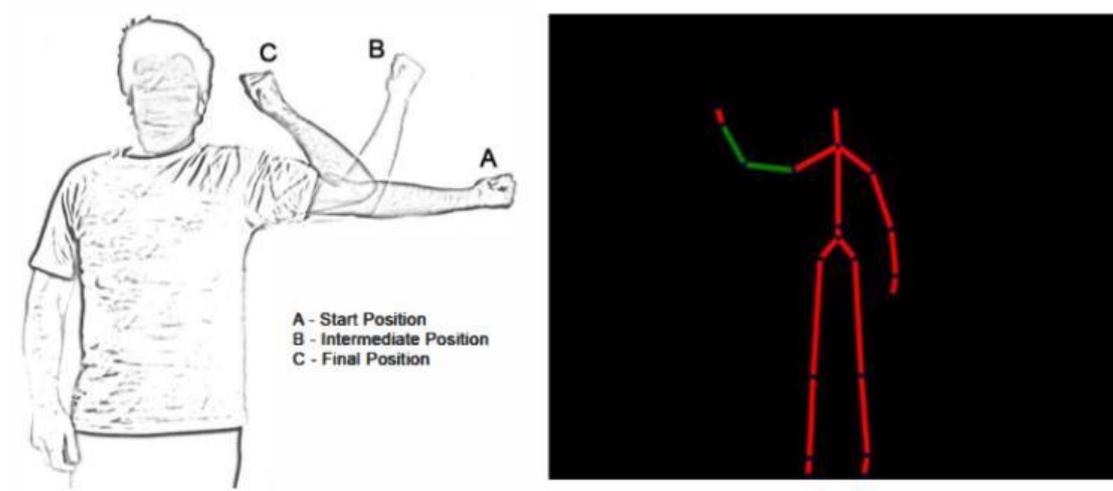


Figure 2.3 – Radio humeral joint flexion exercise [17]

2.1.3. Assisting Physical Therapy with Wireless Sensor Networks [19]

This project [19] is based on a WSN architecture and its main purpose is to provide real time supervision to physical therapy patients that are undergoing training sessions. Several data are retrieved, such as joint angles, oxygen saturation and pulse. Monitoring can be done either locally or remotely with a less intrusive system than the traditional ones.

In [19] it is also demonstrated that the system developed is also possible in a hydrotherapy session, meaning it would be able to function properly under water. Data collected from the sensors is sent to a local computer, being able to provide real time data visualization. This data is stored in the local computer and it provides the ability for a remote user, such as another doctor, to access the data stored in the local computer via Internet. The system doesn't include training plans nor scheduling. Instead, this should be done by the physical therapist in the traditional way. An architecture of the system can be seen in Figure 2.4.

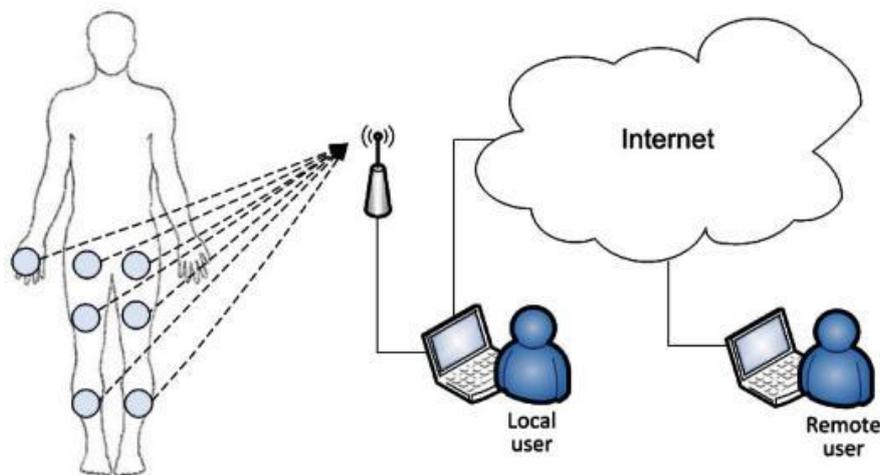


Figure 2.4 – WSN architecture [19]

2.1.4. The Pediatric SmartShoe: Wearable Sensor System for Ambulatory Monitoring of Physical Activity and Gait [20]

In [20], the author's idea is to implement a non-intrusive system that is capable of determining gait and pressure applied on a smart shoe dedicated for children that are affected by cerebral palsy. As it was seen in [17], gait monitoring is possible through the use of a Kinect system, but the patient is limited to the training area. This system [20] provides usage on a free environment and can be used on a daily basis in activities such as walking, sitting and standing. Figure 2.5 shows the prototype developed.



Figure 2.5 – SmartShoe [20]

Only two types of sensors were used, five force sensors and a 3-D accelerometer. The force sensors were located on the heel, on the head of the first, third and fifth metatarsal bones and on the base of the big toe, and these sensors were all soldered on a flexible PCB board. As can be seen in Figure 2.5, a small plastic box contains the accelerometer, a rechargeable battery and a Bluetooth 2.1 wireless interface that would send data to a Microsoft Windows smartphone.

2.1.5. Physical Rehabilitation Assessment based on Smart Training Equipment and Mobile Apps [21]

Postolache et. al. developed a system which focuses on smart walkers and crutches for physical therapy monitoring. Mobile devices are used for visualization of signals given by multiple sensors located in the smart training equipment.

Different types of walkers were considered: a regular walker, a two-wheel walker and a four-wheel walker. Each type of walker is associated with a specific type of sensors that will provide certain data, such as force applied, 3D acceleration and motion capture. Finally, communication used in the system is completely Wireless, resorting to the use of Bluetooth and ZigBee, sending the session's data through the Internet to a remote's server database.



Figure 2.6 – SmartWalker [21]

A mobile application, '*m-health*' was developed in order to retrieve and analyze data on a mobile device. This application was developed in Android Studio which includes a vast set of development tools, such as debug, libraries and built in emulators. Figure 2.7 shows an example of the data provided by the application considering a patient's heart ratio evolution.

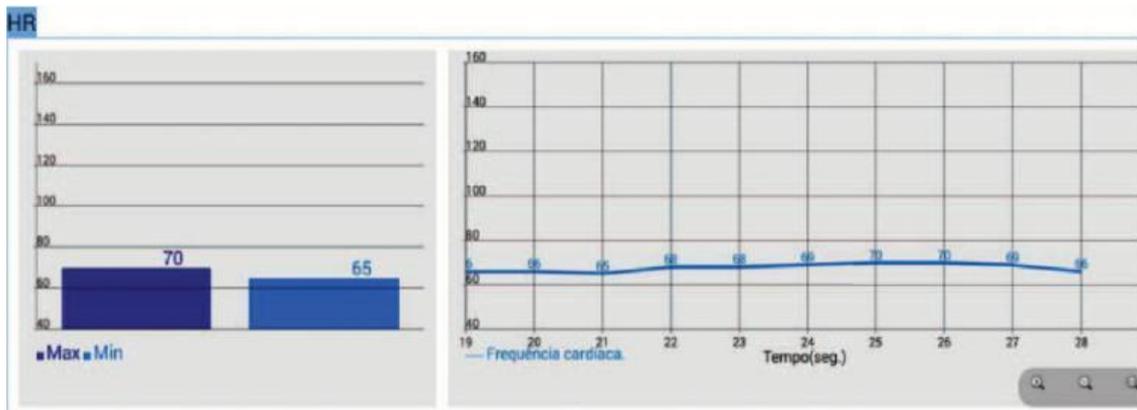


Figure 2.7 – Heart ratio evolution displayed by '*m-health*' application [21]

2.1.6. Smart Cane: Instrumentation of a Quad Cane with Audio-feedback Monitoring System for Partial Weight-Bearing Support [22]

Another object commonly used in physical rehabilitation is a cane, or in this case, a quad cane, which provides more balance to the patient when being used. This project [22] developed by Joseph Mercado focuses on proper cane handling by patients that are undergoing physical therapy treatment. A SmartCane system was developed with the purpose to monitor the weight-bearing performance of the patient while also providing an audio feedback whenever the prescribed weight application has not been accomplished. This value usually ranges from 20 to 25% of the patient' body weight.

Whenever the force applied is not on the accepted range of forces an alarm triggers, providing the user with information to either increase or decrease applied force, and it will be continuous until the force applied by the patient is within the accepted range of forces. Data is stored and visualized in a computer, keeping a log of force measurements available through a GUI developed in an open source program, StampPlot Pro. Figure 2.8 shows the SmartCane prototype including the hardware used.

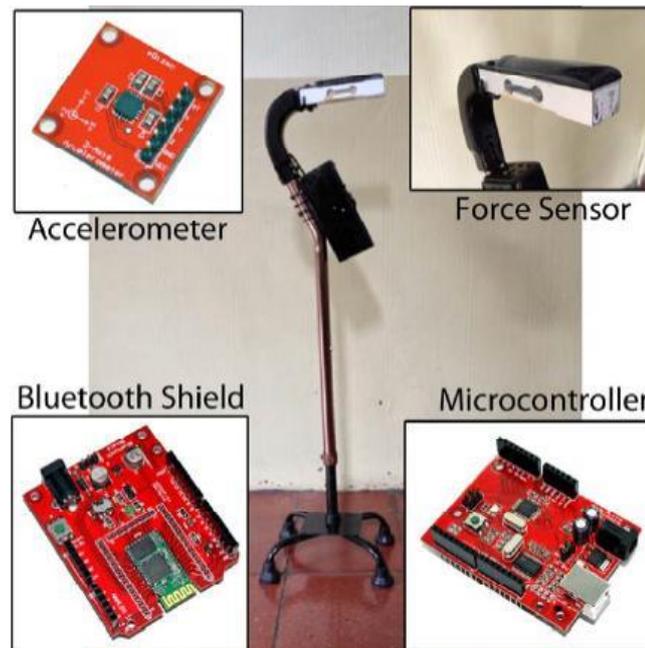


Figure 2.8 – SmartCane prototype and hardware used [22]

An Arduino based microcontroller, Gizduino, containing an Atmega 168 microcontroller was used for sensor data processing, connected to a Bluetooth Shield that would be in charge of sending the processed data to a computer for analysis.

As for sensors, a load cell with a Wheatstone configuration which produces an output signal proportional to the applied force, paired with an instrumentation amplifier AD620 was used in order to retrieve electrical signals big enough to analyze. This sensor was placed on the cane handle as can be seen in Figure 2.8. A low cost 3-axis accelerometer, ADXL335 was also used with the purpose of detecting cane swing. When the tension applied was constant at 1.15V this would indicate that the cane was steady, but when it started movement the value would slightly change, serving as a threshold to initiate the force measurements.

2.2. Android Mobile Applications

Considering [12-13] and the fact that Android is an open source software it was decided that the project would be developed for Android systems. Some examples of medical applications (APP's) available on Google Play are available below and can be installed by any Android device:

2.2.1. 9zest Stroke Rehab & Recovery (see Figure 2.9): *“9zest Stroke Rehab aims to enhance the quality of life by providing easy, affordable, and at-home solution for stroke recovery. The exercises are designed by certified therapists and help in activities of daily living such as eating, drinking, walking, driving, brushing teeth, picking a pen, wearing a shirt, etc, reducing stroke challenges”* [23];

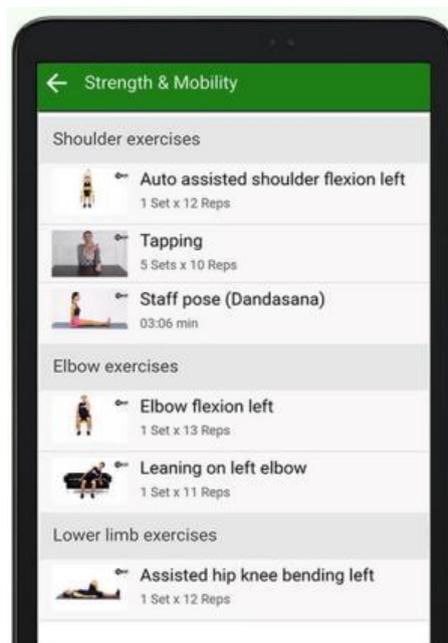


Figure 2.9 – 9zest Stroke Rehab & Recovery APP [23]

2.2.2. Shoulder Rehabilitation Exercises (see Figure 2.10): *“After an injury or surgery, an exercise conditioning program will help you return to daily activities and enjoy a more active, healthy lifestyle. Following a well-structured conditioning program will also help you return to sports and other recreational activities”* [24];



Figure 2.10 – Shoulder Rehabilitation Exercises APP [24]

2.2.3. MEHAB – Mobile Physical Therapy (see Figure 2.11): “*Mehab provides evidence-based self-directed rehabilitation protocols via mobile devices. A low-cost alternative to formal care. It is a comprehensive program that includes simple to follow instructions and information with video demonstration of exercises, and tracking and reporting*” [25];

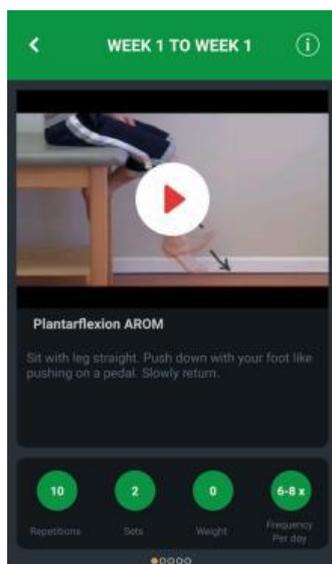


Figure 2.11 – MEHAB – Mobile Physical Therapy APP [25]

2.2.4. PhysioU: Complete Rehab Guide (see Figure 2.12): “*The apps cover a wide span of key clinical content related to Physical Therapy and Rehabilitation. Over 1,000 high definition videos, instructions, research citations and clinical reasoning insights. Used in rehabilitation clinics and education programs throughout the world*” [26];

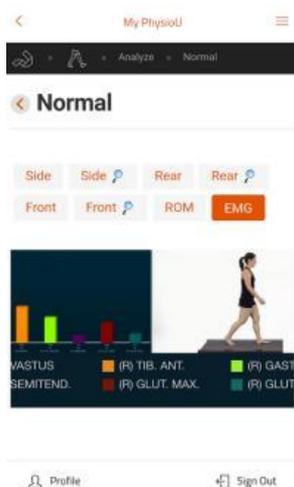


Figure 2.12 – PhysioU: Complete Rehab Guide APP [26]

2.2.5. Healure – Physical Therapy (see Figure 2.13): *“Healure is a physiotherapy exercise app. It relieves neck pain, shoulder pain, wrist pain, elbow pain, back pain and knee pain. The plans are prepared according to inputs taken (in form of assessments) from our clients. These plans are suitable for all age group between 18-65 years”* [27].

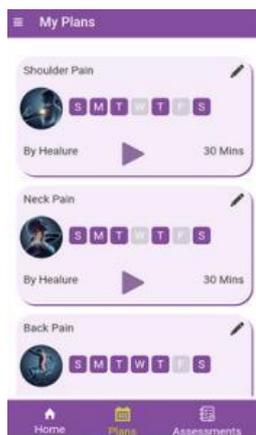


Figure 2.13 – Healure – Physical Therapy APP [27]

In conclusion to this chapter, we realize that there are some challenges ahead of relating the physical therapy area with an IoT system. Most of the treatments require monitoring by physical therapists, which means higher costs for the patient. The introduction and education of these new technological systems to the physical therapists and the development of new sensors that can help with patient monitoring are also challenges that need to be considered in this area that is being introduced in IoT.

Chapter 3 – System Description

3.1. Overview

The SmartCrutch system purpose is to be able to monitor patients in physical rehabilitation sessions that are affected with motor activity limitations and lower limb impairments, allowing for the physiotherapist to have access to the data obtained from the sessions performed, and, with that data, ensure that the patient is getting the best possible treatment. Figure 3.1 shows the entire system architecture.



Figure 1.1 – SmartCrutch System

Instead of using a regular crutch, the patient should use a SmartCrutch so that the physiotherapist can monitor the patient's development. Several sensors are connected to an Arduino which will be then start sending data using Bluetooth to the physiotherapist's Android mobile device once the session starts. Live data can be visualized so the physiotherapist can be aware of which exercises are harder for the patient, and as the session progresses, all data is stored in a local application database for further analysis. It is also possible, through the use of PHP scripts, to provide wireless communication between the mobile device and a remote database located in a remote server, providing data synchronization.

3.2. Users and Applications

The SmartCrutch system only has two different types of users: the physiotherapist who oversees managing the application and starting the session, and the patient, who should follow the instructions given by the physiotherapist while in a physical therapy session.

Two different applications were developed in order to form the SmartCrutch system:

- **Arduino Application:** Developed in Arduino IDE [28] which uses C/C++ programming language. This application will be in charge of receiving all the data transmitted by the sensors. Data will be sent to the physiotherapist's mobile device via Bluetooth on a small set interval to make sure there is good data flow;
- **Mobile Application:** Developed for Android OS in Android Studio [29] which uses JAVA and XML languages, serving as a tool for physiotherapists to view and add new patients while also being able to start new sessions and view old ones.

Both application's development languages can be seen below in Figure 3.2.



Figure 2.2 – SmartCrutch system application's languages

3.3. SmartCrutch System

A time flow diagram explaining how the SmartCrutch system works when a session is started can be seen in Figure 3.3. The letters A-F indicate the time frame on which the system is executed. A login system is used in the application, requiring for all the physiotherapists to be registered, by filling out a small form. Each patient can only be assigned to one physiotherapist, and it is the physiotherapist's job to add his patients in the application itself. Each patient should provide information about himself, such as first name, last name, age, phone number, address and e-mail so the physiotherapist can fill out and create the patient in the application in order to start new sessions.

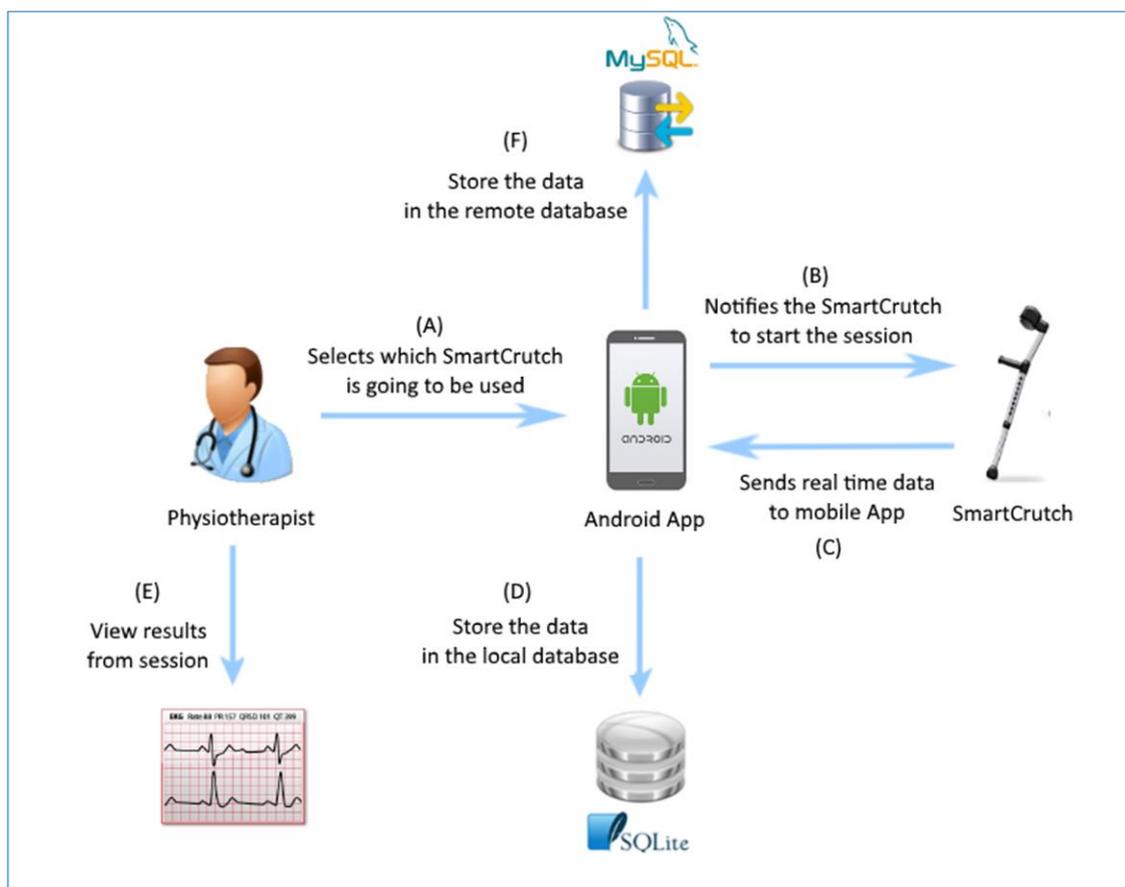


Figure 3.3 – Time flow diagram of SmartCrutch system

After the patient has been created it is necessary to select him/her when starting a new session. Since Bluetooth connection is required, the application includes a button to turn it on or off. After searching for Bluetooth enabled devices in the near area, the physiotherapist should choose the correct SmartCrutch system in case there's more than one turned on (A). Since this system is intended for rehabilitation clinics there is a high probability that there would be more than one system being used.

When the physiotherapist chooses the correct SmartCrutch to connect to, a Bluetooth connection will be established from the Arduino located in the SmartCrutch to the mobile device indicating that the session can be started. This is generally defined as pairing [30] devices (B). The Bluetooth's adapter UUID is used to establish the connection. Besides the UUID, threads [31] and sockets [32] are also important elements regarding connectivity between the devices.

Data is sent from the sensors to the Arduino, which is processed and formatted and will be sent every few milliseconds. This data can be visualized in real time while the session is still in progress, through the use of time charts (C). Data is stored every time it is received and not just at the end of the session. Inserting a small amount of data every few milliseconds instead of inserting the whole session data at the end of the session provides better performance to the overall system (D). After the session has ended the physiotherapist is prompted to insert a session name and write some notes regarding the training session.

The physiotherapist has the option to view any past single training session or overall progress, which includes all the sessions. Individual sessions include elements such as: session date, start time, end time, duration, number, name, notes and sensor values detailed in line charts (see Page 28-29, Figure 3.26 and Figure 3.28 in Annex B – User Manual). Overall progress only includes sensor median values displayed by a bar chart containing individual sessions (see Page 32, Figure 3.31 in Annex B – User Manual).

Storing the data in the remote database is optional but it is recommended to do so after every session in case something happens to the physiotherapist's mobile device. The application initiates a HTTP request by using a PHP script as a middleman, which is hosted on the server. This PHP script will access and connect to the remote MySQL database which will start the process of synchronization, sending the session data located in the mobile device's database to the remote server's database (F).

Chapter 4 – Hardware

After much consideration and research, it was decided that several important hardware components were needed to develop this project, such as:

- A crutch that will include sensors attached to it in order to retrieve important values regarding training sessions;
- Force sensors placed on the crutch's hand support, indicating the hand force applied by the patient;
- A gait sensor that is able to indicate the angle values of the crutch during training, therefore providing gait information to the physiotherapist;
- A load cell sensor that indicates the full weight applied to the crutch by the patient;
- A microcontroller capable of processing sensor data;
- A Bluetooth adapter capable of transmitting sensor data to the physiotherapist's mobile device;
- A board which connects all the electrical devices above with the purpose of reducing the amount of wires;
- A battery with the purpose of making this project an autonomous system;
- Mobile devices with an Android OS for testing purposes.

All the components integration resulted in the SmartCrutch system prototype, which can be seen at the end of this chapter.

4.1. Crutch

Crutches (see Figure 4.1) are an object commonly used in physical therapy which aid patients during their rehabilitation time. However, according to a study published in the *Journal of American Geriatrics Society* (June 2009) it was discovered that 47,000 senior citizens end up in emergency rooms each year because of the improper use of the crutches, which leads to falling and therefore injuries [33].

For this reason, it is very important that the patients that are undergoing physical therapy treatment are using the crutch properly. Some general tips [33] on how to use a crutch are the following:

- Make sure the crutch is about the height of the patient's wrists when the patient's arms are on their side;
- When using a crutch, the patient's arms should be slightly bent when holding on, but the patient shouldn't have to bend forward at the waist to reach it;

- The rubber tips at the bottom of the cane should be periodically checked, making sure they are replaced if they are uneven or worn through.

A single crutch will be used in this project.



Figure 4.1 – Crutch handling

4.2. Force Sensor

Sensors capable of measuring force applied, also known as FSR's [34], should be applied in the hand support of the crutch in order to measure force applied by the patient's grip. This sensor will be placed in the crutch's hand support and will provide the physiotherapist with an idea of the force applied by the patient in the hand support. Resistance in the sensor will be close to infinite when no pressure is applied to the sensor. The sensor's resistance decreases exponentially as the load increases as can be seen in Figure 4.2.

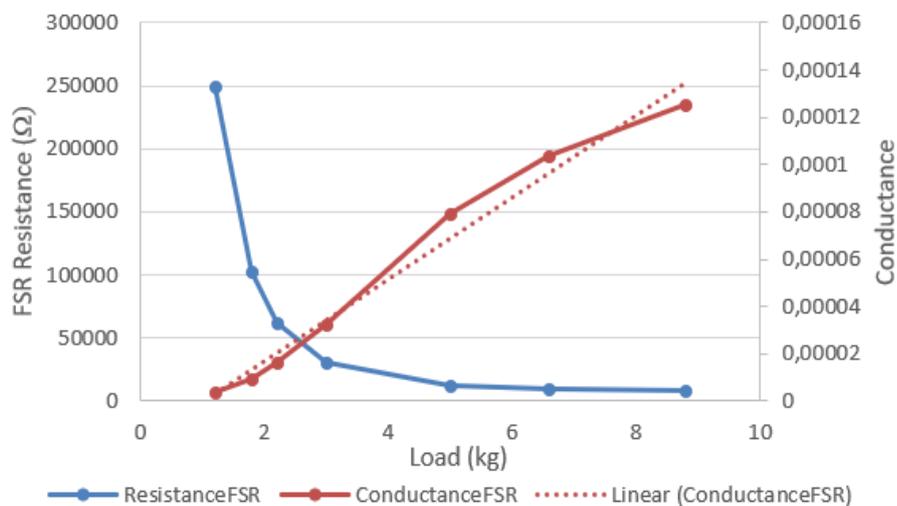


Figure 4.2 – FSR's load-resistance and load-conductance plot

- Initially a **FSR 406** model sensor (see Figure 4.3) [35] was used but it was quickly discovered that this sensor, capable of detecting forces up to 10 kilograms with a sensing area of 1.75x1.5", was used mainly for force detection and not accurate force measuring, therefore proving useless to our application.

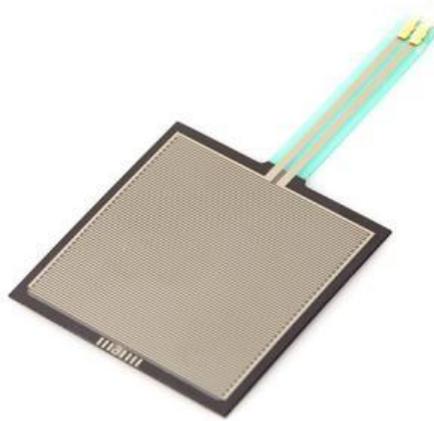


Figure 4.3 – FSR 406

- After further investigation for a new FSR sensor, a **FlexiForce A201** (see Figure 4.4) [36] was considered which worked in the same way as the FSR 406, meaning resistance on the sensor will vary according to the load applied on the sensor. This thin, flexible and customizable sensor have a better linearity than other FSR's while also providing measurements of force up to 445 Newton, or 45 kilograms. Because of the small size of the sensor, with a circular sensing size of only 9.7 millimeters, two of these sensors will be applied to the crutch's hand support to ensure that more area is covered. Some specifications regarding this sensor's performance can be seen in Table 2.

Table 2 – FlexiForce A201 performance specifications

Linearity	$\leq \pm 3\%$ of Full Scale
Repeatability	$\leq \pm 2.5\%$
Hysteresis	$\leq 4.5\%$ of Full Scale
Drift	$\leq 5\%$
Response Time	$\leq 5\mu s$
Operating Temperature	$-40^{\circ}C$ to $60^{\circ}C$



Figure 4.4 – FlexiForce A201

This FlexiForce sensor was tested to evaluate its dynamic performance and while doing so, huge amounts of noise were detected. To counter this, each sensor would be attached to a FlexiForce adapter (see Figure 4.5) [37] that would reduce the existing noise for a result of smoother sampling. This adapter would also help with the calibration of the sensor in order to get accurate values.

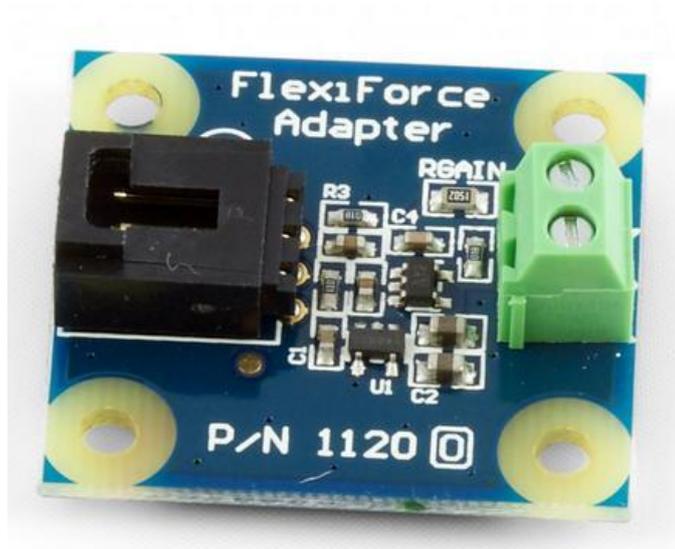


Figure 4.5 – FlexiForce Adapter

In order to calibrate the sensors, a wide range of known weights were applied to the sensing area of each individual FlexiForce sensor and the resulting voltage of each weight was noted, as seen in Table 3.

Table 3 – FlexiForce calibration values

FlexiForce Load 1 (kg)	FlexiForce Voltage 1 (mV)	FlexiForce Load 2 (kg)	FlexiForce Voltage 2 (mV)
0,5	5	1,2	54
1,2	20	2,0	142
1,8	48	3,0	200
2,2	79	3,9	289
3,0	156	4,7	381
5,0	367	6,1	474
6,6	469	7,1	538
8,8	558	8,2	645
9,5	660	9,1	689

A linear interpolation between weights and voltage was done using and the results can be seen in Figure 4.6.

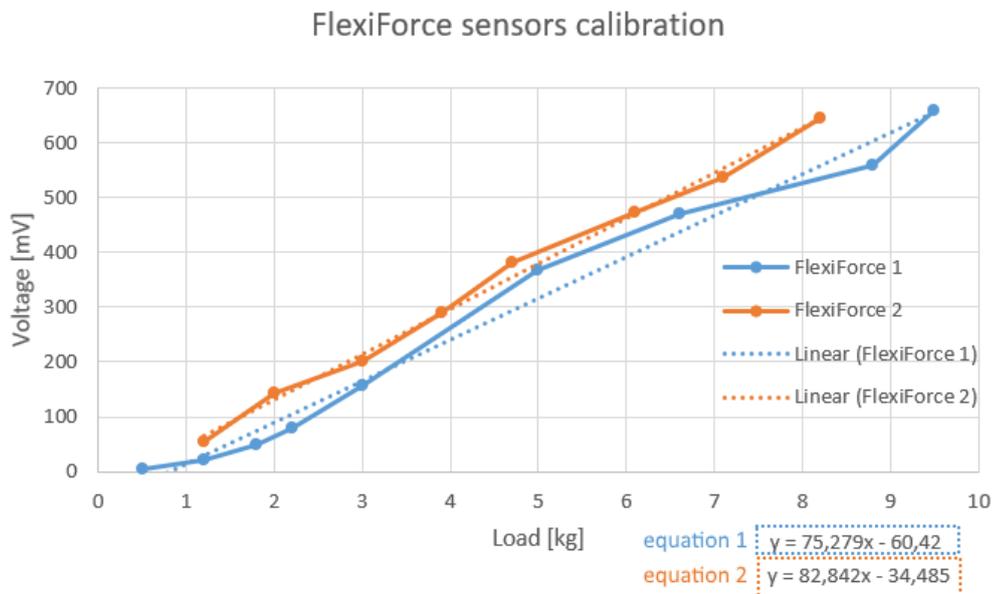


Figure 4.6 – Calibration curves and linear equations for FlexiForce sensors

It is possible to determine a linear equation after applying the linear interpolation using the equation (1):

$$y = mx + b \quad (1)$$

with m and b being provided by the linear interpolation, simply replacing y with the voltage measured from the sensors and solving in order to x will result on the weight applied. To determine force (F) in Newton we simply apply the equation in (2):

$$F = m \times g \quad (2)$$

where m represents the weight measured and g represents the gravitational acceleration.

4.3. Gait Sensor

An Inertial Measurement Unit **AltIMU-10 v4** (see Figure 4.7) [38] was used to track the crutch's movement, therefore tracking patient gait. It contains a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, resulting in a total of 9 degrees of freedom and it operates from 2.5V to 5.5V.



Figure 4.7 – AltIMU-10 v4

This sensor will be used to determine the crutch's orientation and its horizontal and vertical inclination also known as pitch and roll angle, which is shown in Figure 4.8, providing information to the physiotherapist on how the crutch is maneuvered by the patient. Calibration of this sensor is done software-wise, with the assistance of the sensor's Arduino library. Raw values are obtained at first and then transformed into real values, which in this case refers to different angles. The processing of raw values to real values is done taking into consideration the sensor's specifications [38]. Examples of the processing equations used are available in Annex C - Technical Manual, page 11, Figure 1.9.

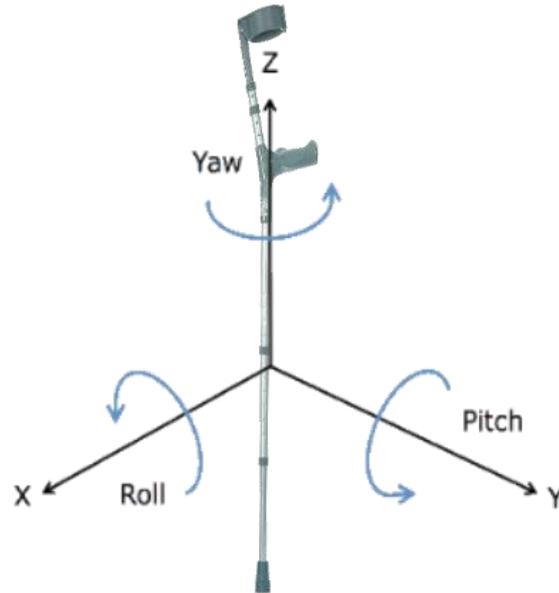


Figure 4.8 – Crutch representing Euler angles

Equation (3) is used to calculate the yaw angle:

$$yaw = atan2(m_y / m_x) \quad (3)$$

The function *atan2* calculates the arctangent in the four quadrants and m_y and m_x indicate the horizontal components of the magnetic field sensed by the magnetometer.

Pitch angle can be determined by (4):

$$pitch = asin(-a_x / g) \quad (4)$$

where *asin* represents the inverse of the *sine* function, a_x represents the acceleration given by the accelerometer in the x-axis and g represents the gravitational acceleration. Finally roll angle can be determined by (5) where a_y and a_z indicate the acceleration in the y-axis and z-axis, respectively.

$$roll = atan2(a_y / a_z) \quad (5)$$

This sensor uses a I²C interface and has several sensitivity values that can be defined by the user. Different values for sensitivity have to be set for each type of sensor. Gyroscope sensitivity can be set at ± 245 , ± 500 or ± 2000 degrees per second, while accelerometer sensitivity values can be set at ± 2 , ± 4 , ± 6 , ± 8 or ± 16 g and finally for the magnetometer values can be set at ± 2 , ± 4 , ± 8 or ± 12 gauss. The lowest sensitivity values were used to provide a better sensing range that fits our project.

The gyroscope can be used to track rotation on a short time accurately, while the magnetometer and the accelerometer can help compensate the gyroscope's drift over time by providing an absolute frame of reference. For accurate reading of the walker's orientation values, sensor fusion will be necessary, so we will be using a Kalman Filter [39]. This filter requires small computational requirements while providing elegant recursive properties. Typical uses of the Kalman Filter include the smoothing of noisy data and providing estimates of parameters of interest, in applications such as GPS, phase locked loops in radio equipment and many more. Despite this filter being over than 50 years old it is still one of the most common and important data fusion algorithms in use nowadays [40].

4.4. Load Cell sensor

Load cell sensors create an electrical signal proportional to the weight applied to the sensor and they are commonly used in the fields of medicine, robotics and industrial, among many others, and they normally use a Wheatstone Bridge configuration [41], which makes it a good choice for measuring weight that the person applies onto the crutch's base. The sensor used, a stainless steel **SNC2C6 Load Cell** (see Figure 4.9) can measure weights up to 50kg. Some specifications regarding this sensor can be seen in Table 4. A simple comparison regarding Table 4 and Table 2, comparing the load cell sensor and the FSR sensor specifications, it is clear that the load cell sensor is more accurate in every aspect, but it lacks flexibility; each sensor has their own set of applications.

Table 4 – SNC2C6 load cell sensor performance specifications [42]

Non-linearity	$\pm 0.3\%$ of Full Scale
Repeatability	$\pm 0.3\%$ of Full Scale
Hysteresis	$\pm 0.3\%$ of Full Scale
Precision	$\pm 0.1\%$ of Full Scale
Sensitivity	2.0 ± 0.1 mv/V
Operating Temperature	-20°C to 70°C



Figure 4.9 – SNC2C6 Load Cell

When tested, this sensor provided very little output by itself when powered at 5V, only reaching a maximum of approximately 10mV. It was clear that a signal conditioner was needed, so by coupling it with an instrumentation amplifier we were able to get a good measurement range. The INA122 is a precision instrumentation amplifier for accurate, low noise signal acquisition, which makes it ideal for this case.

The conditioning circuit needed, connecting the load cell based on a Wheatstone Bridge configuration, to the INA122 amplifier can be seen in Figure 4.10.

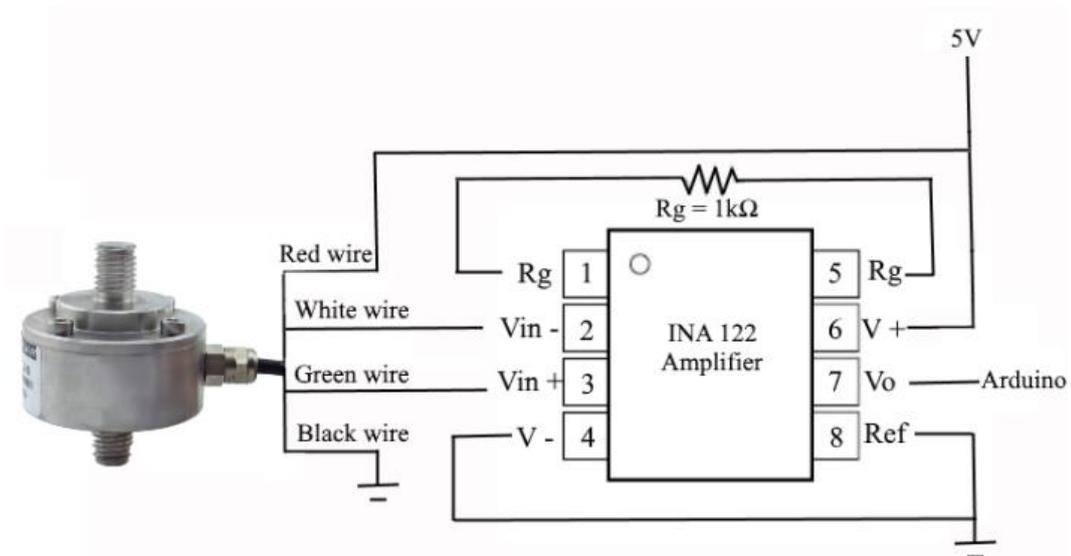


Figure 4.10 – Load cell conditioning circuit

With the value of $R_g=1k\Omega$ the amplifier can provide a gain of $G=205$ according to (6), which is enough to have a good range of measures for calibration purposes.

$$G = 5 + \frac{200k\Omega}{R_g} \quad (6)$$

Calibration is also needed in this sensor, with Table 5 providing the calibration values which resulted in a calibration curve seen in Figure 4.11.

Table 5 – Load cell calibration values

Load (kg)	Voltage (mV)
2,3	65
5	145
7	219
10	312
15	450
20	576
25	762
32,5	985

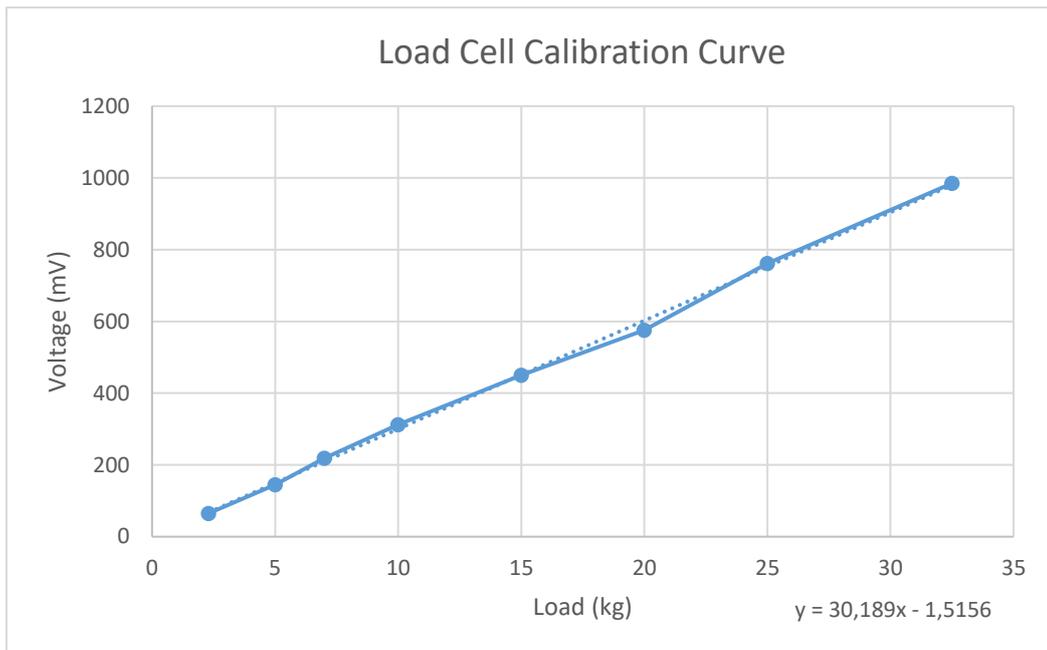


Figure 4.11 – Load cell calibration curve

As expected, the calibration values that originate the calibration curve are more linear than the FSR calibration values seen in Figure 4.6.

4.5. Microcontroller

Microcontrollers are simply a small computer on a single integrated circuit, containing one or more CPU's, memory, programmable inputs and outputs, among others. They are commonly used in automobile engine control systems, medical devices, remote controls, power tools and many more embedded systems.

- A **Raspberry Pi Zero** (see Figure 4.12) [43] was initially considered due to its low cost, small size and the fact that it has a GPU incorporated, plus USB and ethernet connectors. However, its GPIO pins are purely digital and an ADC circuit would be needed to convert the signals since the force and load cell sensors provide an analog signal. Development in this board is usually done in Python as its main programming language with the default firmware being closed source.



Figure 4.12 – Raspberry Pi Zero

- A similar board, an **Arduino Nano** (see Figure 4.13) [44] containing an ATmega328 microcontroller with the ability to read and process analog signals was studied. This board also contains a USB connector and has a I²C serial bus, which is needed to read values from the IMU. Unlike the Raspberry Pi Zero, it has no graphic supports but instead has an open source software application, Arduino IDE. This software application uses C/C++ languages and because of this and the other reasons stated above, we decided to use an Arduino Nano for this project.



Figure 4.13 – Arduino Nano

Some comparisons between the two boards can be seen below in Table 6.

Table 6 – Raspberry Pi Zero and Arduino Nano specifications

	Raspberry Pi Zero	Arduino Nano
Architecture	ARM v6Z	AVR
Pins	17 GPIO (all digital)	14 digital, 8 analog
Operating Voltage	5V	5V
GPU	✓	✗
Size	65mm x 30mm x 5mm	45mm x 18mm x 4mm
Weight	9g	7g

4.6. Bluetooth Adapter

- Since the Arduino Nano doesn't have a Bluetooth interface, a Bluetooth adapter is needed for the sensor data to be transmitted to the mobile device. The first adapter studied was the **Adafruit Bluefruit LE SPI friend** (see Figure 4.14) [45], that promoted the use of BLE connectivity. This adapter uses a common four pin SPI interface (MISO, MOSI, SCK and CS) plus a fifth GPIO pin dedicated to interrupts. With the SPI interface there is no need to worry about baud rates, flow control or giving up a hardware UART port.

Attempts were made to send sensor data to the mobile devices, but they were proven unsuccessful due to some difficulties in coding the Bluetooth communication of data.



Figure 4.14 – Bluetooth SPI friend

- An alternative was found to the latest adapter, **Bluetooth module HC-06** (see Figure 4.15) [46] which has a standard 2 pin UART connection (T_x and R_x) making it easier to connect it to the Arduino. Coding the data communication, unlike the previous adapter, is quite simple so we decided to use this module. Another option was the HC-05 module, which works as both master and slave, meaning the module can connect to other Bluetooth devices and can also receive connections. The difference to our module, the HC-06, is that it only works as a slave, meaning it can only receive connections from other Bluetooth devices, and not start them. This is exactly what was intended so this module fits our project perfectly.



Figure 4.15 – Bluetooth module HC-06

This device can be programmed to change certain aspects of it, such as baud rate, name displayed, and password needed to pair. This can be done using the following schematic (see Figure 4.16), followed by some code available in Annex B – User Manual, Page 5, Topic 1.1. Several tests prove the considered Bluetooth communication and chosen transceiver provides optimal communication coverage taking into account the distance between the smart crutches and the smart phone or tablet of the physiotherapist during the training session. Bluetooth communication should cover a maximum area of 10 meters, which should be more than enough for this project, considering that the physiotherapist needs to be relatively close to the patient while the session is in progress.

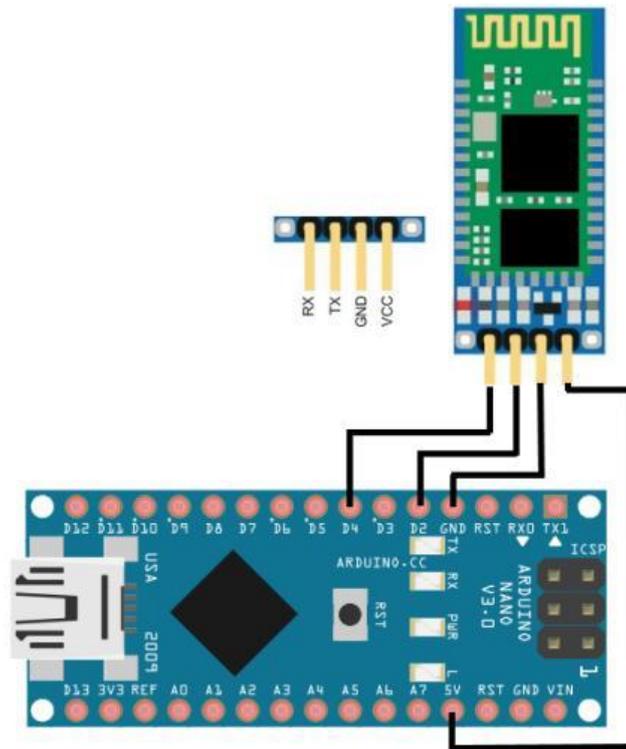


Figure 4.16 – Schematic for Bluetooth HC-06 module configuration

4.7. PCB Board

This PCB board (see Figure 4.17) supports and connects electrical components using conductive tracks from two sheet layers of copper, making it a double-sided layer. The main purpose of this board is to exponentially decrease the use of wires needed in the project. Figure 4.18 indicates where each component should be plugged in.

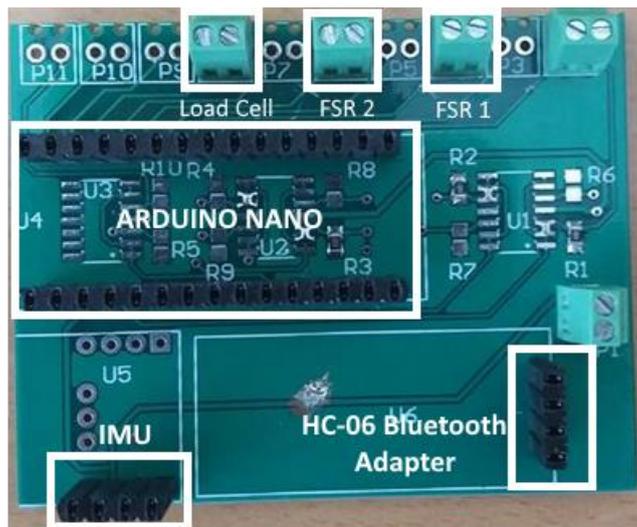


Figure 4.17 – PCB board

A schematic regarding the board connections can be seen in Figure 4.18. Some pins of the Arduino are connected directly in the board to the several existing components seen in the figure. A circuit containing all of the SmartCrutch system’s connections will be showed later in this chapter.

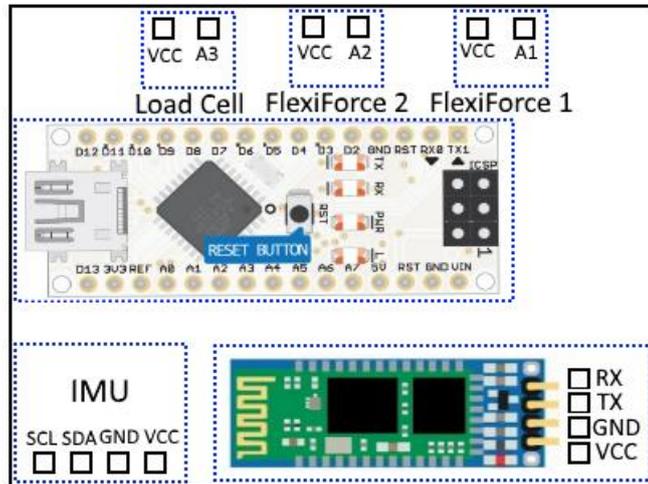


Figure 4.18 – PCB board connections

4.8. Battery

Since the Arduino Nano operating voltage is 5V, a battery is needed to make sure the system is autonomous. Power banks (see Figure 4.19) provide the output voltage necessary for the Arduino to function properly while also making it easy to charge them, simply by connecting a USB cable to a computer. The power bank used provided a capacity of 2200mAh.



Figure 4.19 – Power bank

4.9. Mobile devices

Mobile devices are computing devices small enough to hold and to be operated by people's hands. In the mid late 90's, mobile phones were only used for basic purposes, like calling and texting someone. It was only until the mid-2000's that the first smartphones were introduced, based on Microsoft's Windows Mobile and then BlackBerry smartphones. These phones usually had a QWERTY keyboard that disappeared with time, being replaced by an LCD touchscreen which made smartphones much more enjoyable. The first Android operating system, called HTC Dream, was released in October 2008. Android adaption as relatively slow at first but it started to be popular in 2012 and now dominates the market share of smartphones [47].

Tablet devices follow the same idea as modern smartphones with the only difference being a bigger screen size.

In order to test the application, a smartphone **Samsung Galaxy J5 (2016)** (see Figure 4.20) and a **Samsung Galaxy Tab S2** (see Figure 4.21) were used, making it possible for the physiotherapist to use the application on a phone or tablet.



Figure 4.20 – Samsung Galaxy J5



Figure 4.21 – Samsung Galaxy Tab S2

Table 7 shows the different specifications of both devices. Besides providing a bigger visual interface, the Samsung Galaxy Tab S2 specifications are better than the Samsung Galaxy J5. An important factor is that both devices have Bluetooth and WiFi capabilities. The application developed was tested on both devices and there were no issues with performance.

Table 7 – Samsung Galaxy J5 and Samsung Galaxy Tab S2 specifications [48-49]

	Samsung Galaxy J5	Samsung Galaxy Tab S2
Resolution	720 x 1208 pixels	1536 x 2048 pixels
Pixel Density	282 ppi	320 ppi
Processor	Quad-core, 1200 MHz, ARM Cortex-A53, 64-bit, 28 nm	Octa-core, 1900 MHz, ARM Cortex-A57 and ARM Cortex-A53, 64-bit
GPU	Adreno 306	Mali-T760 MP6
RAM	2 GB	3 GB
Storage (internal)	16 GB	64 GB
Capacity	3100 mAh	4000 mAh
Bluetooth	✓	✓
WiFi	✓	✓

4.10. Electrical Circuit

The final circuit used in this project can be seen in Figure 4.22. All the components pins are connected to the respective Arduino pins, through the help of the PCB board.

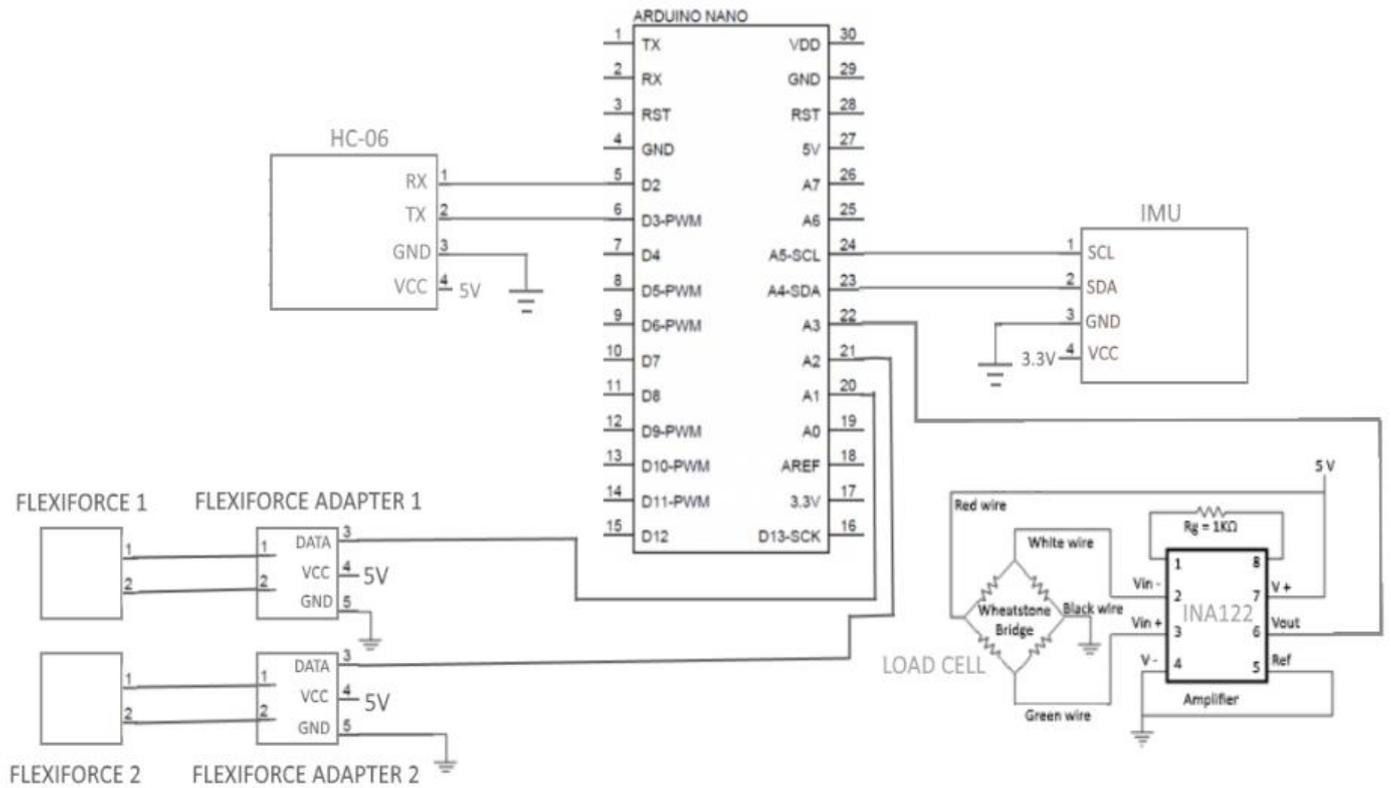


Figure 4.22 – SmartCrutch system electrical circuit

4.11. Prototype

After all the connections were made, a prototype was developed (see Figure 4.23, 4.24 and 4.25) containing all the devices listed in this chapter.

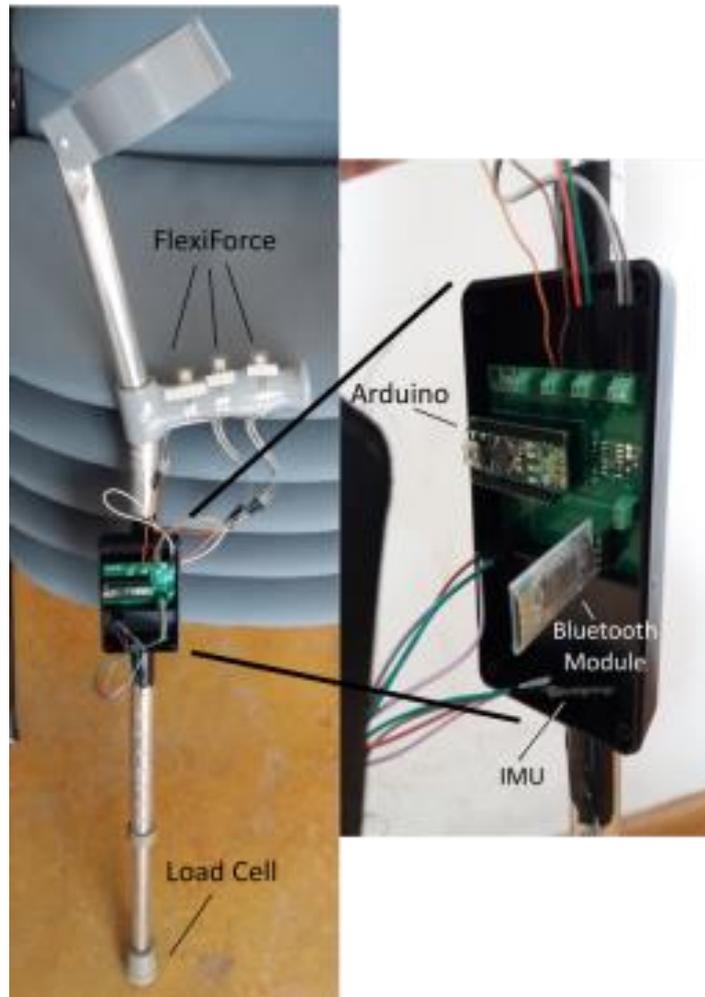


Figure 4.23 – SmartCrutch initial prototype

A few devices attached to the PCB board can be seen in Figure 4.23, as well as the FlexiForce sensors and the load cell sensor location. This figure indicates the prototype in its early development. A finished prototype can be seen below in Figure 4.24, 4.25 and 4.26. The first one shows the components that are held inside the small plastic box, being held by non-conductive duct tape. The IMU must be at a perpendicular position to the ground, which means the box should also be in that same position in order to get accurate values.

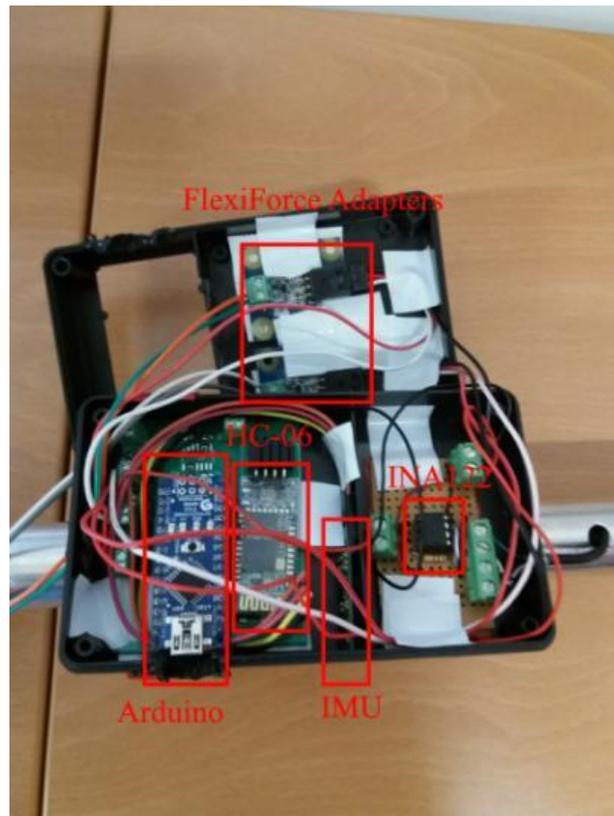


Figure 4.24 – SmartCrutch box components



Figure 4.25 – SmartCrutch final prototype



Figure 4.26 – SmartCrutch final prototype 2

Chapter 5 – Software

For the hardware listed in the previous chapter to function properly in our system, specific software must be implemented. This chapter focuses on discussing the following topics:

- Arduino IDE;
- Android Studio;
- Database;
- Server;
- PHP scripts.

Every component attached to the Arduino will be programmed using the Arduino IDE software, through the use of C/C++. Development of the mobile application will be done using Android Studio, an open source software with the sole purpose of developing Android applications in JAVA and XML. A local and remote database will be created, storing all the vital information needed, such as patient history and session data. Finally, PHP scripts will be developed with the purpose of serving as a middle man for data synchronization between mobile devices using the Internet.

5.1. Arduino IDE

Arduino IDE is an open source software dedicated for development in Arduino boards, such as Arduino Uno, Arduino Mega and in our case, Arduino Nano. This is a cross platform application, meaning it can be developed in different OS's, such as Windows, Linux and macOS. The point of this software is to upload a program into the Arduino's microcontroller's flash memory, with the possibility of changing said code by uploading it again if necessary. It also contains a core library with basic methods, such as setting pin modes to high or low, and reading digital and analog pins for instance. This is also known as firmware.

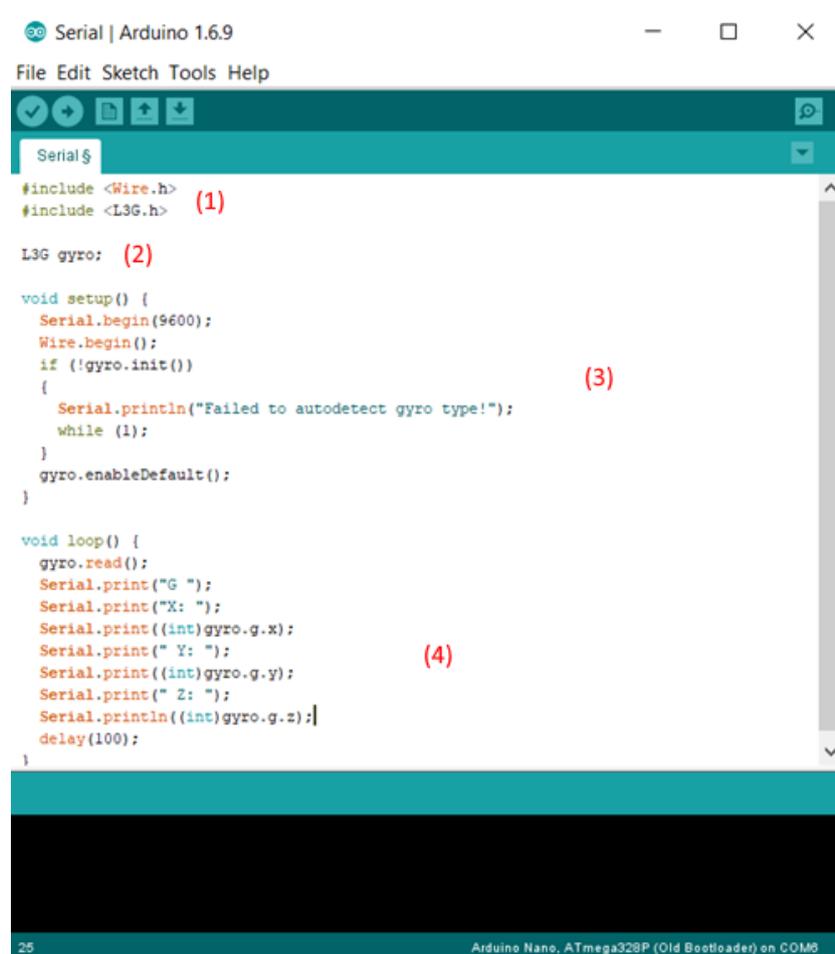
In order to develop an application, a sketch must be created first, containing two mandatory methods:

- `setup()` – only runs once when the board is powered, or the reset button is pressed;
- `loop()` – runs the method continuously without stopping.

The first function, `setup()`, is called when a sketch starts and it is used to initialize variables, pin modes and libraries.

The `loop()` function on the other hand, does exactly what the name suggests, looping consecutively every few seconds, defined by a `delay(time)` function, with time being in milliseconds. Values from the sensors will be actively retrieved given a delay of time in our case.

This program also has access to libraries which are completely free, because it is an open source software, and they can be easily imported to our sketch to provide us with extra functionalities. After correctly importing the libraries an `#include` statement must be inserted at the top of the sketch (before the `setup()` method) for each library. Like any library in programming, the public functions and constants defined by it will be available for us to use them in our sketch. An example of a sketch can be seen in Figure 5.1.



```
Serial | Arduino 1.6.9
File Edit Sketch Tools Help
Serial $
#include <Wire.h>
#include <L3G.h> (1)

L3G gyro: (2)

void setup() {
  Serial.begin(9600);
  Wire.begin();
  if (!gyro.init()) (3)
  {
    Serial.println("Failed to autodetect gyro type!");
    while (1);
  }
  gyro.enableDefault();
}

void loop() {
  gyro.read();
  Serial.print("G ");
  Serial.print("X: ");
  Serial.print((int)gyro.g.x);
  Serial.print(" Y: "); (4)
  Serial.print((int)gyro.g.y);
  Serial.print(" Z: ");
  Serial.println((int)gyro.g.z);
  delay(100);
}
```

Figure 5.1– Arduino IDE sketch example

Figure 5.1 shows an example of a library sketch used to determine the raw gyroscope values of the IMU. In (1), include statements are made for the purpose of importing the

libraries needed for the sketch. The first library, '*Wire.h*' is a library developed with the purpose of allowing communication with I²C devices, such as the IMU used. The second library, '*L3G.h*' is a library used for reading raw values of specific gyroscopes, such as the one included in the IMU, the L3GD20H. In (2) an object of the library is defined and given a name, '*gyro*' for later use. The `setup()` method seen in (3) shows the initialization of variables and libraries used. Serial communication is used between the Arduino board and the computer, at a baud rate of 9600. The baud rate is the rate of which information is transferred in a communication channel, meaning that the serial port of the Arduino is capable of transferring a maximum of 9600 bits per second. I²C communication is then started, and if the gyroscope is not detected, an error is displayed to the user. If not, the gyroscope is enabled and finishes the `setup()` method.

If the `setup()` method was successful, a loop method (4) starts by reading the value obtained by the gyroscope sensor and printing out several things, like the raw gyroscope values of x, y and z. This program will loop indefinitely every 100 milliseconds due to the `delay()` function present at the end of the sketch.

Since this board supports multiple Arduinos, and some Arduinos have different microcontrollers, before uploading a program to the microcontroller, it is necessary that the right devices and microcontrollers are chosen. After that, uploading the program to the Arduino is simply done by connecting a USB cable to the computer, choosing the right COM port and pressing the upload button, as seen in Figure 5.2.

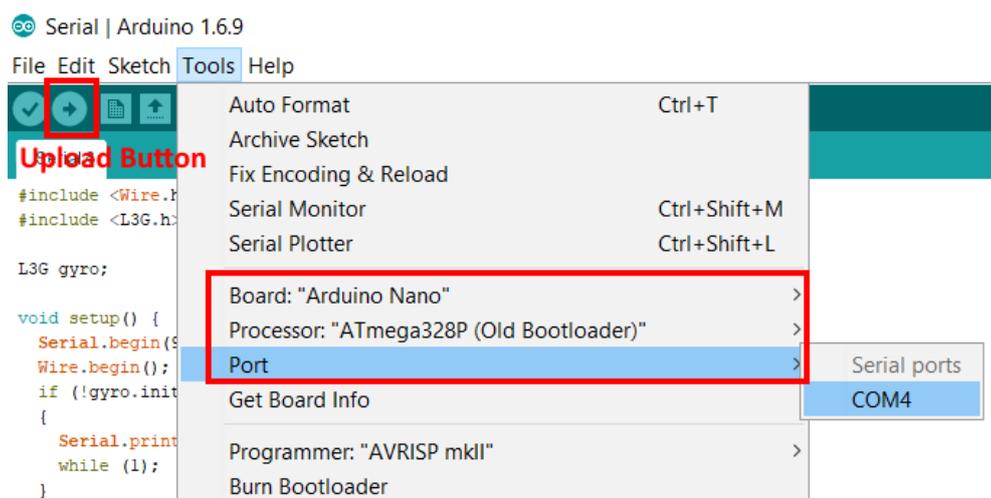


Figure 5.2– Program uploading to Arduino boards

In this project, sensor values are read and sent to the mobile device every 200ms providing a good flow of data. A value too big would not capture relevant data, while

a value too small would overload the database. Therefore, each sensor will store its value in the local database five times per second or 300 times per minute, which provides no issues to the database itself in terms of holding all the data.

5.2. Android Studio

Android Studio is the official IDE for Google’s Android OS designed specifically for Android development. This software, like Arduino IDE, can be installed in Windows, Linux or macOS systems and it is also open source. This software provides a rich UI development environment with templates to give new developers a launching pad into Android development. A free and extensive user guide [50] is available for an introduction to basic Android Studio features.

Google provides information about data regarding the number of devices running a given version of the Android platform [51] (see Figure 5.3).

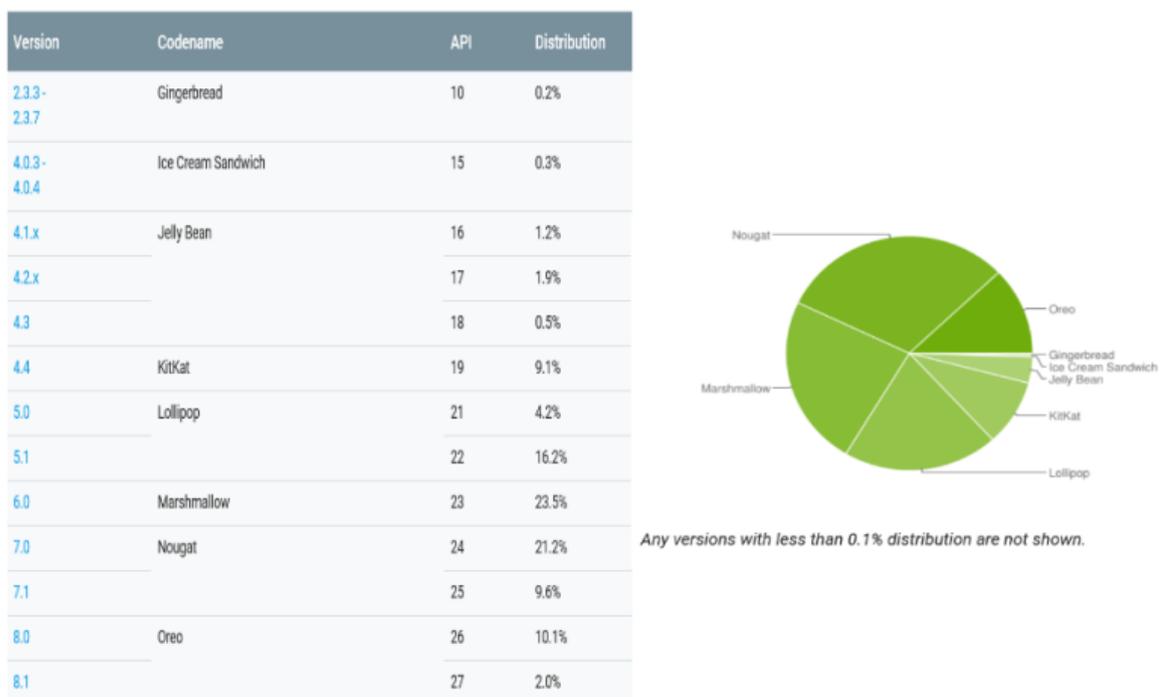


Figure 5.3 – Data on the relative number of devices with a certain version of Android platform collected during a 7-day period ending on July 23, 2018

With the above data in mind, and with the purpose of reaching the maximum possible number of versions, the proposed application was developed including a range from Jelly Bean version (API 16) up to a target version of Oreo version (API 27), reaching a total of 99,5% of Android devices. To synchronize data to the server, access to the internet is required using a mobile network (3G, 4G) or WiFi, and in order to start a session the Android device should have Bluetooth capabilities.

This software main's structure can be divided in four important modules:

- Manifest;
- Activities (JAVA classes);
- Layouts (XML files);
- Gradle.

The application was developed with Android Studio version 3.1.4 and in Figure 5.4 a screenshot of the IDE is shown with the relevant modules seen before.

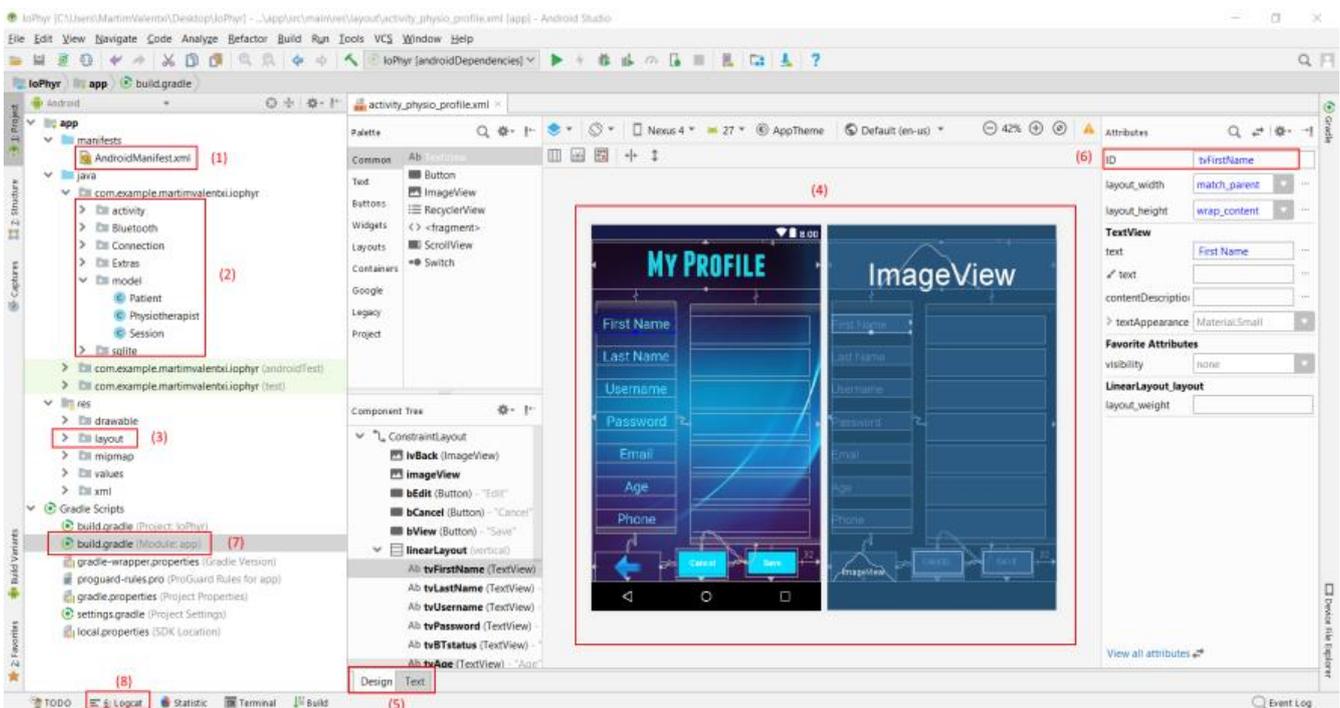


Figure 5.4 – Android Studio IDE

Each application is required to have a single manifest XML file (1) where many important settings regarding the application's functionality will be applied, such as permissions (accessing Internet, camera, Bluetooth, writing to external storage), the icon and label displayed to run the APP, and all the activities used. If an activity is developed but is not included in the manifest file, the application will crash when said activity tries to run.

All activities are programmed in JAVA and can be considered the back-end programming of Android Studio (2). These JAVA classes will contain methods and algorithms to provide the necessary results to the user. Activities by themselves don't

display anything to the user, as any back-end application, so they need to be coupled with a front-end layer. This layer is called layouts (3) in Android Studio and they can be programmed in XML or by drag and dropping components seen in Figure 5.4, such as buttons, text views, image views, and so on. The *res* folder, which stands for resources, besides holding all the layouts, also holds images, string values and color values.

A virtual layout is available (4) for component positioning, and these can be adjusted later by switching from design to text (5), which is where XML code is used. Many virtual emulators are available for download for testing purposes. Every component must have an ID (6) which will be initialized in the respective JAVA class.

Gradle (7) is an advanced build toolkit that automates and manages the build process, allowing flexible build customizations. Relevant information in this file includes minimum and target Android version allowed and libraries. Figure 5.5 shows the build process of an Android application into an executable APK. This software also allows debugging through the use of Logcat (8). If, for instance, an activity tried to run but wasn't declared on the manifest, the application would crash and the information regarding the crash would appear in the Logcat.

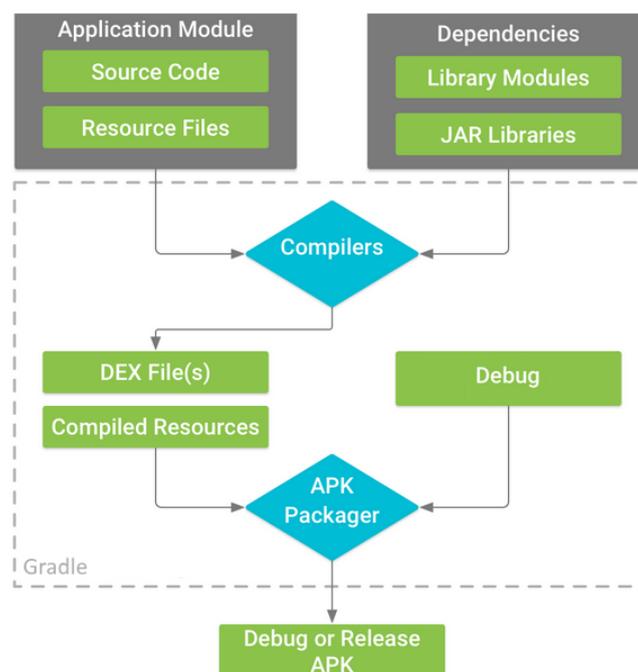


Figure 5.5 – Build process of an Android Application

Steps for the build of an Android application module are the following:

1. Source code is converted into DEX files by the compilers, which include the bytecode that runs on Android devices. Everything else is compiled into resources.
2. The APK packager combines the DEX files and the compiled resources into a single APK. Before the application can be installed and deployed into an Android device, the APK must be signed.
3. In our case, the APK manager will sign the APK with a debug keystore, meaning the application is intended only for testing.
4. Finally, a final APK is generated and made available for testing and usage.

Two layouts had to be developed for each activity because of the different screen sizes, one for a phone and another one for a tablet (see Figure 5.6). All the phone layouts are locked to portrait mode due to its small size, whereas in tablets most layouts are also in portrait mode, with the exception of the ones that provide session data to the user, such as in *activity_bluetooth*, which is the class used for starting a session and viewing live data. The layouts that provide data to the user are locked to landscape mode for a wider view of the data.

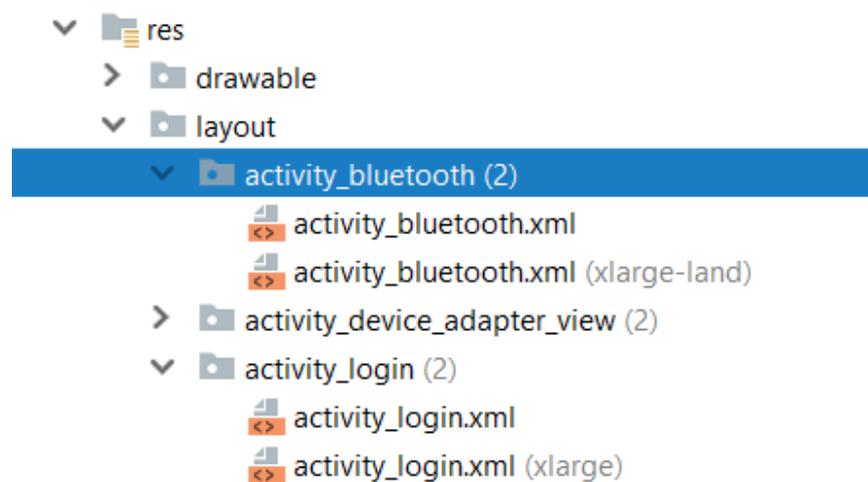


Figure 5.6 – Layouts of different activities

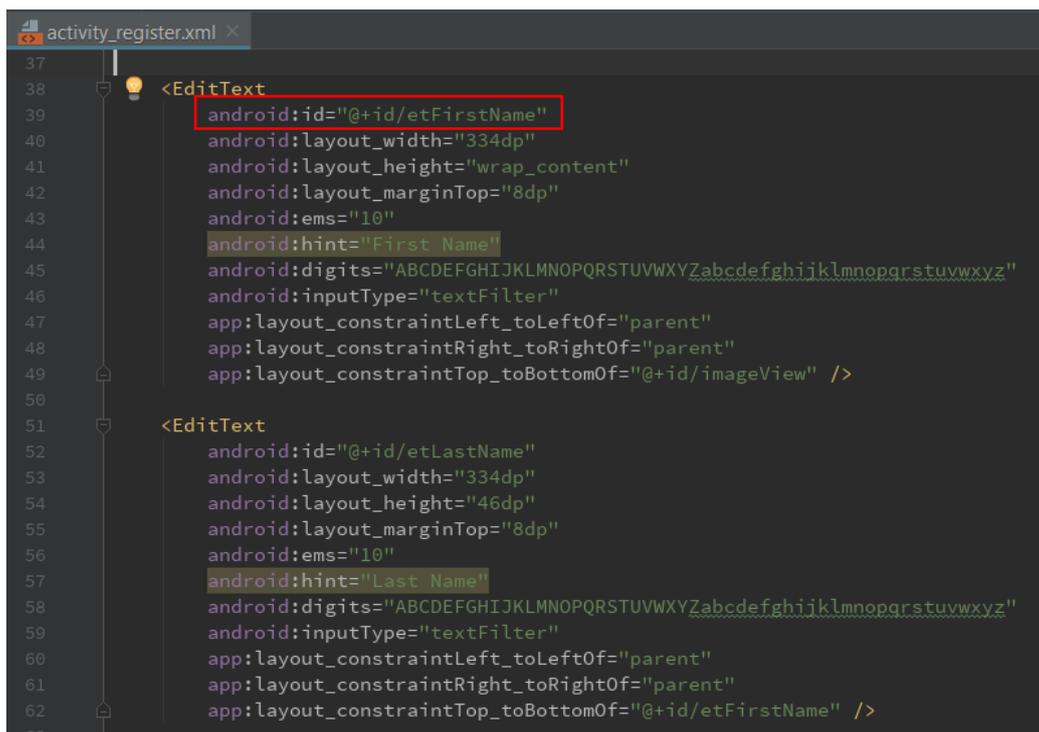
Connection between JAVA classes and the corresponding XML layout is explained below with the help of Figure 5.7 and Figure 5.8.

```
PhysioRegisterActivity.java x
1 package com.example.martimvalentxi.iophyr.activity;
2
3 import ...
20
21 public class PhysioRegisterActivity extends AppCompatActivity { (1)
22     SQLiteOpenHelper openHelper;
23     SQLiteDatabase db;
24
25     EditText etFirstName,etLastName,etUsername,etPassword,etEmail,etAge,etPhone;
26     Button bRegister,bCancel;
27     AlertDialog alert;
28
29
30 @Override (2)
31 protected void onCreate(Bundle savedInstanceState) {
32     super.onCreate(savedInstanceState);
33     setContentView(R.layout.activity_register); (3)
34
35     openHelper = new DatabaseHelper( context: this);
36
37     alert = new AlertDialog();
38
39     etFirstName = (EditText) findViewById(R.id.etFirstName); (4)
40     etLastName = (EditText) findViewById(R.id.etLastName);
41     etUsername = (EditText) findViewById(R.id.etUsername);
42     etPassword = (EditText) findViewById(R.id.etPassword);
43     etEmail = (EditText) findViewById(R.id.etEmail);
44     etAge = (EditText) findViewById(R.id.etAge);
45     etPhone = (EditText) findViewById(R.id.etPhone);
46     bRegister = (Button) findViewById(R.id.bView);
47     bCancel = (Button) findViewById(R.id.bCancel);
48     bCancel.setOnClickListener((v) -> { finish(); });
54 }
```

Figure 5.7 – JAVA class (physiotherapist register activity)

In order to define a class as an Activity it is necessary to extend an Activity, or in this case, an *AppCompatActivity* (1). This last one is chosen because the regular Activity class is static, the code was written once and never updated. The *AppCompatActivity* gets improvements periodically so it is always going to be better and faster, while providing more freedom of usage to the user. Each activity must override an *onCreate* method (2).

This method is going to run when the activity is started, similar to the *setup()* method in the Arduino IDE. This is where variables should be declared. An important function, *setContentView*, must be called to associate the activity class with the corresponding XML file (3). In this case, class *PhysioRegisterActivity* is associated with the XML file *activity_register*. Declaring a variable can be done with the *findViewById* function where the user must associate a variable created in the JAVA to a component's ID created in the XML file (4). Figure 5.8 shows an example of the corresponding XML file, *activity_register*.



```
37
38 <EditText
39     android:id="@+id/etFirstName"
40     android:layout_width="334dp"
41     android:layout_height="wrap_content"
42     android:layout_marginTop="8dp"
43     android:ems="10"
44     android:hint="First Name"
45     android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
46     android:inputType="textFilter"
47     app:layout_constraintLeft_toLeftOf="parent"
48     app:layout_constraintRight_toRightOf="parent"
49     app:layout_constraintTop_toBottomOf="@+id/imageView" />
50
51 <EditText
52     android:id="@+id/etLastName"
53     android:layout_width="334dp"
54     android:layout_height="46dp"
55     android:layout_marginTop="8dp"
56     android:ems="10"
57     android:hint="Last Name"
58     android:digits="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
59     android:inputType="textFilter"
60     app:layout_constraintLeft_toLeftOf="parent"
61     app:layout_constraintRight_toRightOf="parent"
62     app:layout_constraintTop_toBottomOf="@+id/etFirstName" />
```

Figure 5.8 – XML file (physiotherapist register layout)

In Figure 5.8 different resources can be found in each component created in the layout. Two *EditText* components are visible, and each should have a different ID for successfully corresponding to the objects created in the JAVA class. Several attributes are added, such as hint, allowed digits, input type and size of the component.

5.3. Database and Server

All the information created and generated by the application regarding physiotherapist information, patient information and patient's session data will be stored in databases, which makes it one of the main components of the SmartCrutch system.

5.3.1. Local Database - SQLite

To avoid being connected to the Internet or WiFi all the time, the application was developed mainly as offline. What this means is that the application itself must have a local database to store the patient information and data received from the SmartCrutch. Android Studio provides the user with the possibility of creating SQLite databases [51], which is based on on regular SQL databases. This type of database is commonly used in handling structured data where there are relations between different entities of the data. It consists of data query, data manipulation (create, read, update, delete, also known as CRUD), data structure and data access control.

Databases consist mainly of tables and variables, which need to be created by code in Android Studio. A diagram should be drawn on paper for an idea of the database architecture and then translated into code, using functions such as the one seen in Figure 5.9.

```
db.execSQL("CREATE TABLE " + TABLE_PHYSIO + "(ID TEXT PRIMARY KEY, FirstName TEXT, LastName TEXT,  
db.execSQL("CREATE TABLE " + TABLE_PATIENT + "(IDPatient TEXT PRIMARY KEY, FirstNamePatient TEXT,  
db.execSQL("CREATE TABLE " + TABLE_SESSION + "(IDSession TEXT PRIMARY KEY, IDPatientSession TEXT,
```

Figure 5.9 – SQLite functions

In this case, three tables will be created with different names, *TABLE_PHYSIO*, *TABLE_PATIENT*, *TABLE_SESSION*, all of them with different column names, such as specific ID's, first name, last name and so on. Each table should also have a primary key which is designated to uniquely identify all table records. Because of this, a random UUID is created and stored as an ID whenever a physiotherapist, patient or session is created, making sure that the ID is unique. Since this is an offline based application, if synchronization wasn't being done constantly, ID's would overwrite each other, rendering the database useless.

However, Android Studio does not provide a visual interpretation of the SQLite database, so a library for debugging SQLite databases, Android Debug Database [52] was used. This library provides the user with the possibility to view all the created databases in the application, view tables, edit row information and run SQL query's, among others.

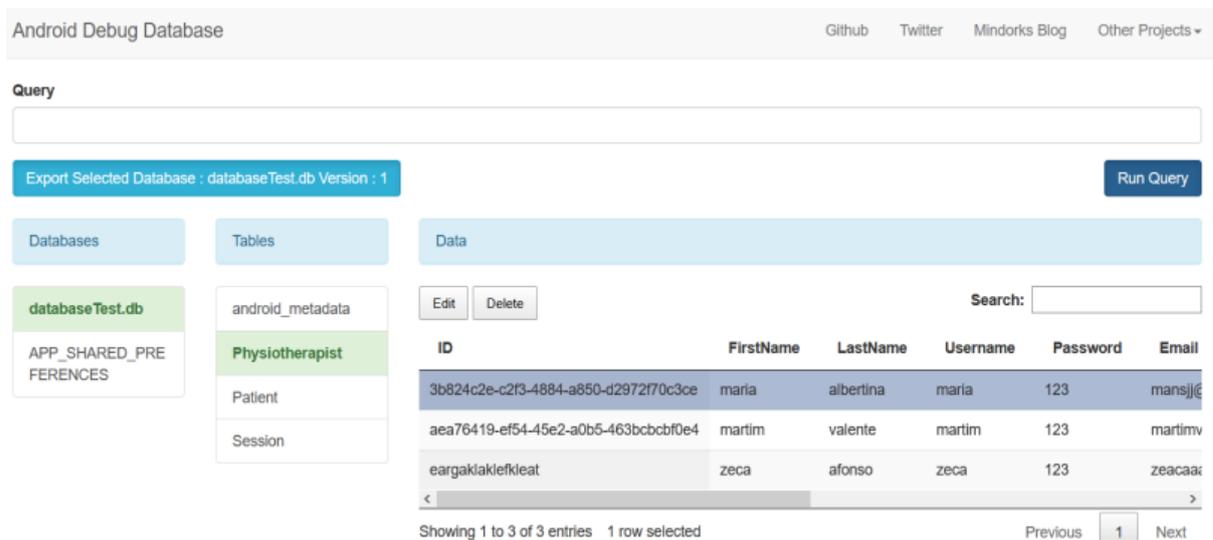


Figure 5.10 – Android Debug Database example

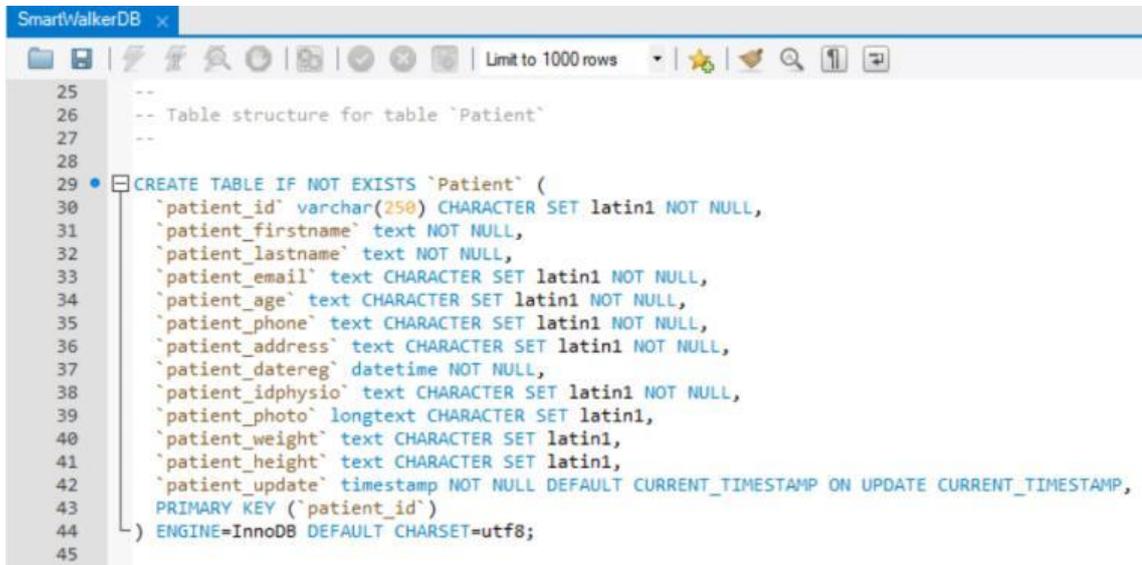
5.3.2. Remote Database - MySQL

For synchronization purposes, a remote database must store all data in case the physiotherapist switches his mobile device, losing all the local data stored. Unlike

ISCTE-IUL

Smart Object for Physical Rehabilitation Assessment

SQLite, MySQL databases provide a visual interpretation on how the architecture of the database is going to look like, with the help of MySQL Workbench [53] software. All the tables and rows are filled, allowing the database to be exported to a single file. Queries can also be created instead of manually filling out the tables. An example is shown in Figure 5.11 for the creation of the Patient table.

The image shows a screenshot of the MySQL Workbench interface. The title bar reads 'SmartWalkerDB'. The main window displays a SQL query for creating a table named 'Patient'. The query is as follows:

```
25 --
26 -- Table structure for table `Patient`
27 --
28
29 CREATE TABLE IF NOT EXISTS `Patient` (
30   `patient_id` varchar(250) CHARACTER SET latin1 NOT NULL,
31   `patient_firstname` text NOT NULL,
32   `patient_lastname` text NOT NULL,
33   `patient_email` text CHARACTER SET latin1 NOT NULL,
34   `patient_age` text CHARACTER SET latin1 NOT NULL,
35   `patient_phone` text CHARACTER SET latin1 NOT NULL,
36   `patient_address` text CHARACTER SET latin1 NOT NULL,
37   `patient_datereg` datetime NOT NULL,
38   `patient_idphysio` text CHARACTER SET latin1 NOT NULL,
39   `patient_photo` longtext CHARACTER SET latin1,
40   `patient_weight` text CHARACTER SET latin1,
41   `patient_height` text CHARACTER SET latin1,
42   `patient_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
43   PRIMARY KEY (`patient_id`)
44 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
45
```

Figure 5.11 – Example of query for patient SQL table

In Figure 5.12 a diagram of the final database can be observed.

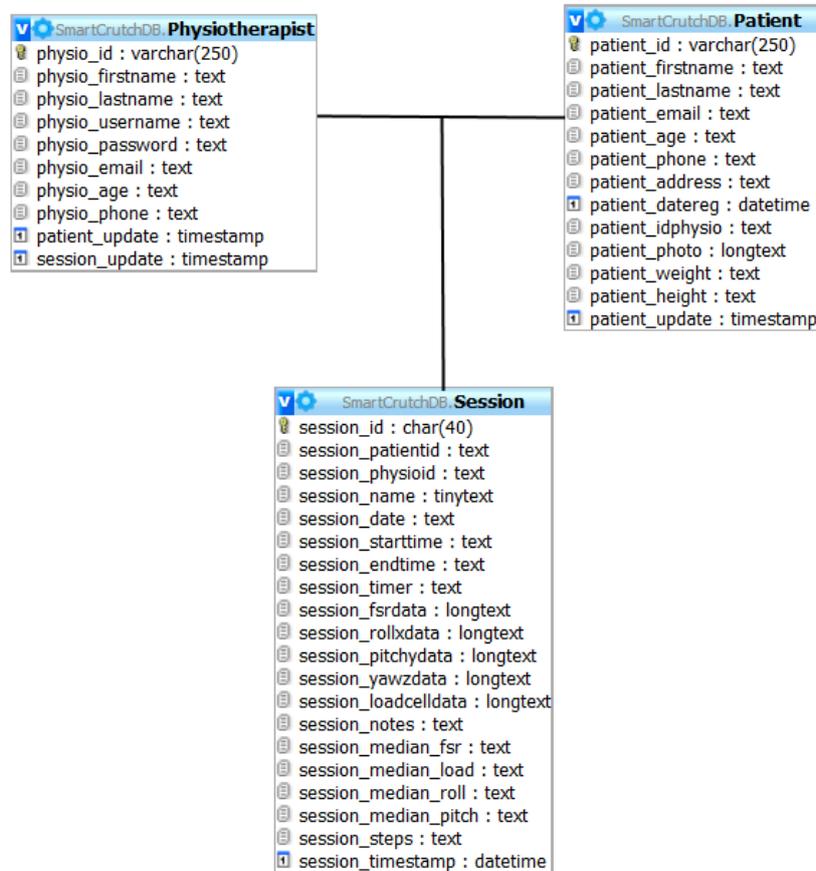


Figure 5.12 – Database diagram

- **Physiotherapist table:** stores the data related to the physiotherapists of the clinic. Each physiotherapist should have their own login information, such as username and password, which are present in this table. Every time a physiotherapist is created, a UUID is generated, making it the primary key of the table. Relevant information, such as first name, last name, email, age and phone are also present in this table. For synchronizing purposes, timestamps are created to update every time a patient or session is added or edited. Each physiotherapist can have multiple patients associated to him.
- **Patient table:** stores the data for patients created in the clinic. Each patient, like the physiotherapists, should have a UUID as primary key. Relevant personal information, such as first name, last name, email, age, address, phone, weight and height can all be edited. This table also stores the date when the patient was created, which cannot be changed. A photo of the patient can be taken at any time, encoding and decoding it in

Base64 [54], which is visible to the physiotherapist, working as a type of profile folder. For synchronizing purposes, another timestamp is updated every time the patient information is changed. Each patient can only be associated to one physiotherapist, meaning the physiotherapist's ID will be stored in the patient's table.

- **Session table:** stores the data for each session created in the clinic. Like the patient and the physiotherapist, the primary key is a random generated UUID when a session is started. It also stores information like the name of the session, date when the session was created and time of start and ending. Notes are also stored after a session is completed. All the data from the sensors (FSR, load cell, IMU) is stored in individual strings, split by ';'. This information, with the conjunction of a timer variable stored in minutes, will be vital for the creation of time charts regarding sensor data visualization. Median values and steps taken are also stored in this table. For synchronizing purposes, a timestamp is created every time a session starts. Each session can only be associated with a single physiotherapist and a single patient, therefore storing both ID's in its table.

5.4. Server

The MySQL database, after being created and exported by MySQL Workbench, can be imported into the server using phpMyAdmin tool [55]. This tool allows the administration of MySQL databases over the Internet. However, remote databases must be stored in a server, unlike the SQLite database, which can be stored in the application itself. The server is responsible for storing all the important information regarding clinic physiotherapists, patients and sessions, making sure the information is centralized in one place and can be accessed remotely by the users, in this case, the physiotherapists.

In order to exchange information between the mobile application and the server's database, REST Web Service [56] is used, making the server responsible for accepting HTTP requests from clients and providing them with HTTP responses. The Apache server is one of the most popular servers used in the world, and like all the software used in this system, it is free and open source.

Accessing the database located on the server is done through the use of PHP, which is a server-side scripting language. An already existing architecture, XAMPP [57] (see Figure 5.13) was used, which combines the open source software listed above in one single software, making it one of the most common configurations for web servers.



Figure 5.13 – XAMPP software

The term XAMPP stands for:

- **X** – Cross Platform;
- **A**pache – HTTP Web Server software;
- **M**ariaDB – Database type (highly compatible with MySQL);
- **P**HP – Programming language;
- **P**erl – Programming language (not used).

Accessing the server for file allocation, such as PHP scripts, was done using another free software, FileZilla client, which is used for file transferring using FTP or, in our case, SFTP protocols. The server, provided by the dissertation supervisor, can be accessed through FileZilla connecting to the server’s IP, and filling out the username and password fields. Figure 5.14 shows the *SmartCrutch* folder on the server and its contents. The application will establish a HTTP connection to the server using the PHP scripts visible in the figure for synchronization purposes.

Filename	Filesize	Filetype	Last modifi	Permissi	Owner/G...
..					
db.php	35 777	PHP File	18/06/2018...	-rw-r--r--	root root
patientsSync.php	2 737	PHP File	21/08/2018...	-rw-r--r--	root root
php-scripts.log	354 934	Text Doc...	21/08/2018...	-rwxrwxr	root root
physioSync.php	1 526	PHP File	21/08/2018...	-rw-r--r--	root root
sessionSync.php	2 875	PHP File	21/08/2018...	-rw-r--r--	root root

Figure 5.14 – Main folder and files on the server

5.5. PHP Scripts

In order to have communication between the server and the mobile application, PHP scripts are used. These files are stored in the server and every time a physiotherapist synchronizes his application, an HTTP connection is established to the server, through the use of the PHP files. The main idea of using an HTTP connection is to enable communication between a client and a server (see Figure 5.15), where a client can submit a HTTP request to the server and then waits for the response from the server.

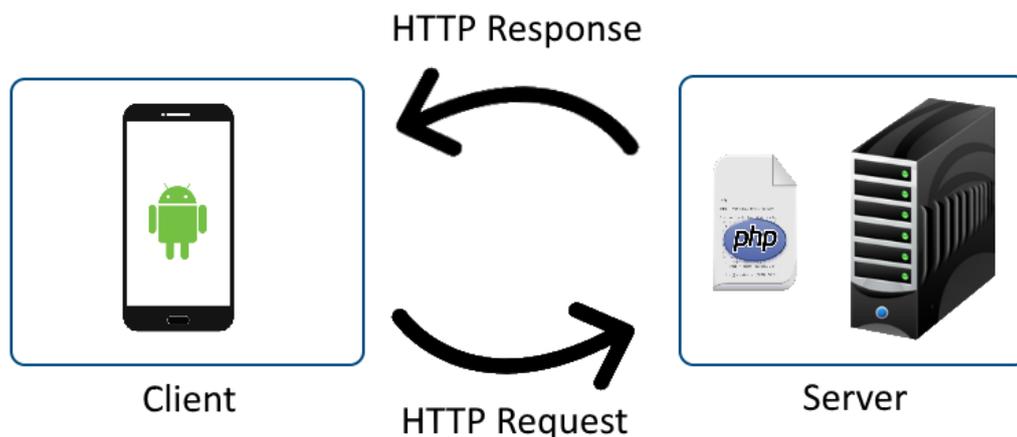


Figure 5.15 – HTTP request-response protocol

This request-response system can be used with the two of the most common HTTP methods, GET and POST. We will be discussing which one we will be using in our project below.

- **GET** – This method requests data from a specified resource;

This method passes the variables (query string) in the URL:

- `/test/GETmethod.php?name1=value1&name2=value2`

Since data is sent in the actual URL, this method is not secure in any ways, as information such as passwords or other sensitive information would be visible in the URL to everyone. There is also a limit to the URL size, and since this method adds the data to the actual URL, it must be smaller than 2048 characters (maximum URL size).

- **POST** – This method is used to send data to a server to create/update a resource.

Unlike the GET method, variables sent are not visible in the URL but instead are stored in the request body of the HTTP request:

- /test/POSTmethod.php HTTP/1.1

Host: iscte.pt

name1=value1&name2=value2

This method doesn't have a restriction on data length (GET is restricted due to URL maximum size) or data type (GET is restricted to only ASCII characters). Since sensitive information regarding patient information is being transmitted, the method used should always be the one providing better security. For this purpose, the POST method was chosen as the HTTP method used. Figure 5.16 shows part of script used for posting physiotherapist's data into the remote database, through the use of *physioSync.php* file and POST methods. Notepad++ was used to develop the PHP scripts.

```

1 <?php
2 ini_set('display_errors', 1);
3 ini_set('display_startup_errors', 1);
4 error_reporting(E_ALL);
5
6 $con = mysqli_connect(" Server IP ", "User", "Password", "SmartCrutchDB");
7 if(isset($_POST['id'][0])){
8
9     foreach($_POST["id"] as $key => $physio){
10         $physio_id = $_POST["id"][$key];
11         $physio_firstname = $_POST["first_name"][$key];
12         $physio_lastname = $_POST["last_name"][$key];
13         $physio_username = $_POST["username"][$key];
14         $physio_password = $_POST["password"][$key];
15         $physio_email = $_POST["email"][$key];
16         $physio_age = $_POST["age"][$key];
17         $physio_phone = $_POST["phone"][$key];
18
19         $statement = mysqli_prepare($con, "INSERT INTO Physiotherapist (physio_id, physio_firstname, physio_lastname,
20 physio_username, physio_password, physio_email, physio_age, physio_phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
21
22         mysqli_stmt_bind_param($statement, "sssssss", $physio_id, $physio_firstname, $physio_lastname,
23 $physio_username, $physio_password, $physio_email, $physio_age, $physio_phone);
24
25         mysqli_stmt_execute($statement);
26     }
27
28     mysqli_stmt_close($statement);
29

```

Figure 5.15 – PHP script example: adding physiotherapists to MySQL database

Initial code, from line 2 to line 4 is used for debugging errors. Debugging and error reporting is not as simple as in Android Studio, so this way we ensure that all the errors from the PHP scripts are sent to a text file located in the SmartCrutch directory. A connection is then started, providing the IP of the server, username, password and database name, using the *mysql_connect* method. Method in line 7 is used to check whether a variable is set or not, in this case the physiotherapist's ID, while also making sure they are not empty. Then, a *foreach* method is used to cycle through all the physiotherapists, while receiving the values sent from the mobile phone with the help of POST method. Information about each physiotherapist, such as first name, last name, username, password, email, age and phone are stored into variables for later use. Prepared

statements [58] are then used for storing data into the remote database. In line 19, *mysqli_prepare* method receives the connection as first argument, and the SQL query as second argument, preparing it and returning a statement handle to be used for further operations on the statement. Variables must be bound to the prepared statement as parameters, using the *mysqli_stmt_bind_param* which receives the previous statement, type of variables sent (in this case Strings, 's') and the variables. Finally, the statement is executed in line 25 with the use of *mysqli_stmt_execute*, storing all variables in the server. When the loop has gone through all the physiotherapists, the statement is finally closed. Figure 5.16 indicates how to retrieve data from the remote server in order to synchronize the local SQLite database.

```
30
31     $sql = "SELECT * FROM Physiotherapist";
32     $result = $con->query($sql);
33
34     if ($result->num_rows > 0) {
35         // output data of each row
36         while($row = $result->fetch_assoc()) {
37             $users[] = $row;
38         }
39         echo json_encode($users);
40     } else {
41         $response = array();
42         $response["success"] = true;
43
44         echo json_encode($response);
45     }
46     $con->close();
47 ?>
```

Figure 5.16– PHP script example: sending physiotherapists to the SQLite database

A query is made to retrieve all the data from the physiotherapist table (line 31) and if there is at least one row of physiotherapist data in the table (greater than 0), the information will be encoded as a JSON message and sent to the mobile application for decoding. If there are no physiotherapists, it simply sends a message to the mobile phone, also encoded in JSON.

Data synchronizing should always be completed, meaning if for some reason the connection between the server and the mobile device is interrupted while data synchronization is in process, the physiotherapist should fully synchronize the application to make sure the databases don't get corrupted. This shouldn't be a problem since synchronization only takes a few minutes, but in the future this issue should be resolved.

Chapter 6 – Mobile Application

This chapter describes the mobile application of the SmartCrutch system, designated as “IoPhyr”. It shows the layout schemes of every activity while also discussing the main features used for the application’s development.

6.1. Main Features

Some important features used in the development of the mobile application are listed below:

- **Authentication** – In order to successfully login in the application, the physiotherapist must be registered in the local database. If the physiotherapist is already registered but the local database has no information related to him, synchronization must be made from the remote database to the local database simply by turning on Wi-Fi and clicking on the synchronize button. Each physiotherapist should only have one account, with a username and password which can be changed after logging in. This authentication will allow for the physiotherapist to view information only regarding patients that are being treated by him.
- **Patient form** – Patient information is visible digitally by information that is shared by the patient to the physiotherapist. This will ensure that no paper is used, while all the patient’s information is stored on the physiotherapist’s mobile device and can be accessed anytime, anywhere. Relevant basic information, such as first name, last name and a photo that can be taken anytime is included in the patient form. Also, information relevant to training purposes such as age, weight, height and date of patient register is also included. Finally, a phone number, address and email fields are also filled in order to have a way to communicate with the patient outside of sessions, if necessary.
- **Data Charts** – Information received by the sensors is represented in line charts and bar charts. A library, *MPAndroidChart* [59] was used in order to represent the results regarding patient sessions. This library is highly customizable and is commonly used for the purpose of visualizing data in charts. Line charts are used for real time data visualization and also when visualizing data from a single session. A bar chart is used for displaying the mean sensor values of each sensor, providing the physiotherapist with an overall development of the patient. An example of both charts is visible in Figure 6.1.

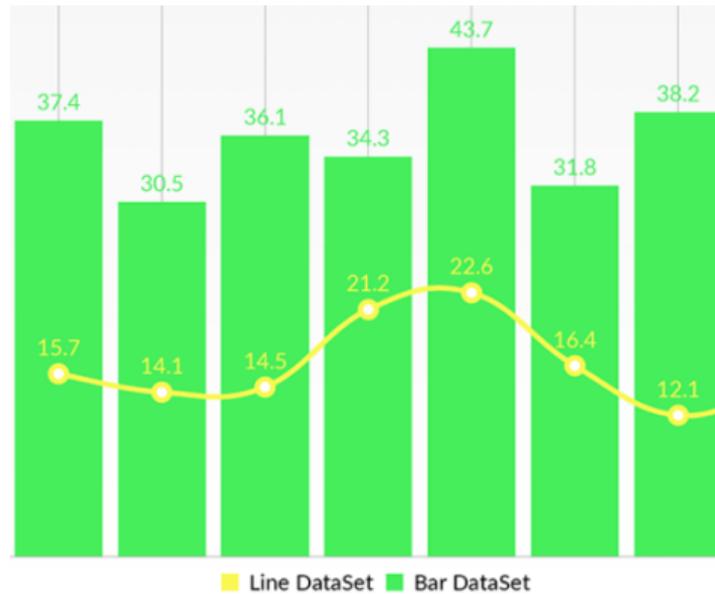


Figure 6.1 – Line chart and Bar chart example

- **Threads** – Threads are commonly used in JAVA, which is known as being a multi-threaded programming language. A multi-threaded program consists of two or more parts of the program that run concurrently, while each part is assigned a different task. This is especially useful when sensor data information is being transmitted by Bluetooth from the SmartCrutch; this way we can receive the data and refresh the GUI at the same time. Figure 6.2 shows the life cycle of a thread.

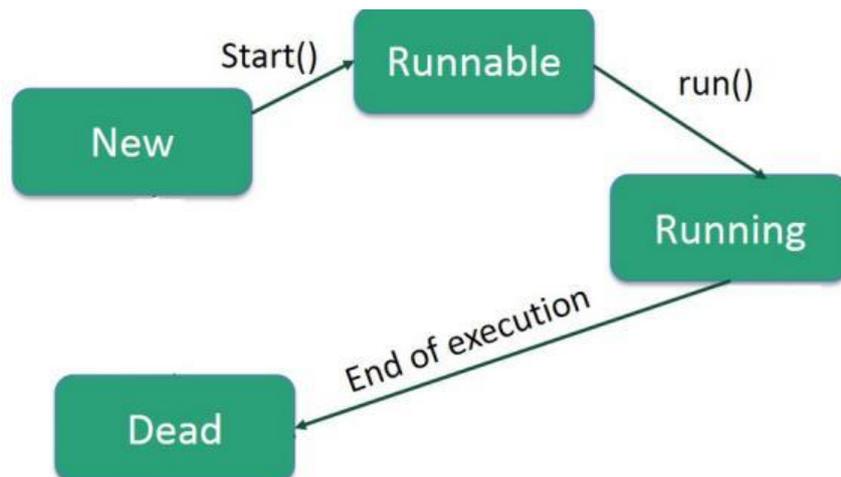


Figure 6.2 – Life cycle of a thread

- **New** – This is the initial state of a thread in its life cycle and it remains in this state until the thread is started.

- **Runnable** – After the thread is started it becomes runnable after executing the *start()* method. In this state, the thread is just in a started phase and is not executing the task.
- **Running** – This is the state of the thread where the task is being executed, after executing the *run()* method. The thread will run concurrently with other existing threads in the program.
- **Dead** – When the task assigned to the thread is completed, the thread is considered being terminated.

Three threads were created for Bluetooth data communication between the mobile device and the Bluetooth adapter connected to the Arduino:

- **Accept thread** – This thread runs while listening to incoming connections, behaving like a server-side client. It will run continuously until a connection is accepted. This thread initiates the server socket when a connection is accepted.
- **Connect thread** – This thread will run while attempting to make an outgoing connection to a device, in this case, the Bluetooth adapter connected to the Arduino. The connection either succeeds or fails. A client socket is created in this thread and then a *connect()* method is called in order to make a connection to the client socket.
- **Connected thread** – This thread runs during a connection with a remote device, handling all incoming and outgoing transmissions. Input streams and output streams are created in the socket in order to receive and send information via TCP. All the information sent from the Bluetooth adapter to the mobile device is treated correctly in this thread.
- **Sockets** - Threads are commonly used with sockets for communication purposes, such as client-server applications. Communication is done via TCP, which provides a reliable point-to-point communication channel between client-server applications. Both server and client must establish a connection with each other by binding a socket to the end of the connection. Socket classes (*ServerSocket* and *Socket*) are used in order to represent the connection between a server and a client (see Figure 6.3). Creation of sockets and data stream information is done inside created threads.

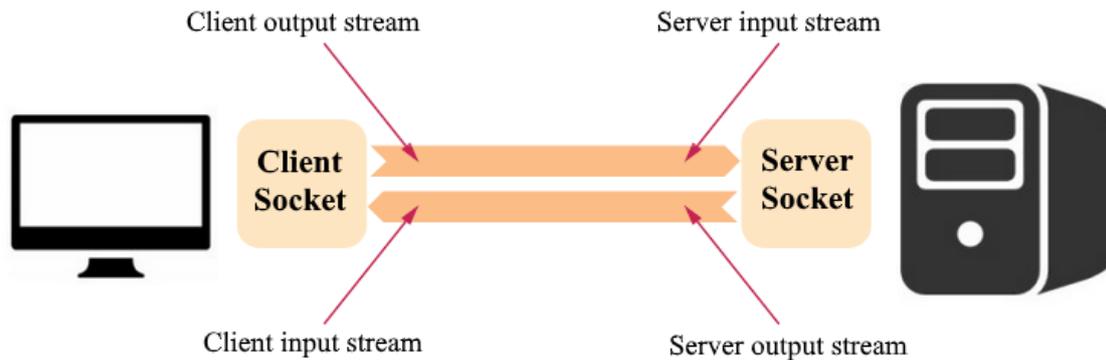


Figure 6.3 – Client-Server socket communication

After communication has ended, it is important to close both data streams and sockets in order to ensure proper TCP shutdown sequence.

- **HTTP Connection** – Communication from the mobile application to the database is done through HTTP requests. For this purpose, an `AsyncTask` class was used, allowing the user to perform background operations and publish results on the UI thread. This class is ideally used for short operations, a few seconds at most, which makes it ideal for database synchronization. Figure 6.4 indicates the steps of an `AsyncTask`. In the application developed, the `onProgressUpdate` method was not used.

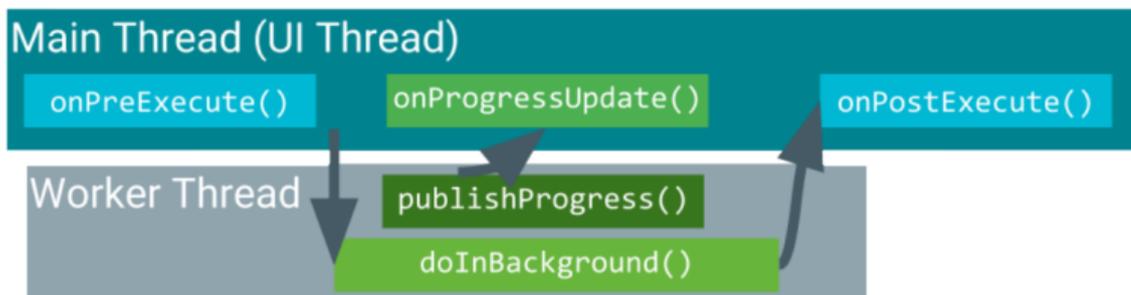


Figure 6.4 – AsyncTask methods

- **onPreExecute** – This method is invoked on the UI thread before the task is executed and will display a dialog visible to the physiotherapist indicating that synchronization is ongoing.
- **doInBackground** – Responsible for all the data processing, this method is called immediately after `onPreExecute` finishes. This is where the HTTP connection is established, and the HTTP request is executed (POST method). It runs on a separate thread, meaning that the UI is not locked while the method is running. Local database data is synchronized with remote database data, and

vice versa. The result of the computation in this step will be passed to the last step, to the *onPostExecute* method.

- **onProgressUpdate** – Not really necessary to use in the application, however it can be invoked in the UI thread after a call to the *publishProgress* method, which is used to display any form of progress in the UI while background computation is still executing.
- **onPostExecute** – This method is invoked by the UI thread after the background computation finishes and receives the result of it as a parameter. This method is called using a Handler and its main purpose will be to dismiss the message shown in the *onPreExecute* method.

6.2. Design and Implementation

When developing a mobile application, the interface design is one of the most important things to keep in mind. The interface should always be simple, clean and intuitive in order to provide the best possible user experience. An application based on icons was developed, providing a clear and simple application usage. Figure 6.5 shows the application's logo on a mobile device. The application was named IoPhyr, which stands for Internet of Physical Rehabilitation.



Figure 6.5 – IoPhyr application logo

When the application is executed, the authentication activity, also known as login (see Figure 6.6), is presented to the user. If the physiotherapist already created an account but is using a different mobile device, or cleared the local database data, he/she should start the synchronizing process by turning on the mobile device's Internet and by clicking on the synchronizing button visible in the top right of the screen.

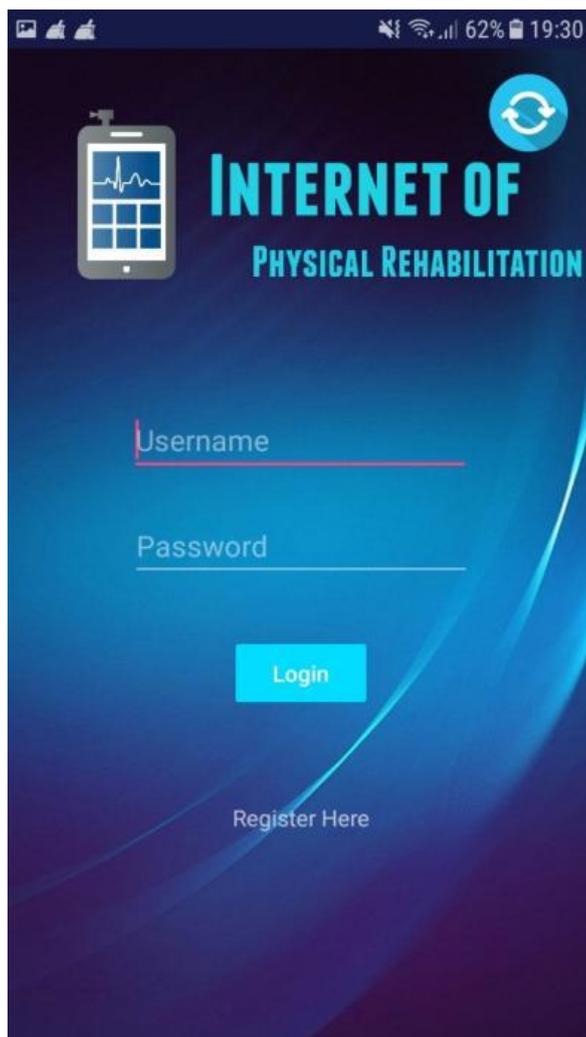


Figure 6.6 – Login activity of the IoPhyr application

It is important to note that the synchronize button in this activity only synchronizes the physiotherapists of the system in order to have a successful login; it does not synchronize the patients and the sessions. The synchronizing button is only shown if there is Internet available to the mobile device, otherwise it should be hidden. If the physiotherapist is already registered, he should simply insert his username and password for authentication purposes, and then click on the Login button. If the physiotherapist is not registered, he should click the text below the Login button, “*Register here*”, which would direct him to a new activity where a registry form for physiotherapists is available, seen in Figure 6.7. Cancel button and arrow icons present in the application simply go back to the previous activity, not saving any changes made.



The image shows a mobile application interface for registering a new physiotherapist. The title is "NEW PHYSIOTHERAPIST" in large, bold, light blue letters. Below the title are seven input fields, each with a light blue underline: "First Name", "Last Name", "Username", "Password", "Email", "Age", and "Phone Number". At the bottom of the form are two buttons: "Cancel" and "Register", both in light blue with white text. The background is a dark blue gradient with a light blue abstract shape on the right side. The top of the screen shows a status bar with icons for signal, Wi-Fi, battery (62%), and time (19:30).

Figure 6.7 – Register activity for physiotherapists

Information such as first name, last name, username, password, email, age and phone number must be filled. Failure to fill out all the data will return in a popup message to the user, preventing him to successfully register while also telling the user what to do. Some regular verifications are also made, such as not allowing two physiotherapists with the same username. After filling up the form correctly, pressing the Register button will add the physiotherapist to the local database and the physiotherapist should also synchronize this new data to the remote database. Random UUID's are created as primary keys in database tables to avoid synchronization issues with regular ID's incrementation.

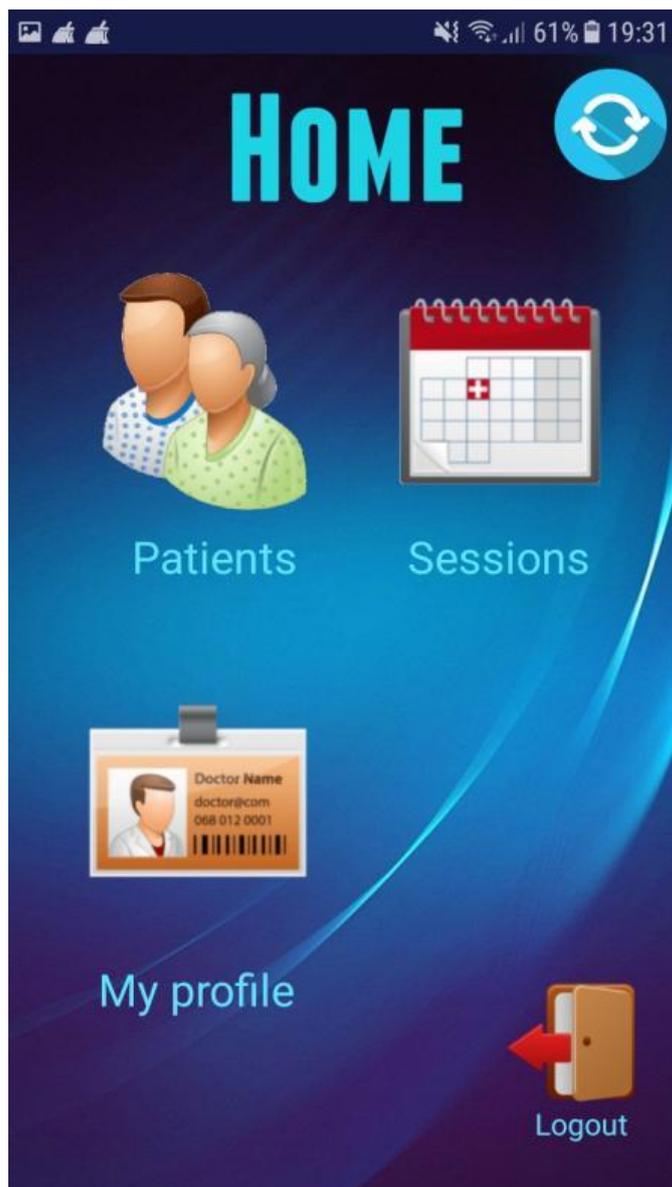


Figure 6.8 – User area activity

After successfully logging in, the physiotherapist is presented with an icon-architecture layout with the intention of providing a clean and simple usage of the application. Figure 6.8 shows the user area activity, where the user can press multiple buttons for different actions. Logging out of the application is simply done by clicking on the Logout icon. It is also possible to view and edit the physiotherapist's own profile by clicking on the My Profile icon. Adding patients or viewing existing patients is done by clicking the Patients icon. Finally, starting new sessions and viewing previous ones is also possible by clicking the Sessions icon. Synchronizing button in this activity will only synchronize patient and session data that are related to the physiotherapist that logged in. This means that the mobile device of each physiotherapist should include only information that is relevant to him, avoiding unnecessary data to be stored in the database.



The screenshot displays a mobile application interface for a physiotherapist's profile. The title 'MY PROFILE' is prominently displayed at the top in a large, bold, light blue font. Below the title, there are seven input fields, each with a label on the left and a value on the right. The fields are: First Name (Martim), Last Name (Valente), Username (martim), Password (123), Email (mvalente@hotmail.com), Age (36), and Phone (968532526). At the bottom of the screen, there is a blue arrow pointing left and a blue rectangular button labeled 'Edit'.

Field	Value
First Name	Martim
Last Name	Valente
Username	martim
Password	123
Email	mvalente@hotmail.com
Age	36
Phone	968532526

Figure 6.9 – Physiotherapist profile activity

All data regarding the physiotherapist that is logged in can be changed in the physiotherapist profile activity (see Figure 6.9). If any field was incorrectly filled while registering, changing it is possible by clicking the Edit button, which would allow the user to edit any field and then save the editing.

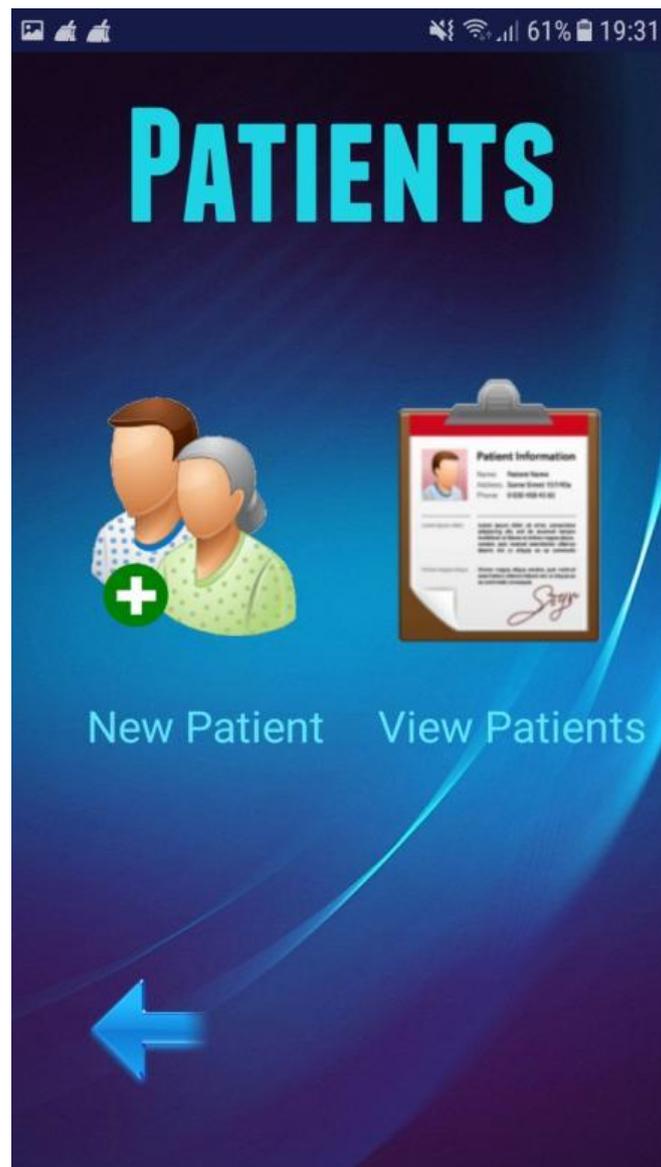


Figure 6.10 – Patients activity

If the physiotherapist decides to view existing patients or add new ones, he should click the Patients icon visible in Figure 6.8. This will bring up a new activity, where two new icons are available: one for creating new patients and one for viewing existing patients (see Figure 6.10).

NEW PATIENT

First Name

Last Name

Email

Age Weight (kg) Height (cm)

Phone

Address

Cancel Register

Figure 6.11 – New patient activity

When the New Patient icon is clicked, a new activity is displayed to the user (see Figure 6.11). This is a regular patient form that most physiotherapists usually fill using paper. By clicking on the photo icon, it is possible to take a picture of the patient, which can be edited any time. Information such as first name, last name, email, age, weight, height, phone and address is also required for the successful register of a new patient. Clicking the Register button will either inform the physiotherapist that the patient was successfully added or, if any field was left in blank (except for photo), it will display an error.

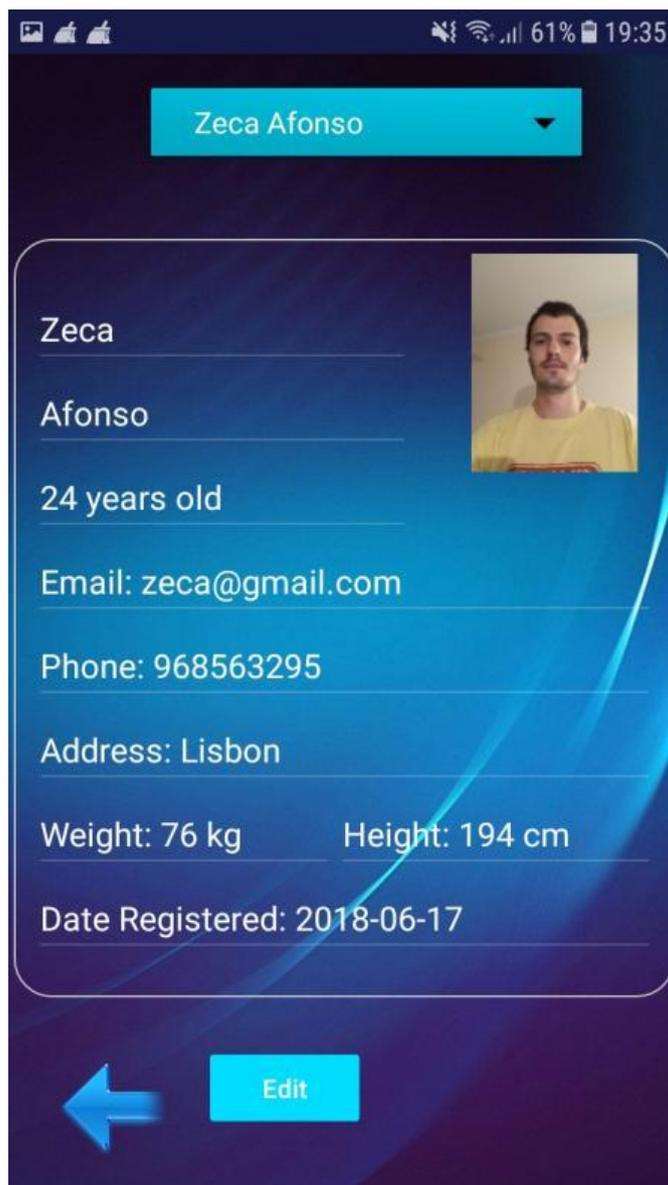


Figure 6.12 – View patients' activity

If instead of creating a new patient, the physiotherapist desires to view information regarding all his patients, he should click the View Patients icon visible in Figure 6.10. This will bring the user to a new activity (see Figure 6.12), where a scroll list of all the patients is found on top of the screen, being possible to view all the patients associated with the physiotherapist, displayed by first and last name. When clicking on each patient, all their information is automatically filled in the form, which can be edited at any time. Date registered indicates the date when the patient was created and it is the only field that can't be edited. To edit patient information, simply clicking the Edit button will allow the user to change said information and allowing a new photo to be taken. After clicking the Edit button, a Save button and a Cancel button appear, where their behavior is exactly the same as the Register and Cancel button in the previous activity (Figure 6.11).



Figure 6.13 – Selecting patient for session activity

Besides viewing his/her personal data and patient data, it is also possible for the physiotherapist to start and view previous sessions of certain patients. When clicking the Sessions icon seen in the user area activity (see Figure 6.8), a similar layout to the patient view activity is displayed, where the physiotherapist must choose the patient that he wishes to start a new session or view old session data. This is done by simply selecting the right patient and clicking the Next button, as seen in Figure 6.13.



Figure 6.14 – Sessions activity

After the patient has been chosen, its name is displayed in the top region of the screen, so the physiotherapist always knows which patient he/she is dealing with. Two new icons are displayed, one for starting a new session and another for viewing previous sessions (see Figure 6.14).



Figure 6.15 – New session activity

Starting a new session can be done by clicking the New Session icon visible in Figure 6.14. A new activity is displayed, where Bluetooth connection can be turned on and off by clicking a button (see Figure 6.15). If Bluetooth is available, a new button is shown that will discover all Bluetooth devices nearby and display them in a list. Pairing with such devices is done by simply clicking the right device. In this case, BTWalker1 was the name given to the Bluetooth Adapter connected to the SmartCrutch. After pairing, a connect button is displayed and when clicked it starts receiving sensor information regarding the patient's training session and displays them in charts. By scrolling down the screen, charts displaying FSR, load cell and IMU data are visible to the physiotherapist for live data analysis.

Each live chart provides a good flow of data, ensuring that the flow of data shown to the physiotherapist is acceptable by indicating a fixed number of visible samples before

the chart moves. Charts Y values are expressed when viewing single session data and X values are displayed in minutes. Live session data is supposed to be a quick visualization of data flowing.

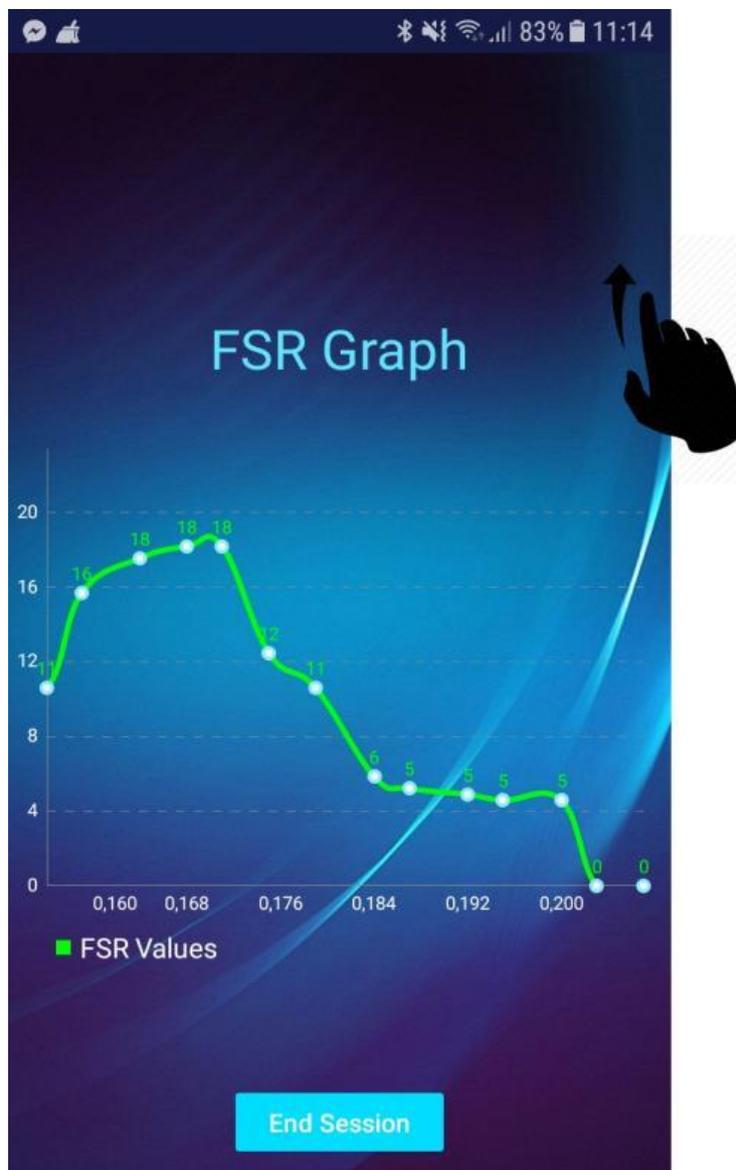


Figure 6.16 – Live FSR sensor data

After the session has started, data from the SmartCrutch is transmitted to the mobile device and is visible by scrolling down, by doing the hand movement present in Figure 6.16. Data shown in this figure is to demonstrate a normal hand grip on the crutch's hand support while slowly decreasing force applied, until value drops to zero, which is when no force is being applied.

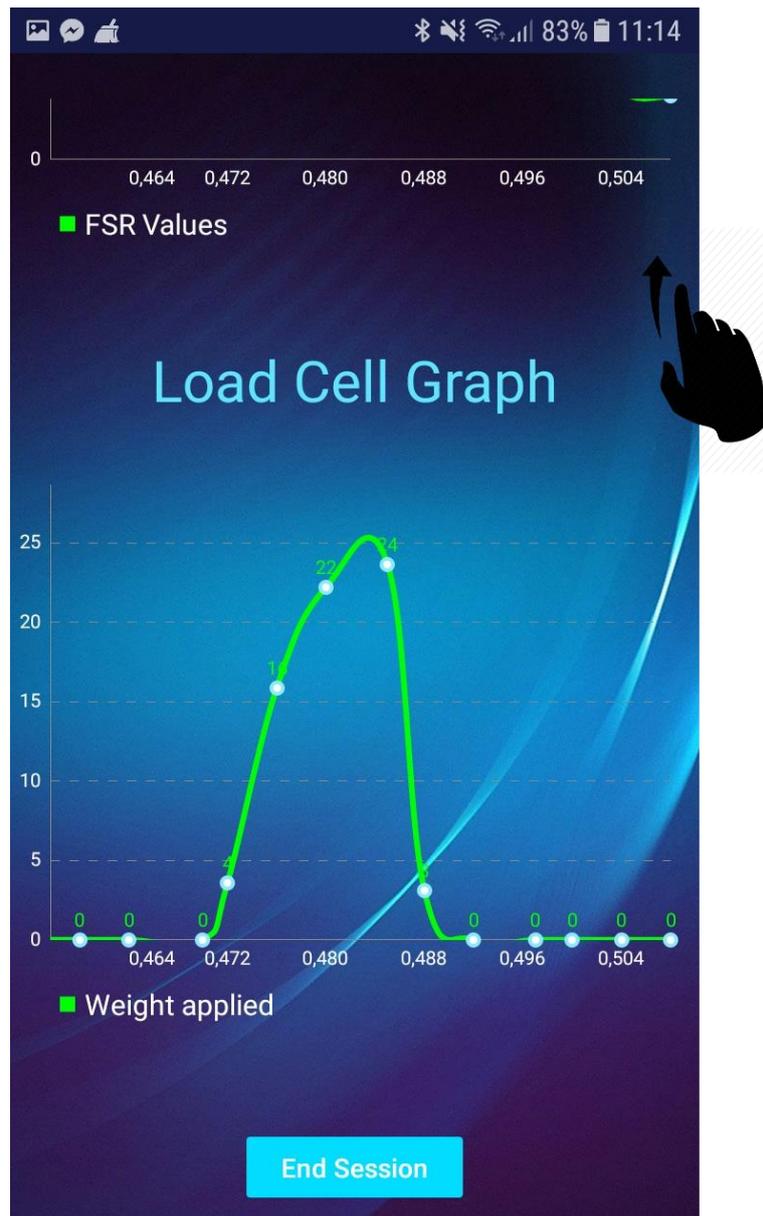


Figure 6.17 – Live load cell sensor data

By scrolling down again the live session activity, a load cell data chart is available. Figure 6.17 shows an example of a step taken. The load cell reached a maximum value of 24 kilograms, and when not in contact with the floor the value dropped to zero.

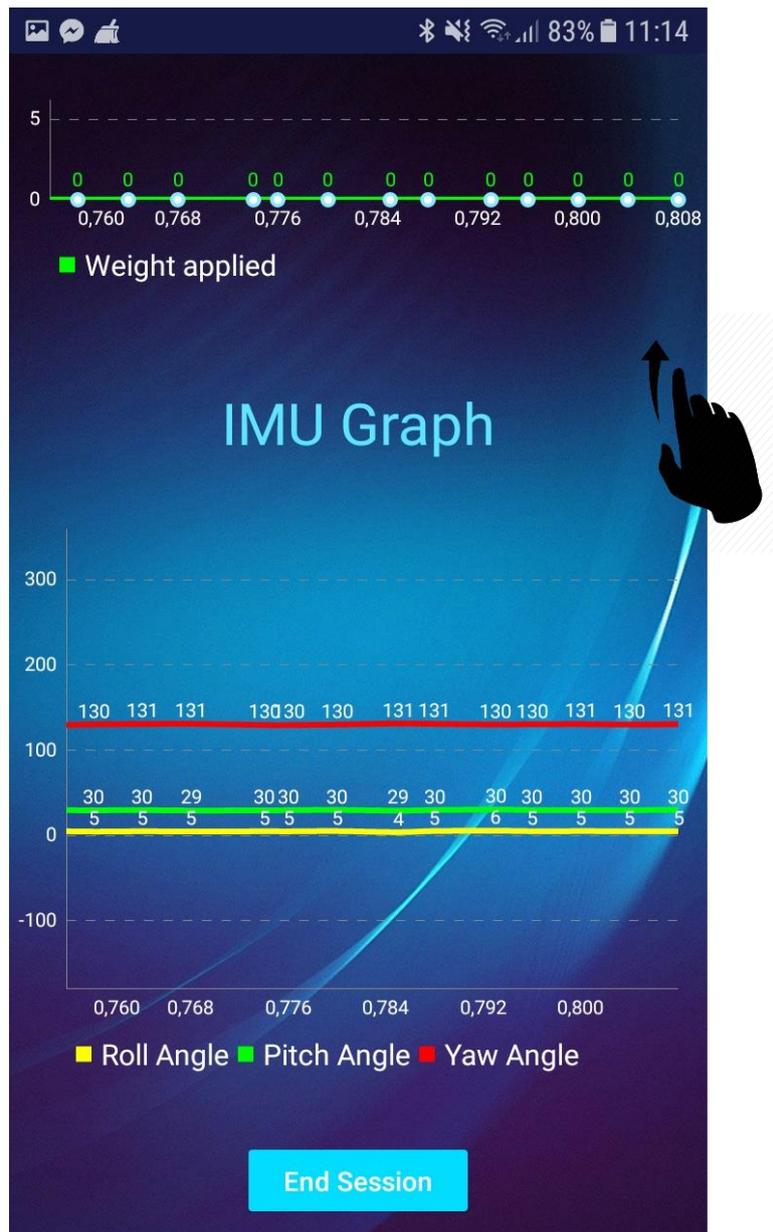


Figure 6.18 – Live IMU sensor data

Finally, the data displayed the IMU sensor is visible in the last chart (see Figure 6.18) where more relevance is given to the roll and pitch angle, which indicate horizontal and vertical inclination of the crutch, respectively. Positive and negative values are possible. Values from 0° to 180° indicate that the crutch is being tilted forward in case of the pitch angle, and towards the right in case of the roll angle. Values from -180° to 0° indicate that the crutch is being tilted backwards in case of the pitch angle, and towards the left in case of the roll angle. This information is visible when the physiotherapist is viewing single data sessions. Finally, yaw angle ranges from 0° to 360° and is simply to indicate when the patient moves the crutch's orientation, which is useful to understand when the patient turned his body in the session.

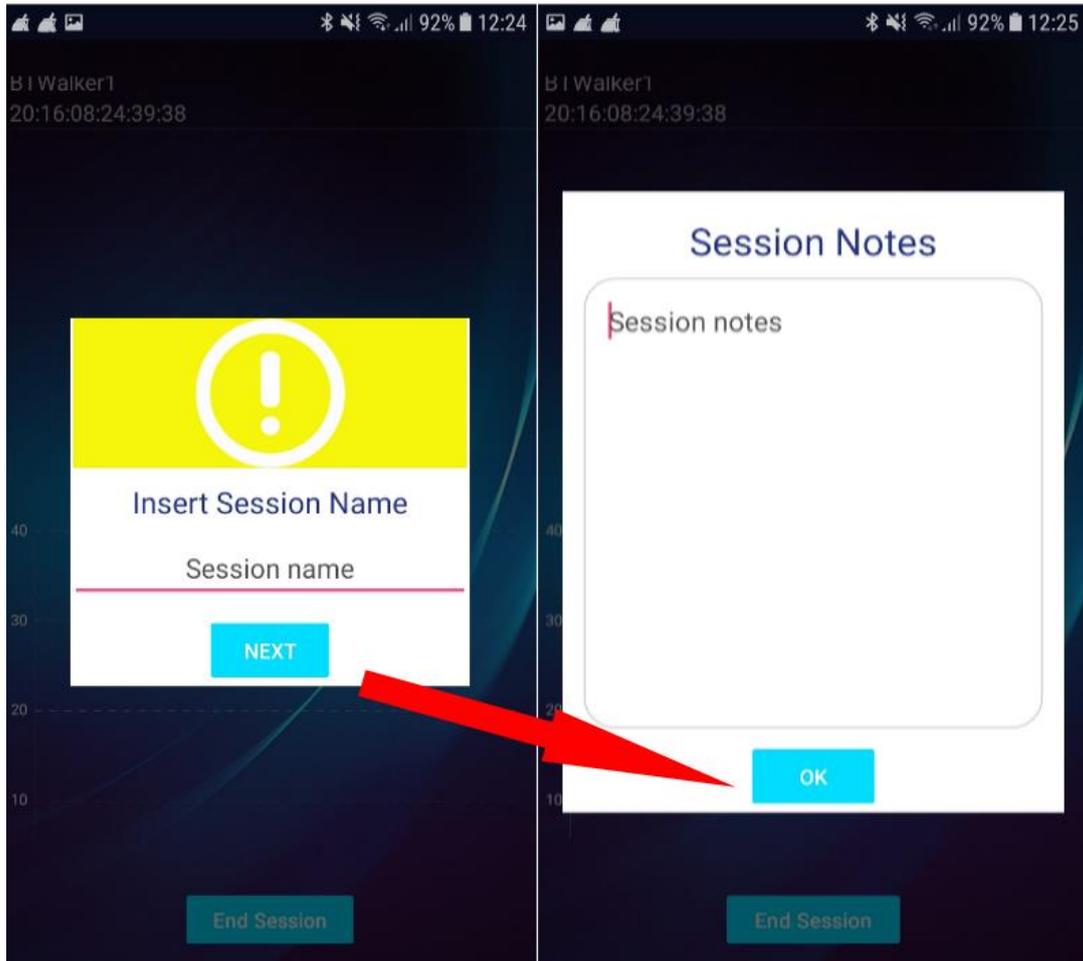


Figure 6.19 – Session name and notes

The session ends when the physiotherapist clicks the End Session button that is always visible during a session, and immediately a popup window shows up asking for the physiotherapist to name the session created. Physiotherapists usually take notes, and like the patient form, this kind of information can also be stored digitally and accessed anywhere, everywhere, instead of resorting to physical information written in paper.

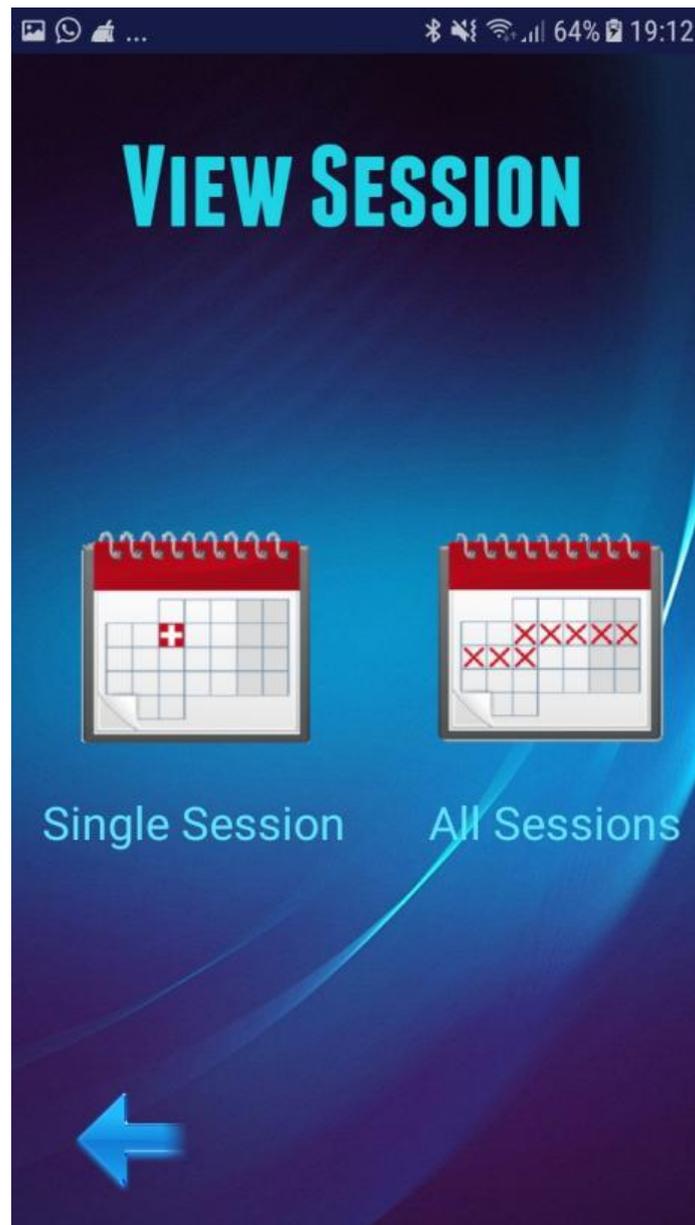


Figure 6.20 – View session activity

After a session has ended, the application automatically enters the Viewing Sessions activity (see Figure 6.20), which can be accessed by clicking the View Sessions icon visible in Figure 6.14. This activity presents the user with two new icons which provide the possibility of viewing single session data, including but not excluding the new session that was created, and the data regarding all the sessions the patient has done.



Figure 6.21 – View single session

If the patient decides to analyze a single session, a list of all the selected patient's session show up (see Figure 6.21), providing information such as session number, name, date and time created. Session duration time is visible once the physiotherapist selects a session. Deleting sessions is also possible by clicking the X icon on the top right of screen.

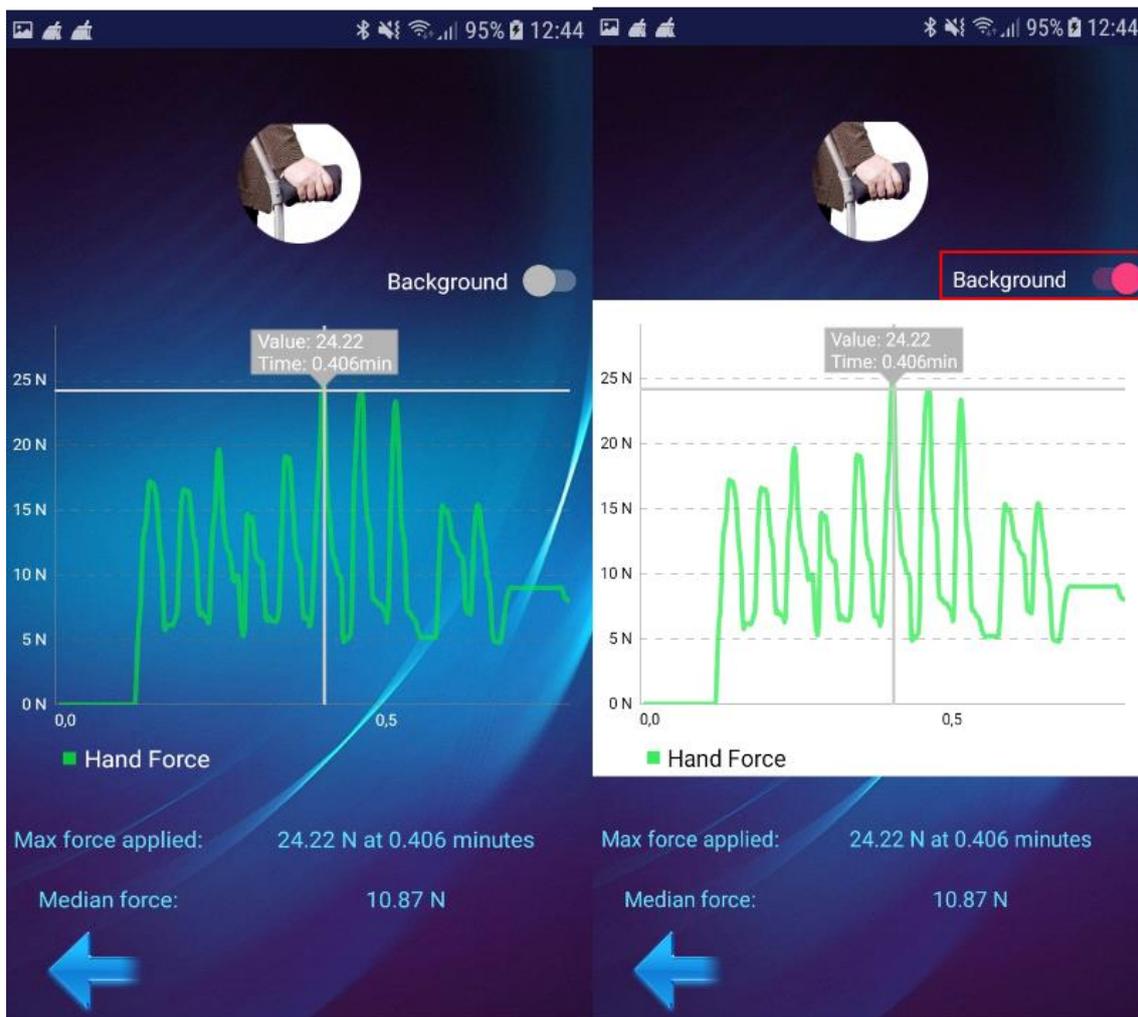


Figure 6.22 – FSR data analysis

When a session has been selected a new activity appears. Notes taken from the session can be seen in this activity, as well as charts containing sensor information and other important information, such as maximum force and median force applied, in the case of the FSR sensor, as seen in Figure 6.22. A transparent background is set as default but if the physiotherapist wants to change it to a white background, simply clicking the toggle Background switch button enables this. Clicking and zooming on the charts is also possible, with the first one providing information about the sample clicked, such as the value and the time at which that value was taken.

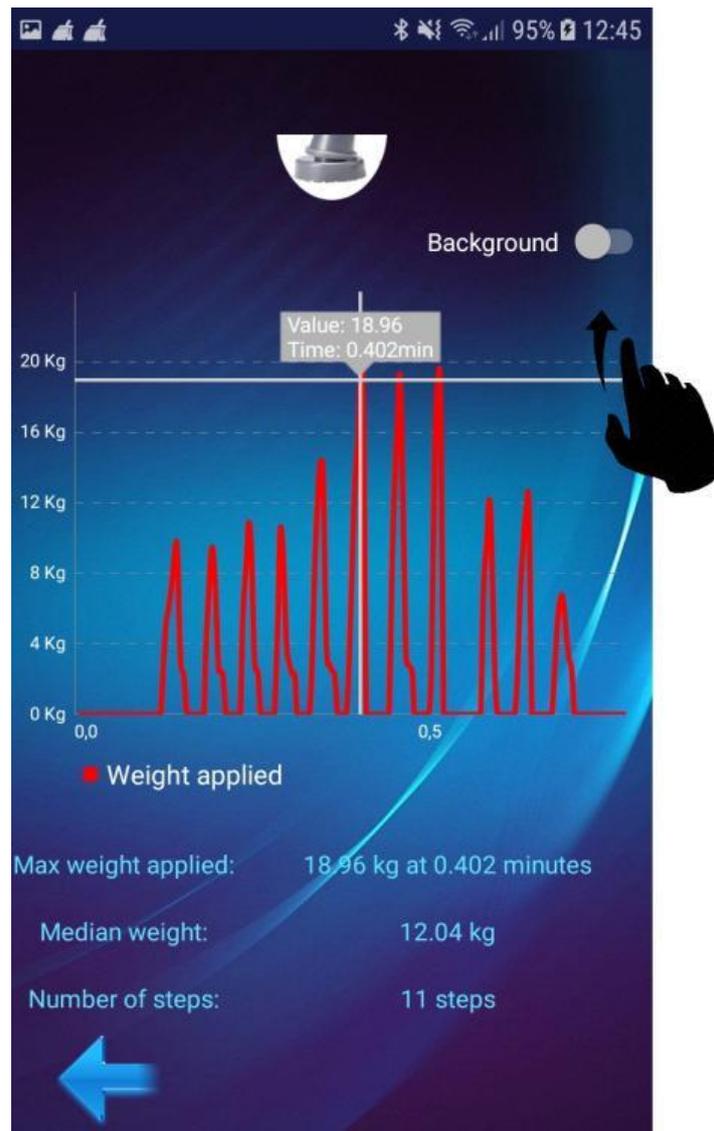


Figure 6.23 – Load cell data analysis

Similar to the live session activity, this activity can also be scrolled down for viewing more information. Figure 6.23 shows the chart of the load cell data acquired, while also indicating the maximum and medium weight applied, as well as the number of steps done by the patient.

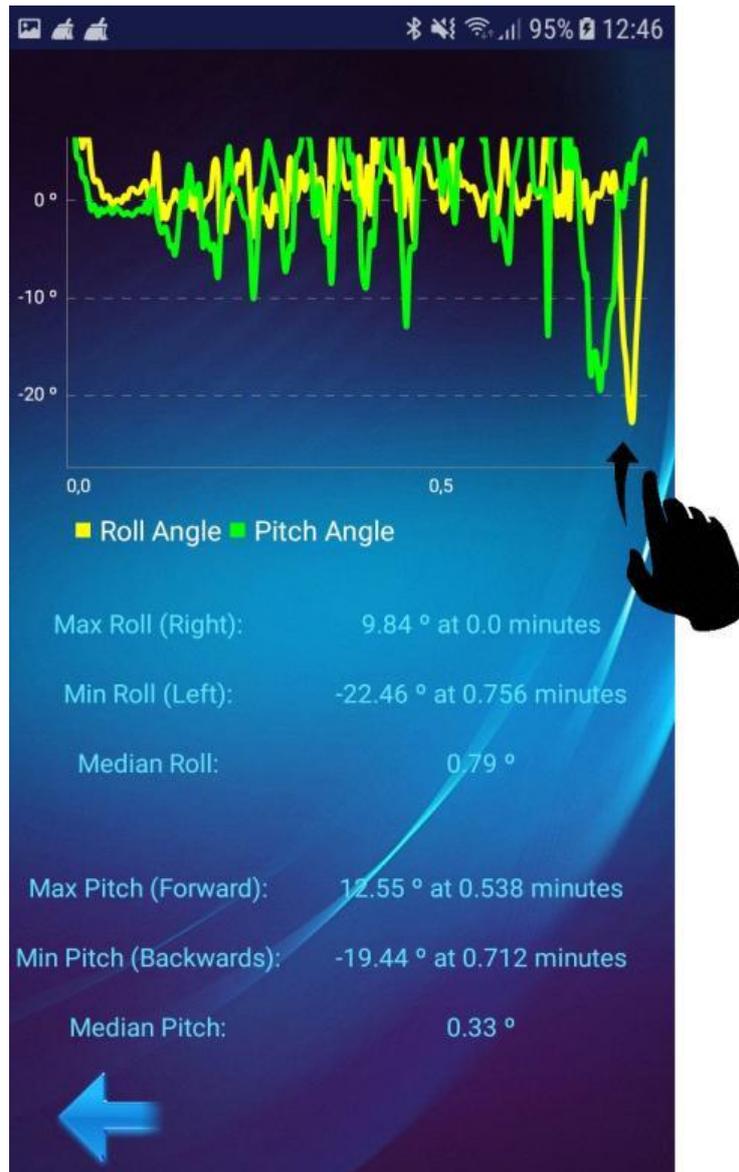


Figure 6.24 – IMU data analysis

Finally, data collected from the IMU is visible in the last chart (see Figure 6.24) where multiple information is also displayed. As mentioned before, values from 0° to 180° indicate that the crutch is being tilted forward in case of the pitch angle, and towards the right in case of the roll angle. Values from -180° to 0° indicate that the crutch is being tilted backwards in case of the pitch angle, and towards the left in case of the roll angle. Maximum and minimum values of both angles are displayed, as well as the median angles, which are a great indicator in the patient's gait.

Instead of viewing a single session, the physiotherapist may choose to view the overall progress of the patient by clicking the All Sessions icon in Figure 6.20.

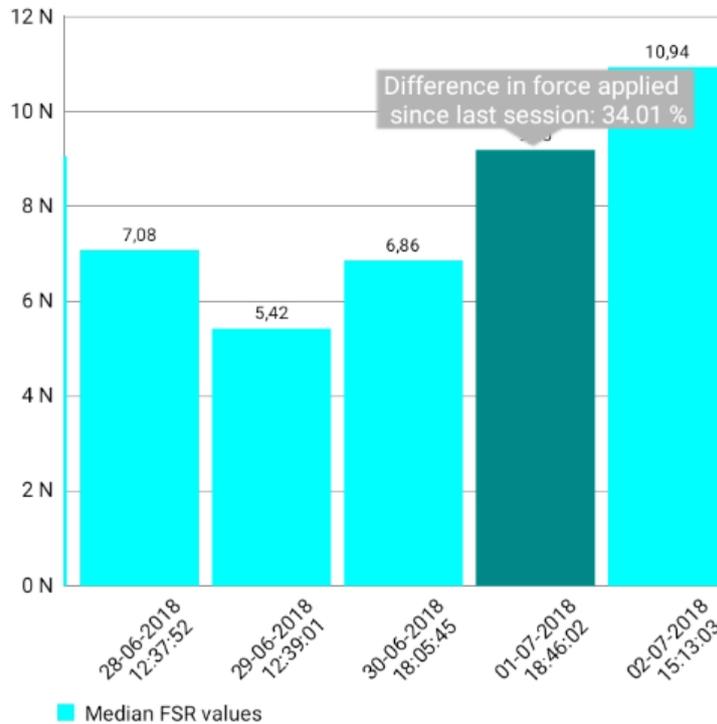


Figure 6.25 – Negative patient progress considering FSR data

Figure 6.25 indicates only one of the bar charts visible to the user, which considers the median FSR force of each session. Charts with other sensor information are visible by scrolling down the activity, as before. Date and time are displayed in the X axis while the mean value is displayed in the Y axis. Only 5 bars were made visible at once to avoid cluttering. By scrolling the chart to the right or left, more sessions will become visible. When clicking a session bar, it will indicate the difference of value between the session clicked and the last session, in percentage. A negative difference means there was less force applied, therefore leading to patient improvement, while a positive difference means the force applied was higher, leading to a negative improvement. This figure indicates a negative patient progress, since after the second visible session the mean force applied kept increasing. When this happens, the physiotherapist should try to find new approaches to training to understand what is going wrong, in order to get the patient back to a positive progress.

Chapter 7 – Results

This chapter focuses on demonstrating the experimental results associated with the SmartCrutch system tests. The first phase of testing revolved around testing each individual sensor to realize if the information being sent from the Arduino was correct. The second phase will be carried out by several healthy patients for real testing sessions. Three males and one female, with certain ages, weights and heights successfully tested the SmartCrutch system for a session duration of one minute. Important information such as number of steps taken, weight applied to the crutch, force applied to the crutch's hand support and gait information related to each patient will be discussed. Room temperature for each test was adequate, like it should be on any physical therapy clinic.

7.1. Individual sensor testing

All the sensors were tested individually to ensure that the values processed on the Arduino were accurate. As mentioned before, calibration was previously done with known weights and forces with the help of a scale.

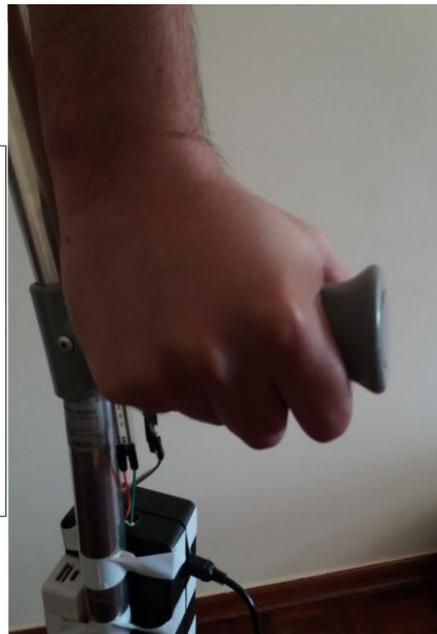
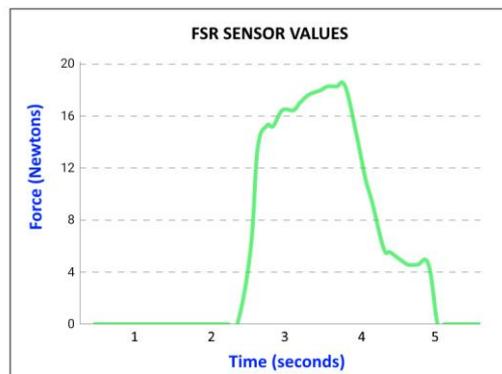


Figure 7.1 – FSR sensor values test

A testing where a patient applies a force on the crutch's hand support resulted in the values plotted in Figure 7.1. This testing was purely made to understand if the sensor was functioning correctly and measuring accurate values. A regular grip was made, reaching a value of approximately 18 Newtons. It was then let go, smoothly, declining to a value of approximately 4.5 Newtons before reaching zero, which was when the patient stopped applying any force to the crutch's hand support.

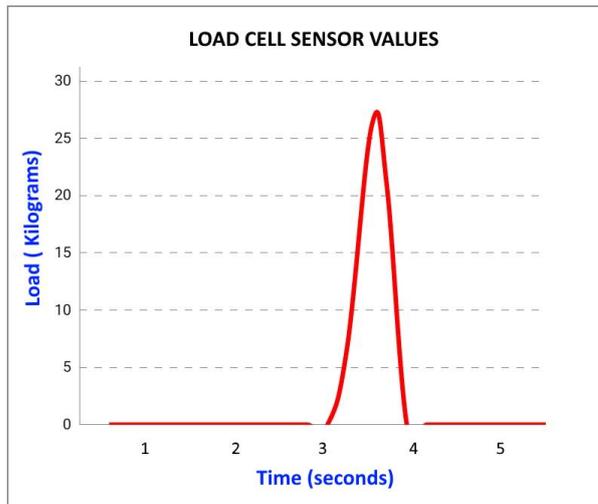


Figure 7.2 – Load cell sensor values test

The load cell sensor was then tested. A single step was taken just to ensure the values received are correct, and as can be seen in Figure 7.2 the sensor provided accurate data. A value of approximately 26 kilograms was obtained in the single step testing. When the crutch wasn't in contact with the floor, the value was zero, as expected.

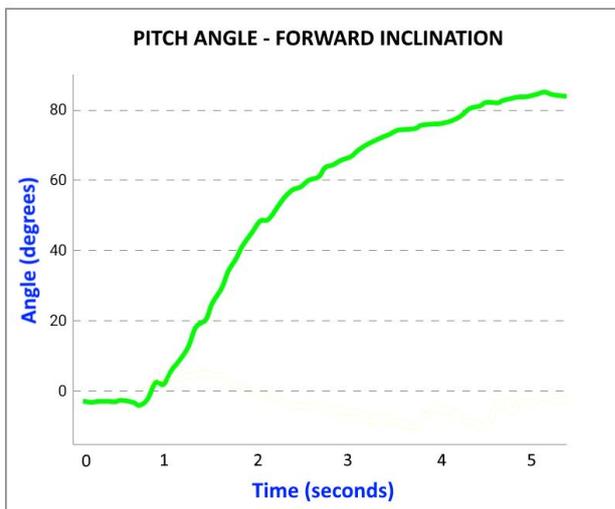


Figure 7.3 – IMU sensor front inclination

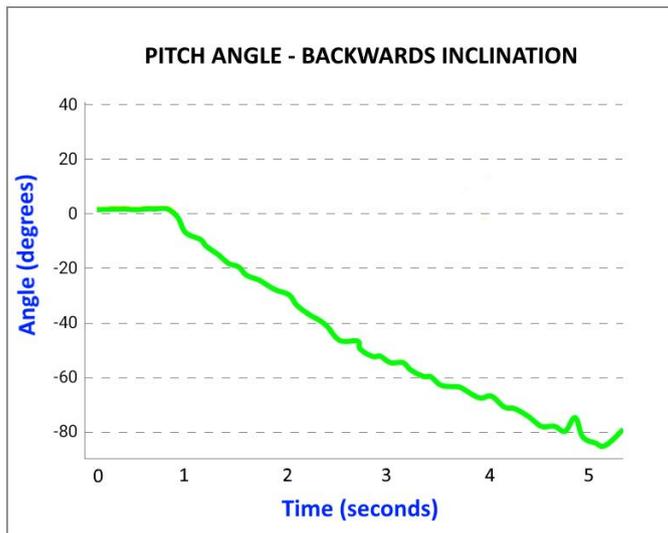


Figure 7.4 – IMU sensor back inclination

Inclination, both vertical and horizontal are important for determining patient gait. Firstly, tests around the pitch angle were made. A test where the crutch is held at approximately 90 degrees vertically from the floor was done, tilting forward as seen in Figure 7.3. The crutch was initially held in the vertical position, providing a value of approximately 0 degrees, slowly increasing to the 90 degrees as expected.

On the other hand, when doing the opposite test, that is, tilting the crutch 90 degrees backwards, the values received ranged from 0 to -90 degrees as expected (see Figure 7.4). As mentioned in chapter 6, positive values of pitch angles indicate a vertical inclination pointing forwards and negative values of pitch angles indicate a vertical inclination pointing backwards. Movement regarding these tests was done slowly, taking an approximate 5 seconds in total.

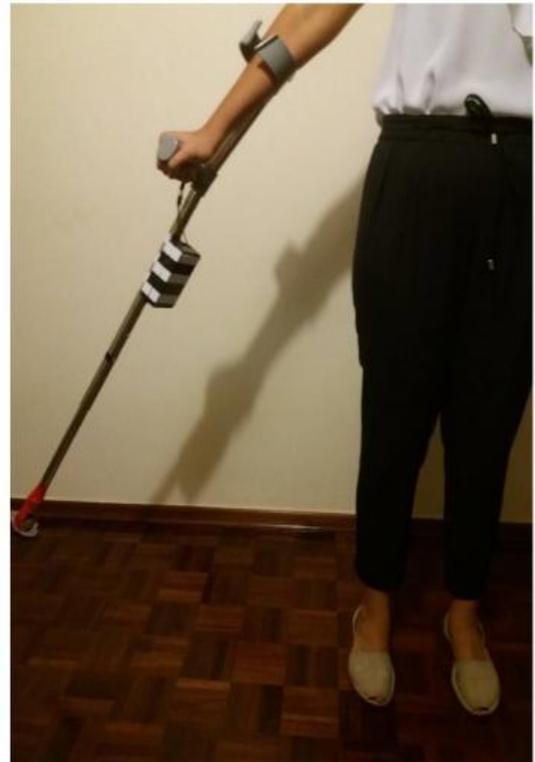
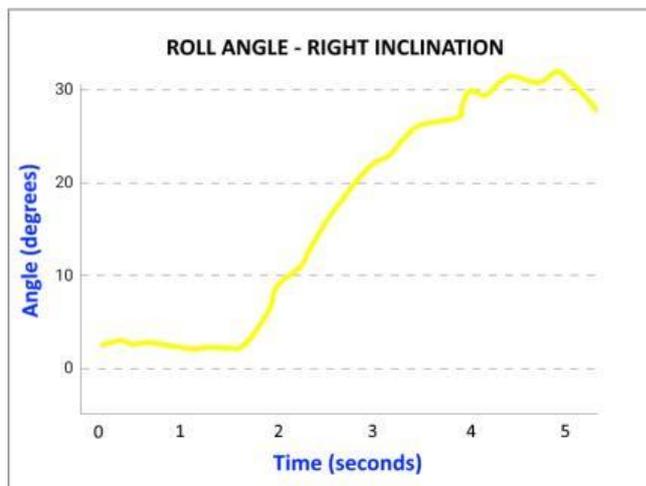


Figure 7.5 – IMU sensor right inclination

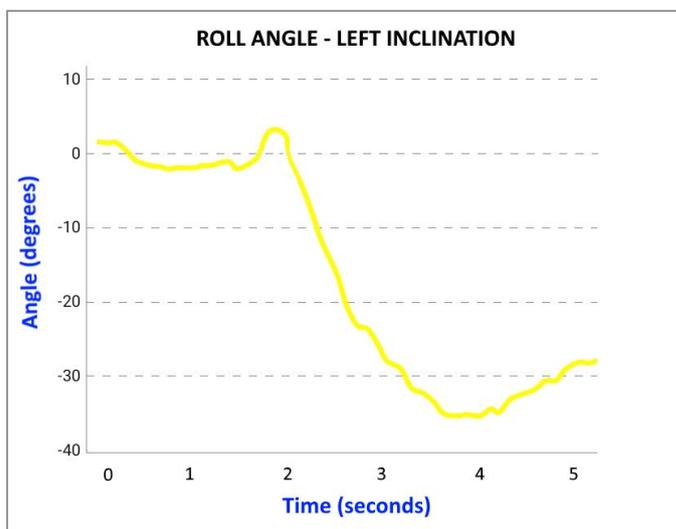


Figure 7.6 – IMU sensor left inclination

Regarding horizontal inclination, roll values are used. This time, an angle of approximately 30 degrees was made. Figure 7.5 indicates the values received when the crutch was tilted to the right, while Figure 7.6 shows the values when the crutch was tilted to the left. As mentioned in chapter 6, positive values of roll angles indicate a horizontal inclination to the right side while negative values indicate a horizontal inclination to the left side. Right and left inclination are always set depending on the patient orientation. In Figure 7.6, despite showing the patient holding the crutch and tilting it to the right, in his/her perspective the crutch is actually tilted to the left. These tests were also done slowly as the previous ones.

Values received by all the sensors were proven accurate providing the possibility of realizing real patient tests.

7.2. Patient tests results

Four healthy patients were chosen to test the SmartCrutch system. For testing purposes, patients were named Patient 1, Patient 2, Patient 3 and Patient 4. Information such as sex, age, weight and height can be seen in Table 8.

Table 8 – Patient information for testing purposes

	<i>Patient 1</i>	<i>Patient 2</i>	<i>Patient 3</i>	<i>Patient 4</i>
<i>Sex</i>	Male	Female	Male	Male
<i>Age</i>	26	26	28	23
<i>Weight (kg)</i>	76	65	95	68
<i>Height (m)</i>	1,92	1,74	1,72	1,69

Each session had a duration of approximately one minute. Information such as number of steps taken, force and weight applied will be some of the results that will be compared between each healthy patient, considering the information in Table 8. Figures 7.7 to 7.9 display the results achieved by the four patients during a one-minute session.

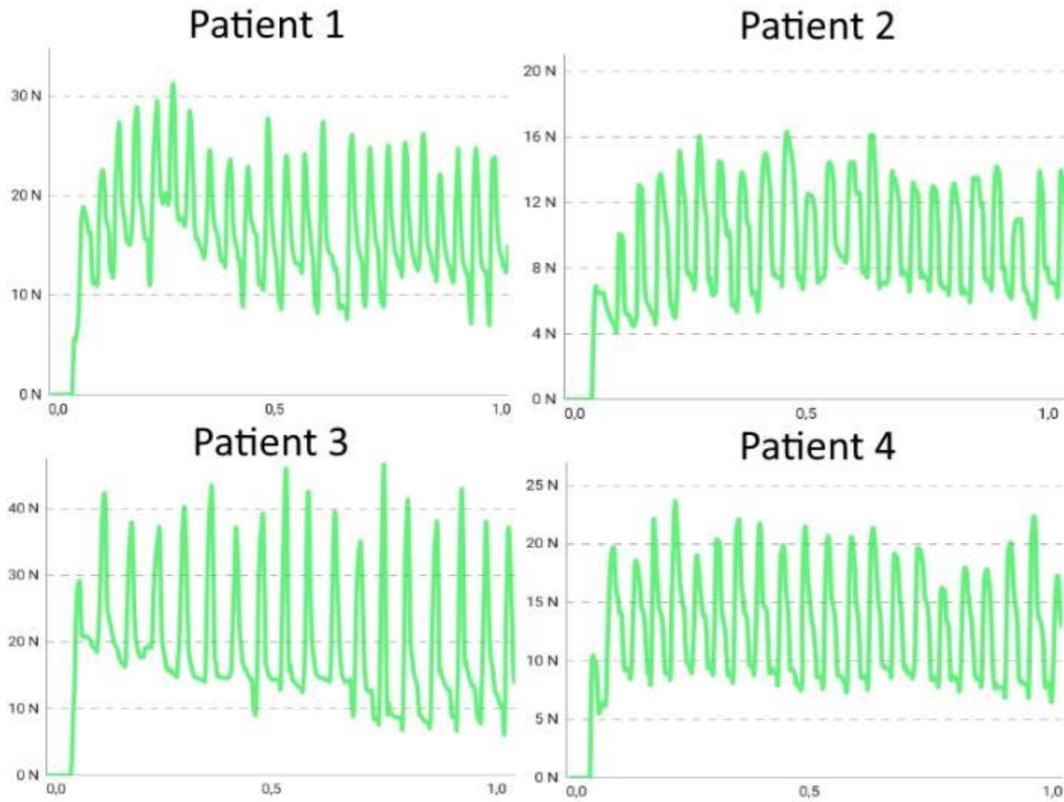


Figure 7.7 – FSR sensor data of a session for each patient

Figure 7.7 indicates the force applied on the hand support by the patients during walking. A pattern can be noted, as the force increases, decreases and increases again afterwards. This is due to the fact that the patients are always applying force to the crutch when holding it, and each time they take a step to move further the force increases because the crutch is being pressed against the floor. Despite the pattern being the same, the values obtained differ from patient to patient, as can be seen detailed in Table 9.

Table 9 – Extra FSR sensor data calculated

	<i>Patient 1</i>	<i>Patient 2</i>	<i>Patient 3</i>	<i>Patient 4</i>
<i>Maximum Force (N)</i>	30,24	16,04	43,77	23,3
<i>Median Force (N)</i>	16,69	9,47	19,75	12,91

The patient who applied most force was Patient 3, while the patient who applied less force was Patient 2. While some people handle crutches differently than others, which is hard to keep track of, weight also plays a big factor in this case. In the cases tested, the patient who weighed more (Patient 3) ended up having a higher median and maximum force, while the patient who weighed less (Patient 2) had the lowest applied force values. Patient 1 was heavier than patient 4, which also resulted in higher force values obtained.

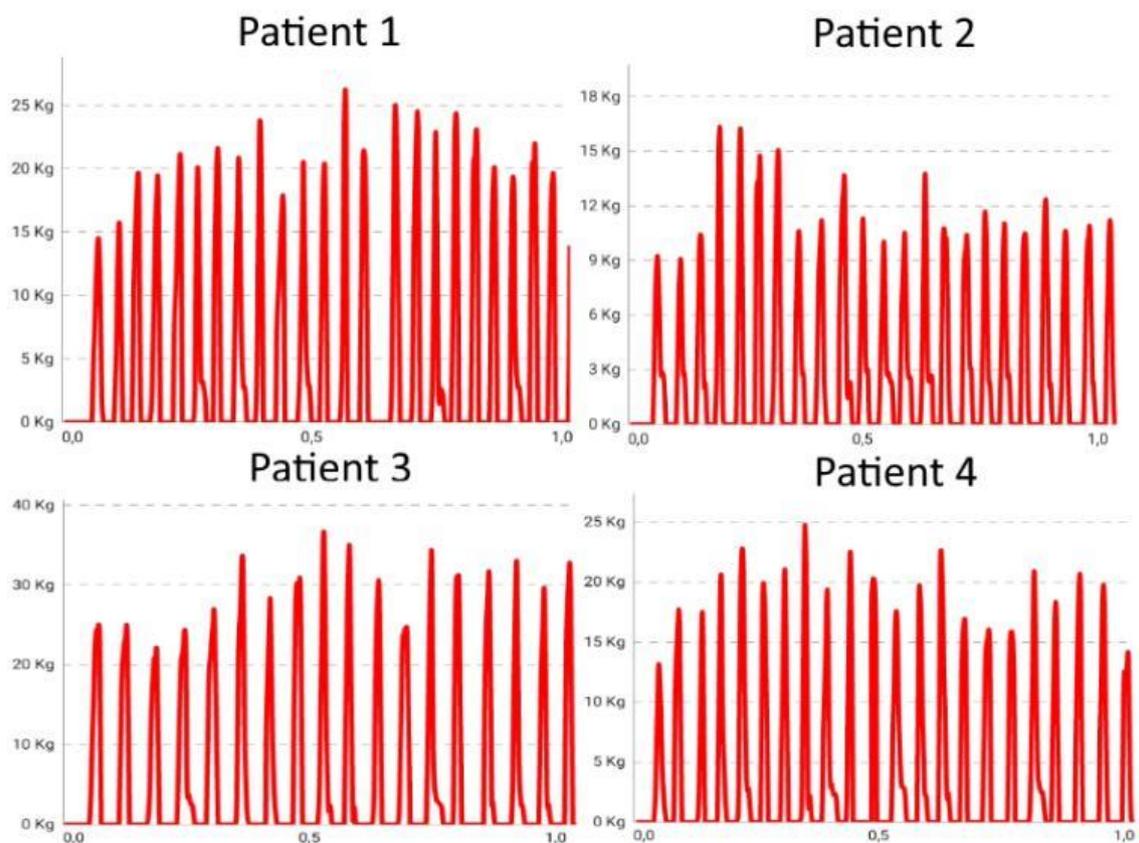


Figure 7.8 – Load cell sensor data of a session for each patient

In Figure 7.8 the data from the load cell regarding the testing session with the four patients was analyzed. All the charts once again provide the same pattern, with some reaching higher values than others, as seen before in Figure 7.7.

Table 10 – Extra load cell sensor data calculated

	<i>Patient 1</i>	<i>Patient 2</i>	<i>Patient 3</i>	<i>Patient 4</i>
<i>Maximum weight (Kg)</i>	23,83	14,76	35,66	22,38
<i>Median weight (Kg)</i>	19,47	10,76	28,23	17,86
<i>Number of steps</i>	23	23	18	22

Table 10 shows additional data regarding the load cell sensor. The pattern of the values is similar to the ones obtained in Table 9, where Patient 3 has the highest values of weight, reaching a maximum weight of 35,66 Kg and a median of 28,23 Kg and Patient 2 has the lowest, only reaching a maximum of 14,76 Kg and a median of 10,74 Kg. Number of steps is also calculated, where Patient 1, 2 and 4 average the same number of steps (23 and 22) while Patient 3 has less steps taken during the session (only 18). This could mean that the patient either has more difficulty walking or he just takes longer to walk.

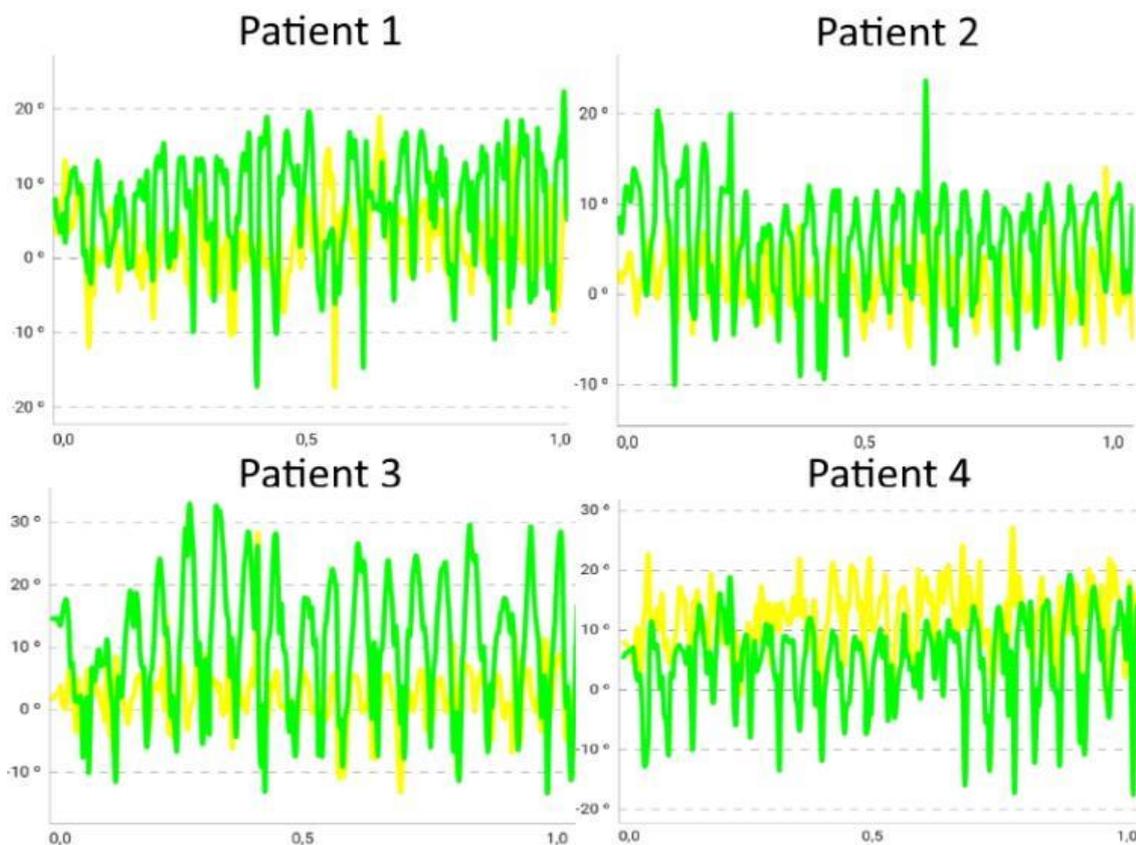


Figure 7.9 – IMU sensor data of a session for each patient

The angle values obtained by the IMU can be seen in Figure 7.9. For a more detailed analysis of the data, we can analyze Table 11.

Table 11 – Extra IMU sensor data calculated

	<i>Patient 1</i>	<i>Patient 2</i>	<i>Patient 3</i>	<i>Patient 4</i>
<i>Maximum roll (right)</i>	18,85 °	13,89 °	28,24 °	27,21 °
<i>Minimum roll (left)</i>	-17,3 °	-6,89 °	-13,19 °	-3,7 °
<i>Median roll</i>	2,49 °	1,46 °	1,82 °	11,92 °
<i>Maximum pitch (forward)</i>	22,22 °	23,66 °	32,78 °	19,32 °
<i>Minimum pitch (backwards)</i>	-16,5 °	-9,59 °	-13,34 °	-17,2 °
<i>Median pitch</i>	6,78 °	5,76 °	10,61 °	4,74 °

Data retrieved from the table indicates several things. A positive roll angle indicates that the volunteers tipped the crutch more to the right side and a negative roll angle indicates that they tipped it more to the left side. Pitch angles follow the same idea, where a positive pitch angle indicates that the volunteer tipped the crutch forwards, which happens when he's taking a step, and a negative pitch angle indicates that the crutch was tipped backwards. After the session, all the patients had positive values of median roll,

ISCTE-IUL

Smart Object for Physical Rehabilitation Assessment

meaning they tend to tip the crutch more to the right side. Values of Patient 1, 2 and 3 are close to zero ($2,49^\circ$, $1,46^\circ$ and $1,82^\circ$, respectively) which means they are generally balanced while walking, but Patient 4 had a value of $11,92^\circ$, indicating that he doesn't maneuver the crutch straight regarding horizontal inclination. Regarding pitch values, all the patients once again had positive values, meaning they tend to tilt the crutch more to the front. This time, Patient 3 had the highest median value of $10,61^\circ$ while Patient 1, 2 and 4 had values of $6,78^\circ$, $5,76^\circ$ and $4,74^\circ$, respectively.

Chapter 8 – Conclusions and Future Work

8.1. Conclusions

In conclusion, we can emphasize that the techniques used in conventional medicine require modernization and improvement. In an area as sensitive as health, it is important that the margin of error is constantly reduced, which can be verified with the implementation of sensors in the various equipment used. With the use of high precision sensors and an IoT system, healthcare has seen a gradual decentralization from the traditional health center-based approach [60].

Knowing that the population that uses physical therapy is not limited to an age group or type of medical condition, the use of sensors will allow progressive and more accurate monitoring on both sides of the patient-physician relationship. It should be noted that one of the advantages associated with this method is the simplicity in which the results are obtained and analyzed, since a large percentage of the population now depends on the Internet for the most basic functions.

With the use of a multimodal IoT system attached to objects of relevance in the field of physiotherapy, data obtained from the force, load and the IMU sensor can be easily visible to the physiotherapist in order to figure out if the patient's development is going on the right path, and if not, change the type of treatment to make sure that it is.

Regarding the initial research questions, the application developed was considered very user friendly and with a brief explanation regarding data analysis, the physiotherapists should have no problems with understanding the data received from the sessions. The system's hardware doesn't obstruct the patient movement at all since very small hardware components were used to make sure the system is as compact as possible. Finally, Bluetooth communication was chosen since it is commonly used for short data transmission.

The prototype system was successful in acquiring, processing and displaying data from the sensors to the mobile application. Data transmission with an interval of 200 milliseconds provided a good flow of data transfer. A bigger value would risk not capturing important data, while a value too small would stress the databases. More systems regarding physical therapy training should be developed in order to have a better monitoring of patient development, resulting in better and faster improvements.

The system also proved to be non-intrusive, relatively cheap, scalable and easy to use. A system with a cost of approximately 150€ (including the crutch) would be very valuable to have in physical therapy clinics which would greatly help physiotherapists to monitor the progress of their patients by using a simple user interface, data charts and patient forms. This system could also be scalable to more than one physical therapy clinic, however, some minor tweaks in the databases and the application's code would be required.

8.2. Future Work

The project had many challenges and as it usually occurs in these lengthy projects, it has many parts that can be improved in the future, such as the following:

- Discuss several features with physiotherapists that could be added to the SmartCrutch system;
- Expand the system to multiple physical therapy clinics, since the current database is only set for one single clinic. Each clinic should have its own database;
- Include more sensors into the system in order to provide more data to the physiotherapists, such as body temperature sensor and heart rate sensor (see Figure 8.1);

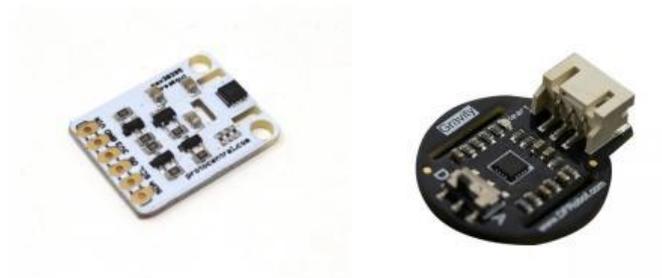


Figure 8.1 – Body temperature sensor MAX30205 (left) and heart rate temperature sensor SEN0203 (right)

- Develop tests and training plans in physical therapy clinics with the help of physiotherapists;
- Modify the system to allow data transmission of two crutches being used simultaneously by the same person;

- MQTT protocol [61] is widely used in IoT systems and should be considered instead of HTTP. An example of the functioning of this protocol is seen in Figure 8.2. An MQTT system consists of clients communicating with a server, often designated as a broker. Information is organized in a hierarchy of topics. A client may either be a publisher (3) or a subscriber (1,2) of information. When a publisher has new data to distribute, it sends a control message with the data connected to the broker. The broker then distributes the data to any clients that were subscribed to that topic (4,5).

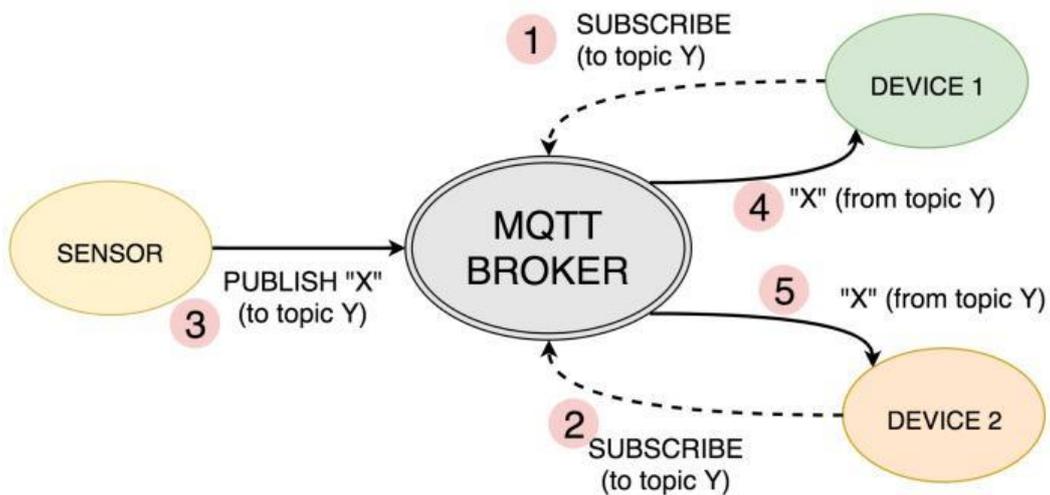


Figure 8.2 – MQTT protocol

References

- 1- "Technology and medicine", *Broughttolife.sciencemuseum.org.uk*, 2018. [Online]. Available:<http://broughttolife.sciencemuseum.org.uk/broughttolife/themes/technologies> [Accessed: 11- Aug- 2018].
- 2- "Physiotherapy & Rehabilitation | Pt Health". *pt Health*. N.p., 2016. Web. 1 May 2018. <https://www.pthealth.ca/service/physiotherapy/>.
- 3- Luca Catarinucci, Danilo de Donno, Luca Mainetti, Luca Palano, Luigi Patrono, Maria Laura Stefanizzi, Luciano Tarricone, "An IoT-Aware Architecture for Smart Healthcare Systems", *IEEE JOURNAL*, vol. 2, no. 6, December 2015.
- 4- R. de Souza, V. C. C. Roza and O. Postolache, "A multi-sensing physical therapy assessment for children with cerebral palsy," *2017 Eleventh International Conference on Sensing Technology (ICST)*, Sydney, NSW, 2017, pp. 1-6.
- 5- "About Physical Therapist (PT) Careers", *Apta.org*, 2018. [Online]. Available: <http://www.apta.org/PTCareers/Overview/>. [Accessed: 20- May- 2018].
- 6- *Forbes.com*, 2018. [Online]. Available: <https://www.forbes.com/pictures/efkk45eifhm/no-9-best-job-physical-therapist/#5e00f31e2d59>. [Accessed: 20- May- 2018].
- 7- "Advancements in PT Tech/Emerging Trends", *WebPT*, 2018. [Online]. Available: <https://www.webpt.com/blog/post/advancements-pt-techemerging-trends>. [Accessed: 20- May- 2018].
- 8- P. S. Malvade, A. K. Joshi and S. P. Madhe, "IoT based monitoring of foot pressure using FSR sensor," *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2017, pp. 0635-0639.
- 9- "Nwave Smart Parking Company", *NWAVE*, 2018. [Online]. Available: <https://www.nwave.io/>. [Accessed: 14- Jul- 2018].
- 10- "Ellis Audiovisual Technology", Ellis, 2018. [Online]. Available: <http://www.ellis-avt.com/what-we-do/home-automation>. [Accessed: 04- Aug- 2018].
- 11- L. Catarinucci *et al.*, "An IoT-Aware Architecture for Smart Healthcare Systems," in *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515-526, Dec. 2015.

- 12- "Number of mobile phone users worldwide 2013-2019 | Statista", *Statista*, 2018. [Online]. Available: <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/>. [Accessed: 20- May- 2018].
- 13- "Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016", *Gartner.com*, 2018. [Online]. Available: <https://www.gartner.com/newsroom/id/3609817>. [Accessed: 20- May- 2018].
- 14- M. S. Uddin, J. B. Alam and S. Banu, "Real time patient monitoring system based on Internet of Things," *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, 2017, pp. 516-521.
- 15- "In Hospital ICUs, AI Could Predict Which Patients Are Likely to Die", *IEEE Spectrum: Technology, Engineering, and Science News*, 2018. [Online]. Available: <https://spectrum.ieee.org/the-human-os/biomedical/diagnostics/in-hospital-intensive-care-units-ai-could-predict-which-patients-are-likely-to-die>. [Accessed: 08- Aug- 2018].
- 16- A. Bustamante, "Home", *Thingier.io*, 2018. [Online]. Available: <https://thingier.io/>. [Accessed: 08- Aug- 2018].
- 17- T. Martins, V. Carvalho and F. Soares, "Development of a system for monitoring and tracking of physiotherapeutic movements in patients with neurological diseases," *2013 IEEE 3rd Portuguese Meeting in Bioengineering (ENBENG)*, Braga, 2013, pp. 1-4.
- 18- J. W. Burke, D. J. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough, "Optimising engagement for stroke rehabilitation using serious games," *Vis Comput*, vol. 25, pp. 1085-1099, August 2009.
- 19- R. C. A. Alves, L. B. Gabriel, B. T. de Oliveira, C. B. Margi and F. C. L. dos Santos, "Assisting Physical (Hydro)Therapy With Wireless Sensors Networks," in *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 113-120, April 2015.
- 20- N. Hegde *et al.*, "The Pediatric SmartShoe: Wearable Sensor System for Ambulatory Monitoring of Physical Activity and Gait," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 2, pp. 477-486, Feb. 2018.

- 21- O. Postolache, "Physical rehabilitation assessment based on smart training equipment and mobile APPs," *2015 E-Health and Bioengineering Conference (EHB)*, Iasi, 2015, pp. 1-6.
- 22- J. Mercado, G. Chu, E. J. Imperial, K. G. Monje, R. M. Pabustan and A. Silverio, "Smart cane: Instrumentation of a quad cane with audio-feedback monitoring system for partial weight-bearing support," *2014 IEEE International Symposium on Bioelectronics and Bioinformatics (IEEE ISBB 2014)*, Chung Li, 2014, pp. 1-4.
- 23- *Play.google.com*, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=com.ninezest.stroke&hl=en>. [Accessed: 03- Sep- 2018].
- 24- *Play.google.com*, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=net.jombalik.shoulderexercise>. [Accessed: 03- Sep- 2018].
- 25- *Play.google.com*, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=com.application.mehab>. [Accessed: 03- Sep- 2018].
- 26- *Play.google.com*, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=com.xfe646e5bcb0.www>. [Accessed: 03- Sep- 2018].
- 27- *Play.google.com*, 2018. [Online]. Available: <https://play.google.com/store/apps/details?id=com.healure.painrecovery>. [Accessed: 03- Sep- 2018].
- 28- "Arduino - Software", *Arduino.cc*, 2018. [Online]. Available: <https://www.arduino.cc/en/Main/Software>. [Accessed: 16- Aug- 2018].
- 29- "Download Android Studio and SDK tools | Android Developers", *Android Developers*, 2018. [Online]. Available: <https://developer.android.com/studio/>. [Accessed: 16- Aug- 2018].
- 30- "Bluetooth pairing Definition from PC Magazine Encyclopedia", *Pcmag.com*, 2018. [Online]. Available: <https://www.pcmag.com/encyclopedia/term/59736/bluetooth-pairing>. [Accessed: 16- Aug- 2018].

- 31- "What is a Thread in an Operating System? - Definition from Techopedia", *Techopedia.com*, 2018. [Online]. Available: <https://www.techopedia.com/definition/27857/thread-operating-systems>. [Accessed: 16- Aug- 2018].
- 32- "What is Socket? - Definition from Techopedia", *Techopedia.com*, 2018. [Online]. Available: <https://www.techopedia.com/definition/16208/socket>. [Accessed: 16- Aug- 2018].
- 33- "Physical Therapists Can Properly Assess & Fit Walking Aids to Prevent Injuries", *Apta.org*, 2018. [Online]. Available: <http://www.apta.org/Media/Releases/Consumer/2009/7/9/>. [Accessed: 27- Aug- 2018].
- 34- "Force Sensing Resistor - How it Works and its Applications", *ElProCus - Electronic Projects for Engineering Students*, 2018. [Online]. Available: <https://www.elprocus.com/force-sensing-resistor-technology/>. [Accessed: 03- Sep- 2018].
- 35- I. Interlink Electronics, "FSR 406", *Interlinkelectronics.com*, 2018. [Online]. Available: <https://www.interlinkelectronics.com/fsr-406>. [Accessed: 03- Sep- 2018].
- 36- "FlexiForce A201 Sensor", *Tekscan*, 2018. [Online]. Available: <https://www.tekscan.com/products-solutions/force-sensors/a201?tab=description>. [Accessed: 03- Sep- 2018].
- 37- "FlexiForce Adapter - 1120 at Phidgets", *Phidgets.com*, 2018. [Online]. Available: <https://www.phidgets.com/?tier=3&catid=3&pcid=4&prodid=91>. [Accessed: 03- Sep- 2018].
- 38- "Pololu - AltIMU-10 v4 Gyro, Accelerometer, Compass, and Altimeter (L3GD20H, LSM303D, and LPS25H Carrier)", *Pololu.com*, 2018. [Online]. Available: <https://www.pololu.com/product/2470>. [Accessed: 03- Sep- 2018].
- 39- H. Ferdinando, H. Khoswanto and D. Purwanto, "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATmega8535," *2012 International Symposium on Innovations in Intelligent Systems and Applications*, Trabzon, 2012, pp. 1-5.
- 40- R. Faragher, "Understand the Basis of the Kalman Filer Via a Simple and Intuitive Derivation", *IEEE Signal Processing Magazine*, September 2012.

- 41- "Wheatstone Bridge Circuit", *ElectronicsTutorials*, N.p., 2018. Web. 1 May 2018.
<https://www.electronics-tutorials.ws/blog/wheatstone-bridge.html/>.
- 42- "SNC2C6 Stainless Steel Tension Compression Button Mini Load Cell, SNC Product Details from Shenzhen Sensor and Control Co., Ltd. on Alibaba.com", *www.alibaba.com*, 2018. [Online]. Available: https://sensor-con.en.alibaba.com/product/60684713799-803398769/SNC2C6_Stainless_Steel_Tension_Compression_Button_Mini_Load_Cell_5kg_50kg_100kg_200kg_300kg_500kg.html. [Accessed: 03- Sep- 2018].
- 43- "Raspberry Pi Zero - Raspberry Pi", *Raspberry Pi*, 2018. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-zero/>. [Accessed: 03- Sep- 2018].
- 44- "Arduino Nano", *Store.arduino.cc*, 2018. [Online]. Available: <https://store.arduino.cc/arduino-nano>. [Accessed: 03- Sep- 2018].
- 45- A. Industries, "Adafruit Bluefruit LE SPI Friend - Bluetooth Low Energy (BLE)", *Adafruit.com*, 2018. [Online]. Available: <https://www.adafruit.com/product/2633>. [Accessed: 03- Sep- 2018].
- 46- *Olimex.com*, 2018. [Online]. Available: <https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf>. [Accessed: 03- Sep- 2018].
- 47- "Smartphone", *En.wikipedia.org*, 2018. [Online]. Available: <https://en.wikipedia.org/wiki/Smartphone>. [Accessed: 06- Sep- 2018].
- 48- "Samsung Galaxy J5 (2016) specs", *Phone Arena*, 2018. [Online]. Available: https://www.phonearena.com/phones/Samsung-Galaxy-J5-2016_id10096. [Accessed: 06- Sep- 2018].
- 49- "Samsung Galaxy Tab S2 8.0-inch specs", *Phone Arena*, 2018. [Online]. Available: https://www.phonearena.com/phones/Samsung-Galaxy-Tab-S2-8.0-inch_id9384. [Accessed: 06- Sep- 2018].
- 50- "Meet Android Studio | Android Developers", *Android Developers*, 2018. [Online]. Available: <https://developer.android.com/studio/intro/>. [Accessed: 06- Sep- 2018].

- 51- "Save data using SQLite | Android Developers", *Android Developers*, 2018. [Online]. Available: <https://developer.android.com/training/data-storage/sqlite>. [Accessed: 06- Sep- 2018].
- 52- "amitshekhariitbhu/Android-Debug-Database", *GitHub*, 2018. [Online]. Available: <https://github.com/amitshekhariitbhu/Android-Debug-Database>. [Accessed: 08- Sep- 2018].
- 53- "MySQL: MySQL Workbench", *Mysql.com*, 2018. [Online]. Available: <https://www.mysql.com/products/workbench/>. [Accessed: 08- Sep- 2018].
- 54- "Base64 | Android Developers", *Android Developers*, 2018. [Online]. Available: <https://developer.android.com/reference/java/util/Base64>. [Accessed: 08- Sep- 2018].
- 55- "phpMyAdmin", *phpMyAdmin*, 2018. [Online]. Available: <https://www.phpmyadmin.net>. [Accessed: 08- Sep- 2018].
- 56- "Apache Axis2 – RESTful Web Services Support", *Axis.apache.org*, 2018. [Online]. Available: <https://axis.apache.org/axis2/java/core/docs/rest-ws.html>. [Accessed: 08- Sep- 2018].
- 57- "XAMPP Installers and Downloads for Apache Friends", *Apachefriends.org*, 2018. [Online]. Available: <https://www.apachefriends.org/index.html>. [Accessed: 08- Sep- 2018].
- 58- "PHP Prepared Statements", *W3schools.com*, 2018. [Online]. Available: https://www.w3schools.com/php/php_mysql_prepared_statements.asp. [Accessed: 17- Sep- 2018].
- 59- "PhilJay/MPAndroidChart", *GitHub*, 2018. [Online]. Available: <https://github.com/PhilJay/MPAndroidChart/wiki>. [Accessed: 04- Mar- 2018].
- 60- N. Kumar, "IoT architecture and system design for healthcare systems," *2017 International Conference on Smart Technologies For Smart Nation (SmartTechCon)*, Bangalore, 2017, pp. 1118-1123.
- 61- M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," *2015 Fifth International Conference on Communication Systems and Network Technologies*, Gwalior, 2015, pp. 746-75.

Annex A – Articles

Article: Mobile Application based on Wireless Sensor Network

This article has been accepted and presented in ISSI 2018 September 7-8, 2018, Shanghai Grand Trust Purple Mountain Hotel, Shanghai, China, International Symposium on Sensing and Instrumentation in IoT Era.



ISSI 2018 1st International Symposium on Sensing and Instrumentation in IoT Era
September 6-7, 2018 - Grand Trustal Purple Mountain Hotel, Shanghai, China

ORGANIZERS

Honorary Chairs
Youfang Huang
Shanghai Maritime University, China
Wei Yan
Shanghai Maritime University, China

General Chairs
Octavian Postolache
ISCTE-IUL & IT, Portugal
Yongsheng Yang
Shanghai Maritime University, China
Xin Wang
Fudan University, China

Technical Program Chairs
Subhas Mukhopadhyay
Macquarie University, Australia
Domenico Capriglione
Università degli Studi di Salerno, Italy
Daqi Zhu
Shanghai Maritime University, China

Publication Chairs
Bin Yang
Shanghai Maritime University, China
Huafeng Wu
Shanghai Maritime University, China
Daofang Chang
Shanghai Maritime University, China

Local Organization Chairs
Chaofeng Li
Shanghai Maritime University, China
Ziyu Lu
Shanghai Pudong Federation of R&D Institutions, China

IMPORTANT DATES

May 11, 2018
Special Session Workshop & Proposals Deadline

June 1, 2018
Paper Submission Deadline

June 25, 2018
Paper Acceptance Notification Deadline

July 15, 2018
Camera-ready Submission & Early Registration Deadline

Call for Papers

ISSI 2018 represents a forum where researchers, scientists, engineers and practitioners around the world will promote the advances in sensing and instrumentation for IoT. The symposium valorizes new and original research in IoT applications fields, such as smart city, transportation, healthcare and farming.

The full version papers (4 to 6 pages) submitted to ISSI 2018 will be accepted based on peer review process. The review process may conduct to some revision requests for the final version of the paper that must be mandatory fulfilled to have the paper accepted for the symposium proceedings.

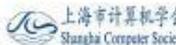
The accepted papers will be published in conference proceedings and will appear in **IEEE Xplore** and **EI Compendex**. Selected papers will be considered for publication in special issue of **Sensors**, **SCI open Journal**.

Topics of the symposium will include, but are not limited to:

- Smart Sensors and Wireless Sensor Networks for IoT;
- Test and Automated Instrumentation for IoT;
- Localization, Algorithms for Smart Sensor Network;
- Software Platforms and Middleware in Smart IoT Systems;
- IoT in Retail Logistics;
- IoT in Supply Chain Finance and Management;
- Computer Vision and Application in IoT;
- IoT & Wearable Solutions for Healthcare;
- IoT System Design Methodologies;
- Big Data Processing in IoT Systems;
- Energy Harvesting and Scavenging for Wireless Sensors Networks and IoT;
- Education and Training for Networked Instrumentation and Measurement;
- Standards for IoT and IoT Security;
- Sensors and IoT for Smart Ports and Logistics;
- Industrial Internet of Things;
- Sensors and IoT for Structural and Infrastructural Health Monitoring;
- IoT for Healthcare;
- IoT Applications in Smart Cities;
- IoT Applications for Smart Home;
- IoT in Smart Farming;

Many other new initiatives and opportunities to encourage your active participation in the conference are planned, that will make ISSI 2018 a vibrant event to meet with people in sensing and instrumentation field.

Visit the conference website for each specific call and additional news:
<http://issi2018.csp.escience.cn>



Mobile Application based on Wireless Sensor Network for Physical Rehabilitation

Pedro Martim V. L. Frango
ISCTE-IUL
Lisbon, Portugal
martimvalente@hotmail.com

Octavian Adrian Postolache
Instituto de Telecomunicações/ISCTE-IUL
Lisbon, Portugal
opostolache@lx.it.pt

Abstract—The technologies associated with smart healthcare are a reality nowadays, however in the physical therapy area there is still lack of patient monitoring during the physical rehabilitation and common usage of walking aids by the patients affected by lower limb impairments. The paper presents a solution to this problem by relying on smart equipment used in physical rehabilitation, more precisely a crutch. Thus, multiple smart sensors are embedded on crutches which will provide to physiotherapist appropriate information regarding the interaction between the patient and the walking aids through a mobile application developed for Android systems.

Keywords—physical therapy, wireless sensor network, Internet of Things, mobile application, monitoring

I. INTRODUCTION

Physical rehabilitation is one of many areas in the health department that has had more demand recently due to population aging, car accidents, sport injuries, among other causes, which leads to a need to optimize the rehabilitation process according with patient evolution.

Currently there is a lack of sensor implementation in the physical therapy area. With the use of smart objects the physiotherapist should be able to get a more appropriate reading of the patient's session which would result in a better and more appropriate treatment [1-2].

The incentive to work for a professional is vital in any profession and according to *Forbes* magazine, physical therapists were ranked as having one of "The Ten Happiest Jobs" according to articles published in 2011 and 2013, while an expected job growth between 2010 and 2020 was an astonishing 39% [3].

Physical therapy's innovations in technology are a work in progress and they offer new opportunities for improved treatment and patient connection. Advances in robotics and bionics provide physiotherapists with better decision making, therefore leading to an improved outcome and patient experience [4]. Analyzing the data received from the system is challenging because different factors must be taken into account, like patient's age, sex, walking style, walking impairments, among others [5].

Recent advances in Internet of Things (IoT) technologies are providing a development of smart systems with the main purpose of improving health care. Some examples are

automatic identification and tracking of people and biomedical devices, correct drug-patient associations, and patient monitoring [6]. By designing a portable low cost and non intrusive IoT system capable of monitoring the patient's sessions, it's possible to analyze certain data that is invisible to the naked eye, like gait patterns.

In this work a smart crutch prototype was considered. Several sensors were attached to it which provides static and dynamic information about aided gait, information that is processed and presented to the physiotherapist through a mobile application regarding the patient's session.

Mobile devices are used on a daily basis and in 2016 an estimated 62,9% of the population worldwide owned a mobile phone [7]. In the 4th Quarter of 2016, the number of phones sold with Android system was 352 million reaching a 81,7% market share, while iOS system phones only sold 77 million phones which resulted in a 17,9% market share [8].

With the data above in mind, an Android application was developed that supports both mobile phones and tablets. In order to start a patient's session the mobile device is required to have Bluetooth connection for it to be able to connect to the smart crutch; however, it's not required if the physiotherapist simply wants to view the patient's previous data sessions.

In the last decade in the field of physiotherapy, the goal has been to develop new systems to make clinical decisions efficiently and rapidly, as can be seen in "*Smart Walker Solutions for Physical Rehabilitation*", by O. Postolache, [9] which focuses on improving quality of life for patients that need physical rehabilitation with the use of SmartWalkers. These walkers measured quantities such as force, acceleration, velocity and heart rate during a training session with the use of sensors and signal conditioning circuits embedded on the walker which provided the physiotherapist visual information regarding the user's balance and gait pattern, transforming a regular walker into a smart walker.

The system used in O. Postolache paper [9] was based in force sensing resistors and motion sensors, providing visible information to the physiotherapist through a mobile or web application through the use of wireless communication, such as Wi-Fi and Bluetooth. Data received from the client was stored in a remote database in order to evaluate the rehabilitation sessions.

II. SYSTEM DESCRIPTION

This paper focuses on a SmartCrutch System (Fig.1) where data processed by the sensors is done by the Arduino IDE, which uses C/C++ language. To reach actual values and not raw values, several libraries and mathematical algorithms were used.

A multimodal IoT system will be used to provide proper data transfer between the smart crutch and the mobile device. An Arduino Nano will receive and process the data received from the sensors attached to the walker, such as force applied by the patient on the crutch's hand support and weight applied by the base of the walker in contact with the floor. The crutch's orientation and inclination is also measured using an Inertial Measurement Unit (IMU). The IMU is used to measure the crutch's orientation, vertical and horizontal inclination (also known as Euler Angles, yaw, pitch and roll).

Data from sensors will be transmitted to the mobile device via Bluetooth using a Bluetooth adapter which will also be connected to the Arduino, providing the physiotherapist with real time data analysis. Small LED's will also be attached to the walker to alert the physiotherapist if one or more of the sensor values received are above the set threshold.



Fig. 1. SmartCrutch System

Different types of sensors will be applied in order to retrieve important information regarding the use of the crutch by the patient. Force sensors in the crutch's hand support, a load cell sensor on the base of the crutch which will measure the weight applied by the patient into the crutch and an IMU that will give us orientation and inclination angle.

A SmartCrutch prototype can be seen in Fig. 2 where all the components are connected to a PCB (Printed Circuit Board) that has an Arduino Nano attached to it, which will be powered by two small 3.7 V batteries connected in serial.

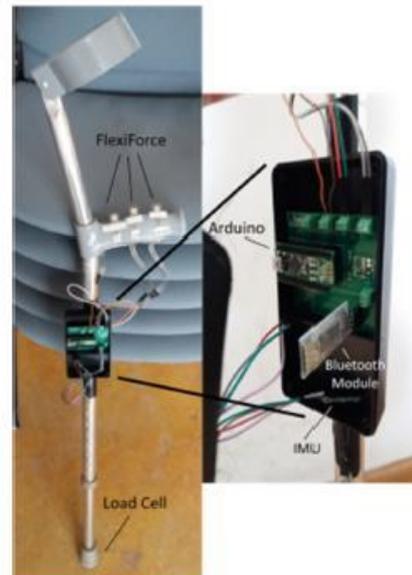


Fig. 2. SmartCrutch prototype

A. Sensors and Conditioning Circuits

A small and piezoresistive [10] FlexiForce A201 sensor was used with the purpose of measuring forces up to 445 Newton. Since the area of contact of this sensor is relatively small, three sensors were placed in the crutch's hand support to make sure that it covers most of the hand support area. They will provide the physiotherapist with an idea of the force applied by the patient's hand.

Calibration is required on these types of sensors to make sure that the results acquired are accurate. In order to calibrate the sensors, a wide range of known weights were applied to the sensing area of each individual FlexiForce sensor and the resulting voltage (V) of each weight was noted. Finally, a linear interpolation between weights and voltage was done, which can be seen in Fig.3.

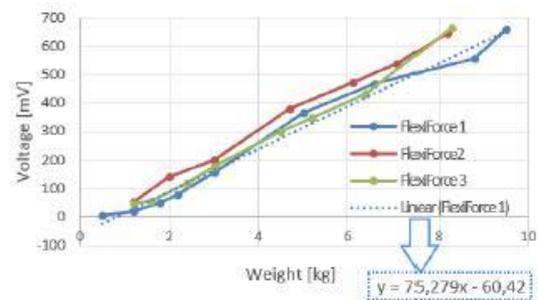


Fig. 3. Calibration curves for FlexiForce sensors

It's possible to determine a linear equation after applying the linear interpolation. Fig.3 also shows an example of a linear equation regarding one of the FlexiForce sensors. Using the equation (1):

$$y = mx + b \quad (1)$$

with m and b being provided by the linear interpolation, simply replacing y with the voltage measured from the sensors and solving in order to x will result on the weight applied.

To determine force (F) in Newton we simply apply the equation in (2):

$$F = m * g \quad (2)$$

where m represents the weight measured and g represents the gravitational acceleration.

An Inertial Measurement Unit AltIMU10 v-4 was used. It contains a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, resulting in a total of 9 degrees of freedom. This sensor will be used to determine the crutch's orientation and its horizontal and vertical inclination also known as pitch and roll angle, which is shown in Fig.4, providing information to the physiotherapist on how the crutch is maneuvered by the patient. Equation (3) is used to calculate the yaw angle:

$$yaw = atan2(m_y / m_x) \quad (3)$$

The function $atan2$ calculates the arctangent in the four quadrants and m_y and m_x indicate the horizontal components of the magnetic field sensed by the magnetometer. Pitch angle can be determined by (4):

$$pitch = asin(-a_x / g) \quad (4)$$

where $asin$ represents the inverse of the $sine$ function, a_x represents the acceleration given by the accelerometer in the x-axis and g represents the gravitational acceleration. Finally roll angle can be determined by (5) where a_y and a_z indicate the acceleration in the y-axis and z-axis, respectively.

$$roll = atan2(a_y / a_z) \quad (5)$$

This sensor uses a I²C interface and has several sensitivity values that can be defined by the user. Different values for sensitivity have to be set for each type of sensor. Gyroscope sensitivity can be set at ± 245 , ± 500 or ± 2000 degrees per second, while accelerometer sensitivity values can be set at ± 2 , ± 4 , ± 6 , ± 8 or ± 16 g and finally for the magnetometer values can be set at ± 2 , ± 4 , ± 8 or ± 12 gauss. The lowest sensitivity values were used to provide a better sensing range that fits our project.

The gyroscope can be used to track rotation on a short time accurately, while the magnetometer and the accelerometer can help compensate the gyroscope's drift over time by providing an absolute frame of reference. For accurate reading of the walker's orientation values, sensor fusion will be necessary, so we will be using a Kalman Filter [11].

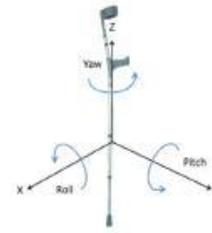


Fig. 4. Crutch representing Euler angles

To measure floor reaction during the crutch's usage a load cell sensor was embedded on the crutch. This type of sensor can measure up to 50 kg and it creates an electrical signal proportional to the weight that is being measured and they are commonly used in the fields of medicine, robotics, and industrial, among many others. The calibration process is the same one used in the FlexiForce sensors, but the final values are displayed in weight and not force. This sensor uses a Wheatstone bridge [12] configuration and without a signal conditioner it provides a low level of voltage output which would result in less accurate values. The conditioning circuit used is based on INA122 instrumentation amplifier, with $G=200$, the reached output voltage for maximum applied weight of 32.5 kg being 1.2 V. The conditioning circuit used is displayed in Fig. 5.

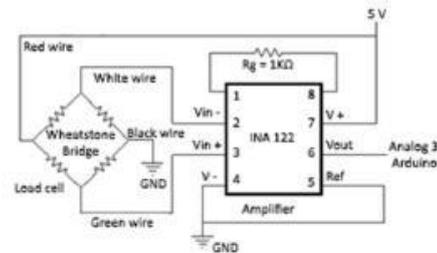


Fig. 5. Load cell conditioning circuit

B. Wireless Communication

Considering the Android OS computation platform wireless communication capabilities a Bluetooth module HC-06 was used for wireless communication between the Arduino and the mobile device. After the connection is established based on implemented embedded software implemented on the level of microcontroller (Atmel MEGA 328P) the signals acquired from the measurement channels and primary processed are transmitted to Android OS device and visualized by physiotherapist. Data analysis component will be done on smart phone level where the values of the previously processed signals will be visible in plot graphs, with the use of *MPAndroidChart* library. Several tests proves the considered Bluetooth communication and chosen transceiver provides optimal communication coverage taking into account the distance between the smart crutches and the smart phone or tablet of the physiotherapist during the training session.

III. MOBILE SOFTWARE

The mobile application was developed with Android Studio IDE, a free software that supports development for Android OS platforms. This application supports phone layouts, which can be used for instance to record a quick session, and also supports tablet layouts which provide a bigger screen, resulting in a better view to analyze data from previous sessions.

The developed APPs is a native application one, offering the possibility of a full offline mode, meaning connection to the Internet is not needed to start and store a new session or view previous sessions.

A local SQLite database provides the data storage of each physiotherapist's related patients and training sessions. If the physiotherapist decides to use a different mobile device than he's used to, it's possible to synchronize data between the local database, which would be empty, and a remote MySQL database located in the Cloud simply by turning on the Internet on the mobile device, which would start the synchronization of the local SQLite database with the remote MySQL database. This is possible based on developed PHP scripts that are in charge of handling the communication and synchronization between the local and the remote database.

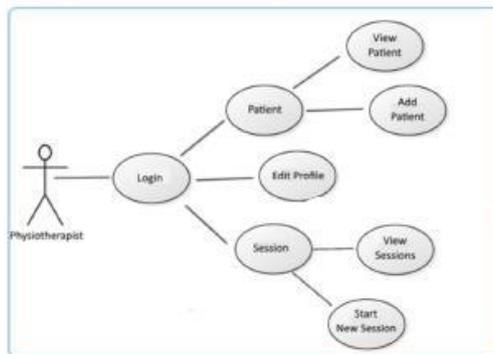


Fig. 6. Actions available to the physiotherapist in the mobile application

For each physiotherapist a registration is required associated with the implemented login function. After registration patient registration is carried out including elements of patient electronic health record. In Fig. 6 multiple actions in the mobile device available to the physiotherapist can be observed. After logging in successfully the physiotherapist is able to view and edit his own profile, as well as adding new patients and view existing patient profiles. Other available actions include starting a new session and view any session previously done.

A patient form should be displayed as can be seen in Fig. 7 where their personal data is visible to the physiotherapist. Instead of having to file a patient file on a piece of paper, patient information can be stored and accessed easily on the application.

Before starting a training session the physiotherapist should perform the session configuration including the patient ID and smart equipment ID (crutches ID). After pairing, data is transmitted to the mobile device and visualized in real time by physiotherapist that may analyze the evolution of the patient during the training. Data processing is needed on the application level when receiving values from the Arduino, which will be displayed in plot charts and stored accordingly in a local SQLite database.

After a session has ended, the physiotherapist can name the session and can also write notes regarding the session that could be useful in the future for a better and more precise treatment.



Fig. 7. Patient form in the mobile application

IV. RESULTS AND DISCUSSION

Each patient's training session is presented in a list by session ID, session date and session duration. Results are visible in time charts where each chart indicates different types of values, such as force applied, weight and Euler angles from the Flexiforce, load cell and IMU sensors, respectively.

If the physiotherapist desires it's possible to change the background color of the plot chart from transparent to white for a different perspective view by toggling a switch button located above each plot. By clicking the chart data a small grey square shows up indicating the value and time of the data clicked, which can also be seen in the figures below.

Several healthy volunteers with different ages and weights tested the system. In the following figures data from two volunteer sessions can be seen. Charts on the left side indicate volunteer number 1 and charts on the right side, with the white background, indicate volunteer number 2. It's also important to note that volunteer number 2 was older and also weighed more than volunteer number 1.

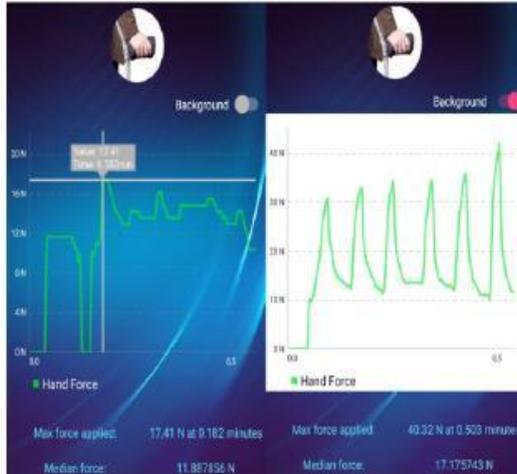


Fig. 8. Session displaying FlexiForce data for two volunteers

Fig. 8 shows the force applied on the hand support by the volunteers during walking. The difference between the applied forces is clearly defined, thus the volunteer 2 is applying more force on the crutch while walking, resulting in a maximum force of 40.32 N while volunteer 1 only apply a maximum force of 17.41 N. The first volunteer removed his hand from the hand support briefly in the beginning of the session in order to adapt his hand better to the hand support, which can be seen at the start of the left chart when the value drops to zero.

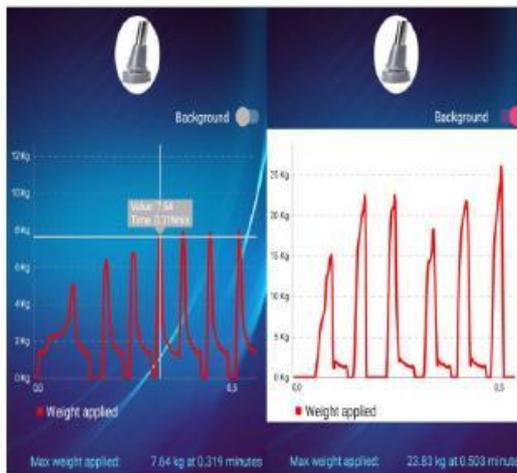


Fig. 9. Session displaying load cell data for two volunteers

In Fig. 9 the data from the load cell sensor was acquired and analyzed. Both charts show a similar pattern, however, the values are higher again for volunteer number 2, with values

reaching up to 23.83 kg. This is an indicator that this volunteer applies more weight to the crutch while walking. With further analysis, by visualizing the peaks of the chart, it's possible to note that the first volunteer did 7 steps with the crutch, which is one more than volunteer number 2 did.



Fig. 10. Session displaying IMU data for two volunteers

Fig.10 shows the angles acquired by the IMU. A positive roll angle indicates that the volunteers tipped the crutch more to the right side and a negative roll angle indicates that they tipped it more to the left side. Pitch angles follow the same idea, where a positive pitch angle indicates that the volunteer tipped the crutch forwards, which happens when he's taking a step, and a negative pitch angle indicates that the crutch was tipped backwards.

Data from both volunteers is very similar with the main difference being that volunteer number 2 tends to tip more the crutch when moving to the left side, which can be seen by the difference of lowest roll angles in both volunteers' data.

V. CONCLUSION AND FUTURE WORK

By using a multimodal IoT non invasive system attached to an object used in physical therapy, such as a crutch, data obtained, such as force, acceleration and gate pattern can be easily analyzed in order for the physiotherapist to understand if the patient is progressing in a positive way, and if not, change the way of treatment.

With the use of sensors in a IoT system it's possible to have a more precise monitoring over the patient where the data is acquired by the sensors, processed by a microcontroller, transmitted using wireless communication protocols and then visualized by the user on a mobile application. The mobile application will also be in charge of data processing in order for the physiotherapist to understand the data that was received, being able to store it in a local SQLite database for further analysis and also storing it remotely on the Cloud when Internet is available to the mobile device.

In the future more sensors could be added to the system that would provide the physiotherapist with more information regarding the session, such as heart rate sensors and body temperature sensors. Another goal would be to use Big Data analytics in order to examine and process all the data acquired in each patient session, resulting in new application elements such as prediction models. As for communication protocol, MQTT is widely used in IoT systems [13] so it should be considered instead of HTTP.

ACKNOWLEDGMENT

This research is supported by Instituto de Telecomunicações (IT-IUL) at ISCTE-IUL, Lisbon, Portugal and Fundação para a Ciência e Tecnologia (FCT) project PTDC/DTP-DES/6776/2014.

REFERENCES

- [1] R. de Souza, V. C. C. Roza and O. Postolache, "A multi-sensing physical therapy assessment for children with cerebral palsy," *2017 Eleventh International Conference on Sensing Technology (ICST)*, Sydney, NSW, 2017, pp. 1-6.
- [2] "About Physical Therapist (PT) Careers", *Apta.org*, 2018. [Online]. Available: <http://www.apta.org/PTCareers/Overview/>. [Accessed: 20-May-2018].
- [3] *Forbes.com*, 2018. [Online]. Available: <https://www.forbes.com/pictures/efkk45eifhm/no-9-best-job-physical-therapist/#5e00f31e2d59>. [Accessed: 20-May-2018].
- [4] "Advancements in PT Tech/Emerging Trends", *WebPT*, 2018. [Online]. Available: <https://www.webpt.com/blog/post/advancements-pt-techemerging-trends>. [Accessed: 20-May-2018].
- [5] P. S. Malvade, A. K. Joshi and S. P. Madhe, "IoT based monitoring of foot pressure using FSR sensor," *2017 International Conference on Communication and Signal Processing (ICCSPP)*, Chennai, 2017, pp. 0635-0639.
- [6] L. Catarinucci *et al.*, "An IoT-Aware Architecture for Smart Healthcare Systems," in *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515-526, Dec. 2015.
- [7] "Number of mobile phone users worldwide 2013-2019 | Statista", *Statista*, 2018. [Online]. Available: <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/>. [Accessed: 20-May-2018].
- [8] "Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016", *Gartner.com*, 2018. [Online]. Available: <https://www.gartner.com/newsroom/vid/3609817>. [Accessed: 20-May-2018].
- [9] O. Postolache *et al.*, "Smart walker solutions for physical rehabilitation," in *IEEE Instrumentation & Measurement Magazine*, vol. 18, no. 5, pp. 21-30, October 2015.
- [10] A. Nallathambi, T. Shanmuganantham, D. Sindhanaiselvi, "Design and Analysis of MEMS based Piezoresistive Pressure sensor for Sensitivity Enhancement", *Materials Today: Proceedings*, Volume 5, Issue 1, Part 1, 2018, Pages 1897-1903.
- [11] H. Ferdinando, H. Khoswanto and D. Purwanto, "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATmega8535," *2012 International Symposium on Innovations in Intelligent Systems and Applications*, Trabzon, 2012, pp. 1-5.
- [12] "Wheatstone Bridge Circuit", *ElectronicsTutorials*, 2018. [Online]. Available: <https://www.electronics-tutorials.ws/blog/wheatstone-bridge.html>. [Accessed: 29-May-2018].
- [13] Mqtt.org. (n.d.). *MQTT*. [online] Available at: <http://mqtt.org/> [Accessed 29 May 2018].

Article: **Smart Object for Physical Rehabilitation**

This article has been accepted and presented in EPE 2018 October 18-19, 2018, Faculty of Electrical Engineering, Iași, Romania, International Conference and Exposition on Electrical and Power Engineering.



10th ANNIVERSARY EPE 2018
10th INTERNATIONAL CONFERENCE AND EXPOSITION ON ELECTRICAL AND POWER ENGINEERING
October 18-19, 2018
IASI, ROMANIA
"Gheorghe Asachi" Technical University of Iasi
Faculty of Electrical Engineering

On behalf of the Program Committee, it is a honour and pleasure to invite you to attend the 10th International Conference and Exposition on Electrical and Power Engineering (EPE 2018) to be held in Iași, Romania, on October 18-19, 2018. Organized by the Faculty of Electrical Engineering of Iasi and SETIS Association, the EPE Conference is now a tradition, being confirmed as an important international event in the electrical engineering area. It started in 1999 with the 1st edition. It is organized every two years with the intent of attracting a wide national and international audience from both academic and industrial communities. Contributions from all research communities working in the field of electrical engineering or in appropriate fields are welcomed to join our conference.

Symposium Chair
Marinel TEMNEANU
Technical Program Chairs
Cristian FOSALAU
Cristian Gyozo HABA
Organizing Committee Chairs
Alexandru SALCEANU
Dorin Dumitru LUCACHE
Publication Chair
Mihai GAVRILAS

Power electronics and electrical drives

Education in electrical and power engineering

Electromagnetic field and electrical circuits

Emerging technologies for electrotechnical materials

Quality, reliability and safety

Important dates

Extended abstract submission	April 16, 2018
Abstract notification	May 4, 2018
Full paper submission	June 15, 2018
Full paper notification	July 15, 2018
Camera ready submission	August 25, 2018
Registration and fee payment	September 15, 2018

The conference is technically co-sponsored by IEEE Romanian Section and is included in the IEEE Conference database with record number #43946. All effectively presented papers that fulfill the Conference requirements will be included in the IEEE Xplore database.

Electrical apparatus

Renewable energy

Automation and robotics

Power systems

Industry applications

Metrology and measurement systems

International Workshop on Advances in Rehabilitation Engineering Applications

3rd Workshop on Lighting: From Energy Efficiency to Light Pollution

Workshop on Internet of Things

International Workshop with Industry on Electromagnetic Properties of Materials and Dedicated Applications

Electrical machines

Computer science applications in electrical engineering

5th International Workshop on Electromagnetic Compatibility and Engineering in Medicine and Biology

Automotive applications

www.epe.tuiasi.ro/2018

Full fee, non-IEEE members	250 EUR	Includes attendance at all scientific programs, Proceedings in CD/DVD form, gala dinner, lunches and coffee breaks.
Full fee IEEE members	200 EUR	
Full fee PhD Students	125 EUR	
Full fee IEEE PhD Students	100 EUR	
Accompanying person	100 EUR	Includes gala dinner, lunches and coffee breaks.

Contact
Prof. Cristian Fosalau
epe@tuiasi.ro
cfosalau@tuiasi.ro
www.epe.tuiasi.ro/2018

Smart Object for Physical Rehabilitation Assessment

Pedro Martim V. L. Frango
ISCTE - IUL
Lisbon, Portugal
martimvalente@hotmail.com

Prof. Octavian Postolache
Instituto de Telecomunicações, ISCTE - IUL
Lisbon, Portugal
opostolache@lx.it.pt

Abstract—This paper presents a design and implementation of a novel system applied on physical rehabilitation monitoring and walking aid usage monitoring. Currently there are fewer systems that provide the patient monitoring during the rehabilitation process by physiotherapists, which may lead to less adequate diagnostic techniques for the patient's physical condition. With the addition of sensors in a equipment that is used by the users during physical therapy, such crutches it is possible to continuously monitor the evolution of the gait balance making correlation with the outcome of the rehabilitation outcomes. System description, elements of software implementation and preliminary results are included in the paper.

Keywords—Internet of Things, physical therapy, wireless sensor network, cloud computing, monitoring, mobile application, smart crutch

I. INTRODUCTION

Nowadays, population aging, stroke events that affect an increased number of population that requires physical therapy which makes the costs to increase in the last decade, with new technologic solutions being studied in order to optimize the rehabilitation process. Improving the performance in the healthcare field is a big obstacle in today's era, where some of the main issues are dealing with nursing staff availability and treatment costs [1-2]. Another obstacle is developing systems that help physiotherapists to make decisions quickly and effectively regarding patient treatment, ensuring they get the best possible treatment [3].

Currently physical therapy training is imposed by physician and its applied using different specific rehabilitation equipments that will help the patients to assure balance during gait activity. By mounting sensors on traditional walking aids it will be possible to obtain the evolution of dynamic and kinematic parameters associated with user gait. The use of smart equipment with communication capabilities and software developed associated with electronic health records will allow data analysis regarding physical rehabilitation progress [4].

The implementation of sensors is growing everyday in the most diverse areas, one of them being physical therapy. A paper related to this topic was written by O. Postolache, "Physical Rehabilitation Assessment based on Smart Training Equipment and Mobile Apps" [5] which focuses on smart walkers and crutches for physical therapy monitoring. Mobile devices will

be used for visualization of signals given by multiple sensors located in the smart training equipment.

Different types of walkers were considered: a regular walker, a two wheel walker and a four wheel walker. Each type of walker is associated with a specific type of sensors that will provide certain data, such as force applied, 3D acceleration and motion capture. Finally, communication used in the system is completely Wireless, resorting to the use of Bluetooth and ZigBee, sending the session's data through the Internet to a remote's server database.

In our article, we also focus on the development of a multimodal IoT (Internet of Things) that includes multiple sensors embedded in crutch, an equipment commonly used by the users during gait rehabilitation period. These sensors are connected to an Arduino Nano and provide the values such as, force applied by the patient on the crutch's handle and force applied to the floor. The orientation of the crutch during gait is also measured using and IMU. The IMU is used to measure orientation, vertical and horizontal inclination (through Euler angles [6]: yaw, pitch and roll).

The physiotherapist, using an Android phone/tablet and a developed application, will be able to have live data visualization through the use of a Bluetooth adapter which will also be connected to the Arduino. This data will be stored in the application's database making it possible to visualize the data of all previously performed sessions regarding the patients of the physiotherapist.

Each physiotherapist should have its own account in the application. To avoid the loss of local data it is possible to synchronize the phone/tablet's application database, SQLite, with a remote MySQL database located in the Cloud with the use of PHP scripts.

II. SYSTEM DESCRIPTION

The main purpose of this paper is to focus on SmartCrutch System (Fig.1). Data processed by the sensors is done by the Arduino IDE, which uses C/C++ language. Libraries and mathematical algorithms were used to determine the final visible data. Bluetooth communication with the mobile device is possible by attaching a Bluetooth module to the Arduino.



Fig. 1. SmartCrutch system

Different types of sensors will be applied in order to retrieve important information regarding the use of the crutch by the patient. Force sensors in the crutch's hand support, a load cell sensor on the base of the crutch which will measure the weight applied by the patient into the crutch and an IMU that will give us orientation and inclination angles. The Arduino will be powered by a 5V power bank that will also be attached to the crutch. A SmartCrutch prototype can be seen in Fig.2, which also includes a PCB board with many preset connections to minimize the use of wires in the system.

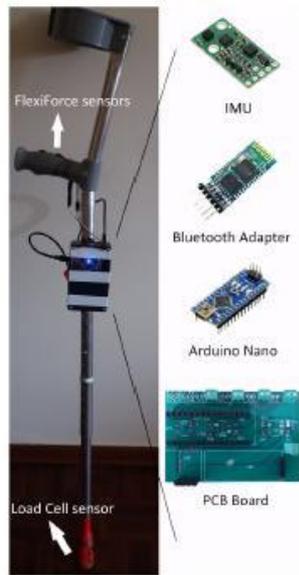


Fig. 2. SmartCrutch prototype

A. Force sensor - FlexiForce A201

This small and versatile piezoresistive [7] sensor is designed to measure forces up to 445 Newton. These sensors will be placed in the crutch's hand support and will provide the physiotherapist with an idea of the force applied by the patient in the hand support. Resistance in the sensor will be close to infinite when no pressure is applied to the sensor. The sensor's resistance decreases exponentially as the load increases as can be seen in Fig.3. To properly calibrate the sensors, linear interpolation was used.

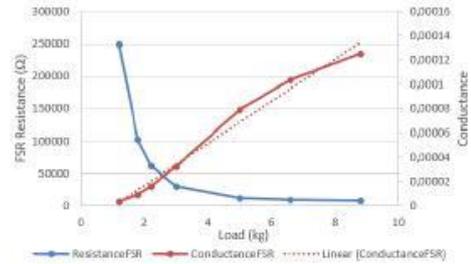


Fig. 3. FlexiForce calibration plot

The above curve was plotted having in account (1), where voltage is proportional to the inverse of the FSR resistance,

$$V_0 = V_{CC} * (R / R + R_{FSR}) \quad (1)$$

where V_0 is the voltage returned by the analog output in the Arduino, V_{CC} equals to 5V and R is the pull-down resistor used with the value of $1M\Omega$. In order to get the resistance value of the FSR we simplified the above equation and resulted in (2).

$$R_{FSR} = ((V_{CC} - V_0) / V_0) * R \quad (2)$$

Load-resistance curve is exponential as can be seen above in Fig. 3, however load-conductance curve is linear, which makes is easier for calibration purposes. Conductance (C) can be easily calculated with (3). Finally, conversion from weight to force is simply done by applying Newton's Second Law.

$$C = 1 / R_{FSR} \quad (3)$$

B. Inertial Measurement Unit

The used Inertial Measurement Unit, the AltIMU 10 v-4 contains a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, which therefore results in having 9 degrees of freedom. It will be used to determine the crutch's orientation as well as its pitch and roll angle, providing the physiotherapist with data on how the crutch is maneuvered by the patient.

This sensor, unlike the force sensors, uses a I2C interface and has available a couple of sensitivity values that can be defined by the user and unlike the other sensors used, calibration can be done via an Arduino library, resulting in a much easier calibration process. The operating voltage of the sensor ranges from 2.5V to 5.5V, so we decided to power it with 3.3V that the Arduino Nano can supply.

The gyroscope is used to track rotation with precision on a short time frame, while the magnetometer and the accelerometer are used to help compensate the gyroscope's drift over time by providing an absolute frame of reference. For accurate reading of values, sensor fusion will be necessary, so we will be using a Kalman Filter [8].

C. Load Cell

Load cell sensors are often used to measure force or weight applied while using a Wheatstone Bridge configuration [9], which makes it a good choice for measuring weight that the person applies onto the crutch's base. The sensor used was able to measure weights up to 50kg. However, it provides very little

output by itself, in the order of around a maximum of 10mV, so by coupling it with an instrumentation amplifier we are able to get a good measurement range. The INA122 (Fig. 4) is a precision instrumentation amplifier for accurate, low noise signal acquisition, which makes it ideal for this case.

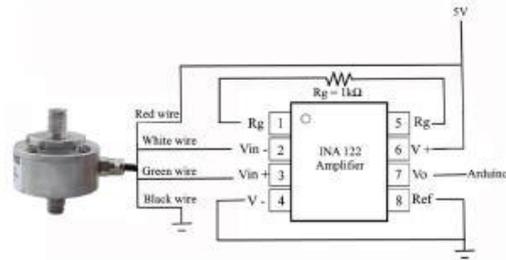


Fig. 4. Instrumentation amplifier INA122 connections

With the value of $R_g=1k\Omega$ the amplifier can provide a gain of $G=205$ according to (4), which is enough to have a good range of measures for calibration purposes.

$$G = 5 + (200k\Omega/R_g) \quad (4)$$

D. Bluetooth communication module – HC-06

Communication is done through the use of a Bluetooth module that is connected to the Arduino. After pairing the mobile device with the Bluetooth module, data will begin to be transmitted and it will be visible to the physiotherapist for further analysis. Bluetooth communication should cover a maximum range of 10 meters.

Pairing to these devices should only be possible for the physiotherapists and no one else, so it's possible to configure the modules previously with a list of set baud rates and custom names and passwords.

III. SOFTWARE

The mobile application was developed using Android Studio, which supports both tablet and phone as long as these have an Android system. Programming is mainly done in JAVA with the exception of the different application layouts, which is done in XML.

This will be a native application and it will be able to store data in an offline mode, meaning there will be no restriction of having to be connected to the Internet to run the app. However, it will be possible to send and receive data, via PHP scripts with the use of HTTP commands, to a server located in the Cloud, with the purpose of having synchronization between the local application's database and the remote MySQL database.

After registering, physiotherapists are always asked for their login credentials after starting the app. After that they have a wide variety of options available to them, such as viewing and editing their own personal information and also being able to add, view and edit existing patients. Patient information includes certain fields such as name, age, address, phone number, photo and date of registration. It's also possible to start a new session after choosing the patient that will be doing it,

view any patient individual session data and also view a patient overall progress since he started treatment. All this information can be safely stored in the physiotherapists mobile device, which is easily accessible, making the use of paper less needed.

IV. RESULTS AND DISCUSSION

Several tests were done with healthy volunteers, with their age, weight and height all being considered into the results. The following figures display an example of the charts displayed to the physiotherapist regarding a single patient session. Each session is indicated by number, name, date and time, while also being able to view the notes added by the physiotherapist at the end of the session. After clicking a chart a grey marker shows up indicating the value and time of the point clicked as can be seen in the figures below.



Fig. 5. FlexiForce chart data session

Fig. 5 shows a screenshot of the application while displaying the data obtained by the FlexiForce sensor in a chart. It's possible to understand that the peaks of force showing in the chart are when the patient is applying force on the hand support while taking a step, with a maximum force of 20.64 N. While the patient is only holding the crutch, the force applied was around 8 N as can also be seen in the above figure while the median force of the whole session was 10.94 N.

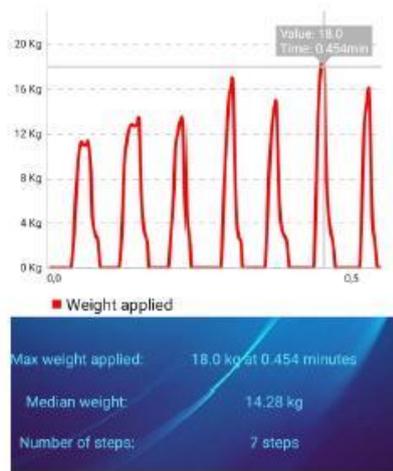


Fig. 6. Load cell chart data session

Data from the load cell sensor can be seen in Fig. 6 which is somewhat similar to the FlexiForce sensor data, however value drops to zero in this case because there is no weight being applied on the crutch base when the patient is taking a step, while on the hand support there is always force being applied even if the patient is just holding the crutch.

Additional information can be acquired with load cell data, such as maximum weight applied to the crutch, median weight and number of steps taken. An algorithm was developed for the last two variables where median weight only counts when the patient is actually applying force to the crutch, and number of steps is counted by the number of the peaks in the chart.

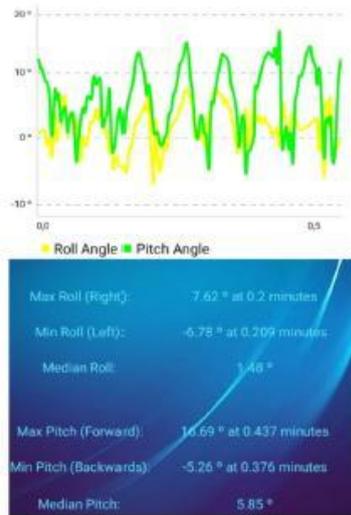


Fig. 7. IMU chart data session

IMU data is shown in Fig. 7 where the angles are shown to the physiotherapist to indicate the vertical and horizontal inclination of the crutch. For horizontal inclination, roll angle is used, and the application clearly indicates which sides does the crutch tends to tip, either being left or right, while also displaying the median angle, being 1.48 ° in this case.

Pitch angle indicates the vertical inclination of the crutch, which shows that the patient tends to tip the crutch more forward, reaching a maximum value of 16.69 °, while the backwards value only reached 5.26 °. This resulted in a median vertical inclination of 5.85 ° meaning this patient tends to tip the crutch more forward than backwards.

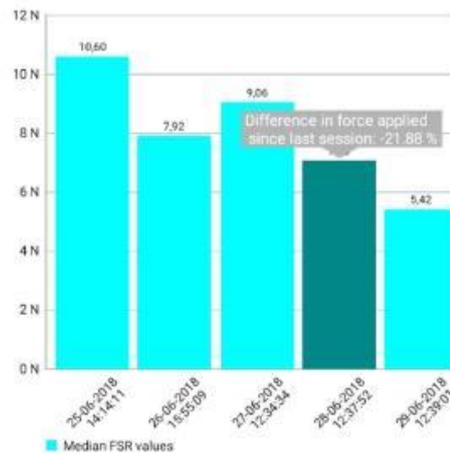


Fig. 8. Positive patient progress considering FlexiForce data

Fig. 8 shows five different sessions regarding FlexiForce sensor data. Other charts are also available in the app and follow the same thought. This can be used to view how the patient is progressing in his treatment and the above figure indicates a positive progression since the median force applied by the patient's hand decreased overall over a period of time. By clicking a column, a marker shows up indicating the percentage of increase or decrease taking into account the value of the previous session.

The opposite example can be seen in Fig. 9 where median force applied by the patient increased in the last 3 sessions, meaning the physiotherapist should probably change his way of treatment in order to get the patient back to a positive development.

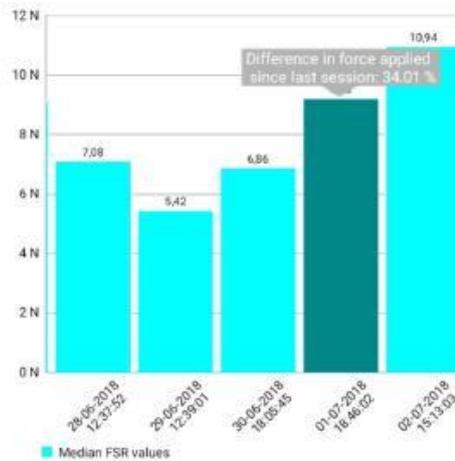


Fig. 9. Negative patient progress considering FlexiForce data

V. CONCLUSIONS

In conclusion, we can emphasize that the techniques used in conventional medicine require modernization and improvement. In an area as sensitive as health, it is important that the margin of error is constantly reduced, which can be verified with the implementation of sensors in the various equipment used. With the use of high precision sensors and an IoT system, healthcare has seen a gradual decentralization from the traditional health center-based approach [10].

Knowing that the population that uses physical therapy is not limited to an age group or type of medical condition, the use of sensors will allow progressive and more accurate monitoring on both sides of the patient-physician relationship. It should be noted that one of the advantages associated with this method is the simplicity in which the results are obtained and analyzed, since a large percentage of the population now depends on the Internet for the most basic functions.

With the use of a multimodal IoT system attached to objects of relevance in the field of physiotherapy, in this case, a crutch, data obtained from the two types of force sensors and the IMU can be easily visible to the physiotherapist in order to figure out if the patient's development is going on the right path, and if not, change the type of treatment to make sure that it is.

As future work, MQTT protocol should be implemented for communication with the Cloud for synchronization purposes as it is widely used in IoT systems [11] while offering more security to the whole system. A Big Data analytics system should also be developed in order to develop new application elements, such as prediction models.

ACKNOWLEDGMENT

This research is supported by Instituto de Telecomunicações (IT-IUL) at ISCTE-IUL, Lisbon, Portugal and Fundação para a Ciência e Tecnologia (FCT) project PTDC/DTP-DES/6776/2014.

REFERENCES

- [1] "Physiotherapy & Rehabilitation | Pt Health". *pt Health*. N.p., 2016. Web. 1 May 2018. <https://www.pthealth.ca/service/physiotherapy/>.
- [2] Luca Catarinucci, Danilo de Donno, Luca Mainetti, Luca Palano, Luigi Patrono, Maria Laura Stefanizzi, Luciano Tarricone, "An IoT-Aware Architecture for Smart Healthcare Systems", *IEEE JOURNAL*, vol. 2, no. 6, December 2015.
- [3] I. Edwards, M. Jones, J. Carr, A. Braunack-Mayer and G. M. Jensen. "Clinical reasoning strategies in physical therapy." *Phys Ther.*, vol. 84, pp. 312-330. 2004.
- [4] Linder JA, Ma J, Bates DW, Middleton B, Stafford RS. Electronic Health Record Use and the Quality of Ambulatory Care in the United States. *Arch Intern Med*. 2007;167(13):1400-1405.
- [5] O. Postolache, "Physical rehabilitation assessment based on smart training equipment and mobile APPs," *2015 E-Health and Bioengineering Conference (EHB)*, Iasi, 2015, pp. 1-6.
- [6] Janota, Aleš et al. "Improving the Precision and Speed of Euler Angles Computation from Low-Cost Rotation Sensor Data." Ed. Stefano Mariani. *Sensors (Basel, Switzerland)* 15.3 (2015): 7016-7039. *PMC*. Web. 1 May 2018.
- [7] A. Nallathambi, T. Shanmuganatham, D. Sindhanaiselvi, "Design and Analysis of MEMS based Piezoresistive Pressure sensor for Sensitivity Enhancement", *Materials Today: Proceedings*, Volume 5, Issue 1, Part 1, 2018, Pages 1897-1903
- [8] H. Ferdinando, H. Khoswanto and D. Purwanto, "Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATmega8535," *2012 International Symposium on Innovations in Intelligent Systems and Applications*, Trabzon, 2012, pp. 1-5.
- [9] "Wheatstone Bridge Circuit", *ElectronicsTutorials*, N.p., 2018. Web. 1 May 2018. <https://www.electronics-tutorials.ws/blog/wheatstone-bridge.html>.
- [10] N. Kumar, "IoT architecture and system design for healthcare systems," *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, Bangalore, 2017, pp. 1118-1123.
- [11] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," *2015 Fifth International Conference on Communication Systems and Network Technologies*, Gwalior, 2015, pp. 746-75.

Annex B – User Manual



Departamento de Ciências e Tecnologias da Informação

User Manual

Smart Object for Physical Rehabilitation Assessment

Pedro Martim Valente Lima Frango

Supervisor:
Dr. Octavian Postolache, Assistant Professor,
ISCTE-IUL

Co-Supervisor:
Dr. Vítor Viegas, Assistant Professor,
Escola Naval

October 2018

Contents

Figures	116
User Manual.....	117
Chapter 1 – Arduino application installation	118
1.1. Bluetooth HC-06 module configuration	118
1.2. SmartCrutch system file configuration	121
Chapter 2 – Mobile application installation.....	122
1. Installing the application using the apk file provided.....	122
2. Installing the application using Android Studio	126
Chapter 3 – IoPhyr Mobile application.....	128

Figures

Figure 1.1 – Arduino IDE.....	118
Figure 1.2 – Connections for Bluetooth configuration.....	119
Figure 1.3 – Connections for Bluetooth configuration.....	119
Figure 1.4 – Bluetooth configuration file	120
Figure 1.5 – Steps for uploading Bluetooth configuration file.....	120
Figure 1.6 – SmartCrutch configuration file	121
Figure 1.7 – SmartCrutch system circuit	122
Figure 2.1 – USB options	123
Figure 2.2 – Provided .apk file copied into the created folder	123
Figure 2.3 – Turning on unknown sources option on the mobile device	124
Figure 2.4 – APK Manager Pro software	124
Figure 2.5 – Installation of IoPhyr APP using APK Manager	125
Figure 2.6 – IoPhyr APP	125
Figure 2.7 – Extracted .rar file provided	126
Figure 2.8 – Open IoPhyr project using Android Studio.....	126
Figure 2.9 – Running the application in Android Studio	127
Figure 3.1 – IoPhyr APP icon.....	128
Figure 3.2 – IoPhyr login activity.....	128
Figure 3.3 – Physiotherapist register	129
Figure 3.4 – Errors displayed to the user when registering a physiotherapist.....	129
Figure 3.5 – Successful register of a physiotherapist	130
Figure 3.6 – Synchronization of physiotherapists	130
Figure 3.7 – User Area	131
Figure 3.8 – Physiotherapist profile view.....	132
Figure 3.9 – Physiotherapist profile edit	132
Figure 3.10 – Patients activity	133
Figure 3.11 – Adding a new patient	133
Figure 3.12 – Error notification when creating a patient.....	134
Figure 3.13 – Success notification when creating a patient	134
Figure 3.14 – Patient form.....	135
Figure 3.15 – Patient list.....	135
Figure 3.16 – Patient data edit	136
Figure 3.17 – Success notification when editing patient data	136
Figure 3.18 – Selecting patient for session activity.....	137
Figure 3.19 – Sessions activity	137
Figure 3.20 – New session activity.....	138
Figure 3.21 – Live FSR sensor data	139
Figure 3.22 – Live load cell sensor data.....	139
Figure 3.23 – Live IMU sensor data.....	140
Figure 3.24 – Session name and notes.....	141
Figure 3.25 – View session activity	141
Figure 3.26 – View single session	142
Figure 3.27 – View notes taken	142
Figure 3.28 – FSR data analysis	143
Figure 3.29 – Load cell data analysis	144
Figure 3.30 – IMU data analysis	145
Figure 3.31 – Negative patient progress considering FSR data	146
Figure 3.32 – Positive patient progress considering FSR data	146

User Manual

This manual aims to present the features available to the physiotherapist in the IoPhyr mobile application, explaining in detail how to properly work with the application. Chapter 1 explains software installation regarding Arduino software which will be uploaded to the Arduino Nano designated to process sensor values. Chapter 2 is intended to explain two possible methods of installation of the IoPhyr mobile application. A method that includes installation through the use of Android Studio software is present, as well as one that only requires the .apk file provided. Chapter 3 will focus on explaining the proper use of the mobile application after installation, to be used by physiotherapists.

Chapter 1 – Arduino application installation

In this chapter installation of all the Arduino software used, such as configuration of the Bluetooth HC-06 module and the processing code uploaded to the Arduino for sensor data reading and processing is done. For this purpose, Arduino IDE software must be installed in the computer that will upload the files to the Arduino. This software is free and can be downloaded at <https://www.arduino.cc/en/Main/Software>. It is available for Windows, Mac OS and Linux systems. After the software has been installed, executing it results in Figure 1.1.



Figure 1.1 – Arduino IDE

1.1. Bluetooth HC-06 module configuration

The Bluetooth module that will be used for sensor data transmission between the Arduino and the mobile device needs to be properly configured beforehand. For proper configuration, connections from the Bluetooth module to the Arduino must be done, visible in Figure 1.2. The Arduino should be connected to a computer through USB.

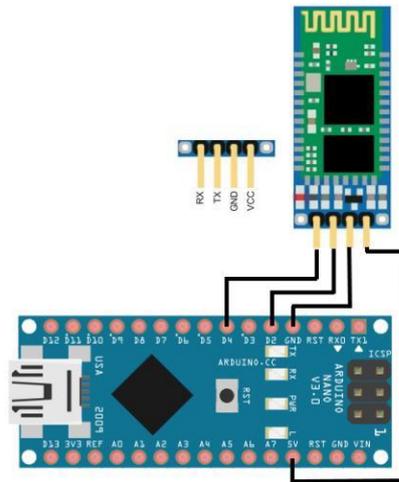


Figure 1.2 – Connections for Bluetooth configuration

After the connections are made, a file provided must be opened by Arduino IDE, by clicking on File -> Open -> select file *IoPhyr_Bluetooth.ino* -> Open, as Figure 1.3 indicates.

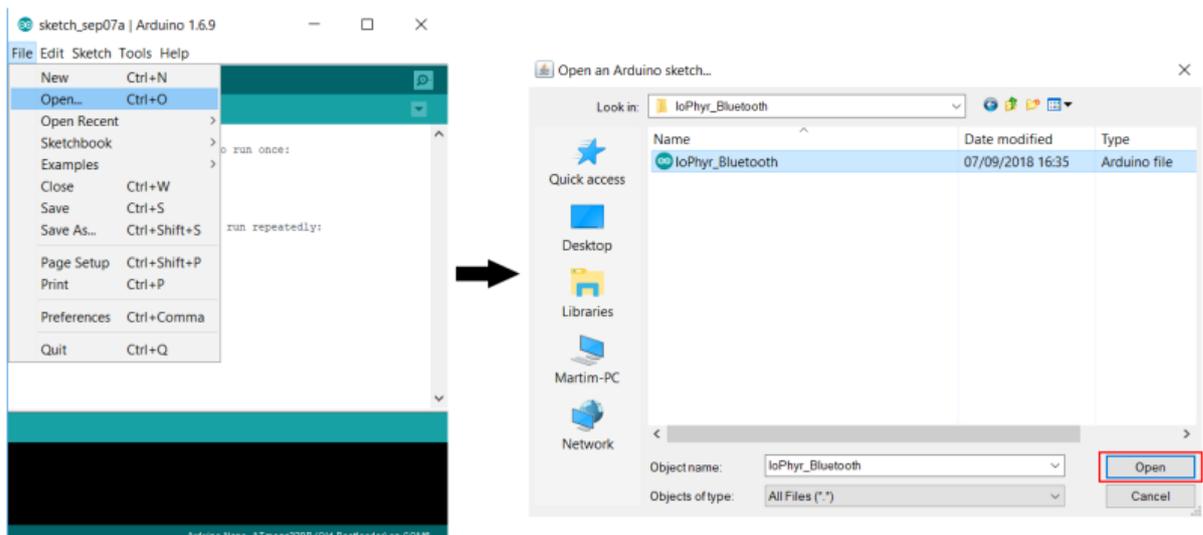
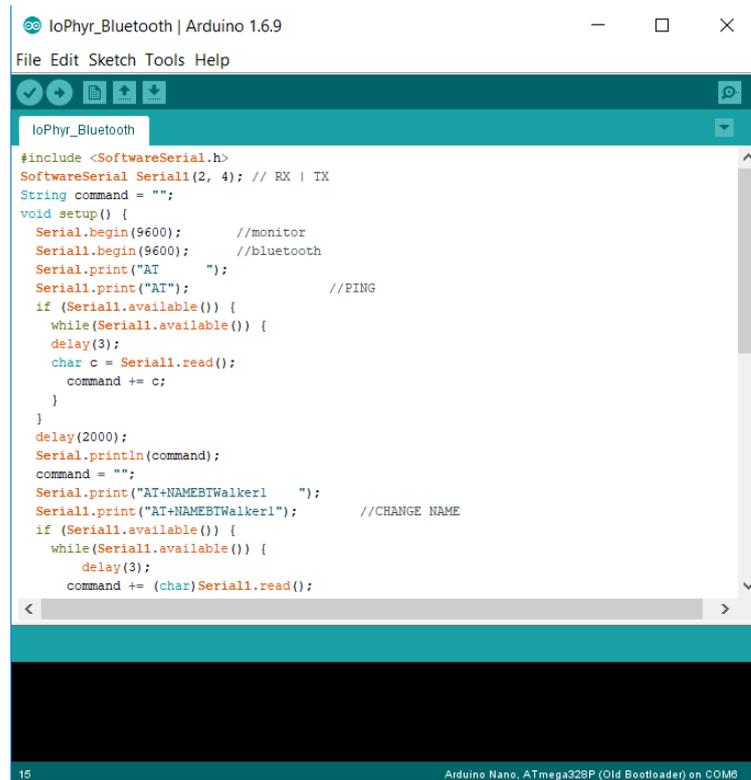


Figure 1.3 – Connections for Bluetooth configuration

The following window should open with all the data ready for configuration, as seen in Figure 1.4. Commentaries along the file are visible, in this case, only changing the name is visible, but changing password and baud rate is also possible by simply editing the indicated fields.



```
#include <SoftwareSerial.h>
SoftwareSerial Serial1(2, 4); // RX | TX
String command = "";
void setup() {
  Serial.begin(9600); //monitor
  Serial1.begin(9600); //bluetooth
  Serial.print("AT");
  Serial1.print("AT"); //PING
  if (Serial1.available()) {
    while (Serial1.available()) {
      delay(3);
      char c = Serial1.read();
      command += c;
    }
  }
  delay(2000);
  Serial.println(command);
  command = "";
  Serial.print("AT+NAMEBTWalker1");
  Serial1.print("AT+NAMEBTWalker1"); //CHANGE NAME
  if (Serial1.available()) {
    while (Serial1.available()) {
      delay(3);
      command += (char)Serial1.read();
    }
  }
}
```

Figure 1.4 – Bluetooth configuration file

Finally, in order to upload the information to the Bluetooth adapter through the Arduino, the file, also designed as sketch, must be uploaded. Several steps must be taken into account before, as seen in Figure 1.5.

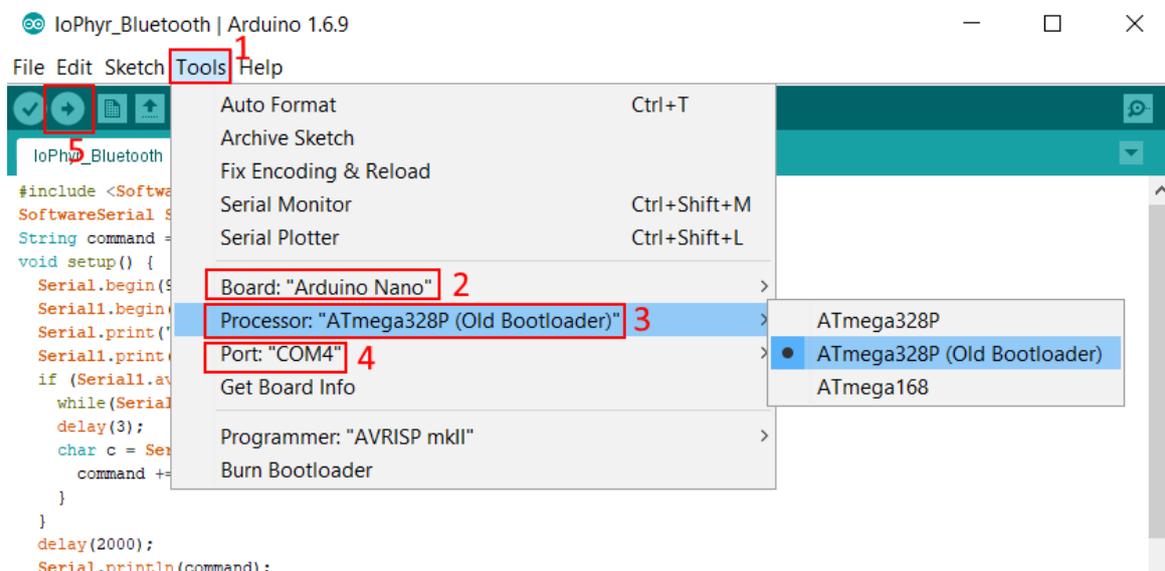


Figure 1.5 – Steps for uploading Bluetooth configuration file

1. Click the Tools tab on the Arduino IDE;
2. Choose the right Board, in this case, Arduino Nano;
3. Choose the right processor. This varies depending on the version of the Arduino Nano. In this case an ATmega328P is used. Boards sold from Arduino since

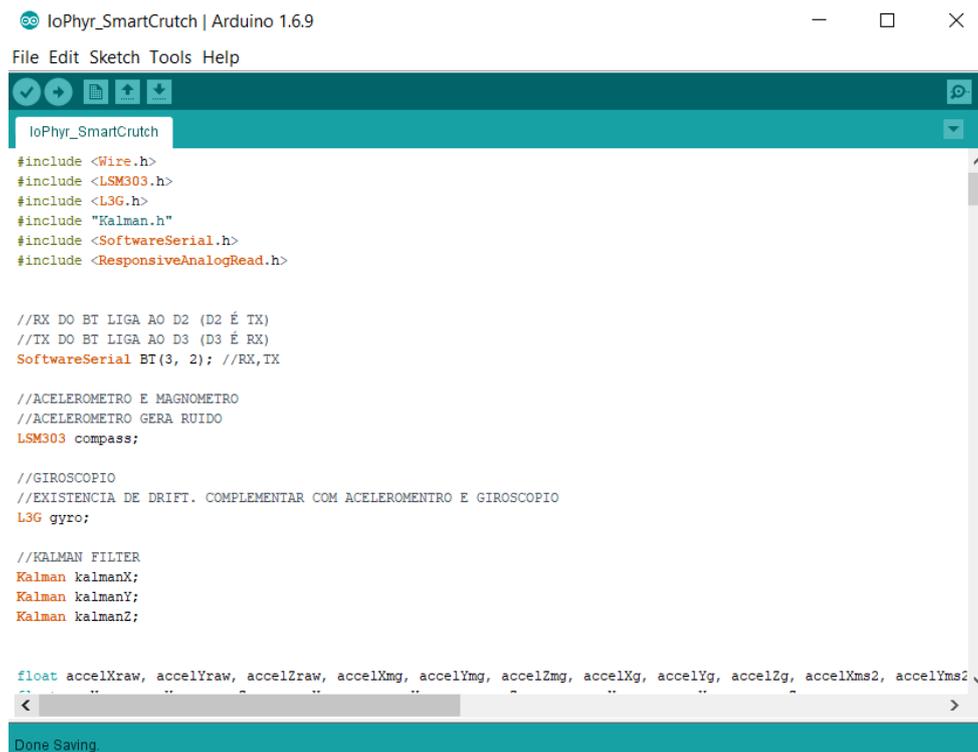
January 2018 use a new bootloader. Since the Arduino used was bought before, a new bootloader must be selected.

4. Select the right port on which the Arduino is connected to via USB;
5. Click the Upload button and the Bluetooth module is now configured.

The name used for the adapter was “BTWalker1”, with a password “1234” and a baud rate of 9600.

1.2. SmartCrutch system file configuration

The main file that will be ran continuously by the Arduino for sensor reading and data transmission must be uploaded after the Bluetooth module configuration. For uploading this file to the Arduino, nothing should be connected to the Arduino itself, with the exception of an USB cable connected to a computer. Another provided file, *IoPhyr_SmartCrutch.ino* should be uploaded to the Arduino using the same steps as Figure 1.3 but choosing the new file this time. Figure 1.6 shows the expected result.



```
IoPhyr_SmartCrutch | Arduino 1.6.9
File Edit Sketch Tools Help
IoPhyr_SmartCrutch
#include <Wire.h>
#include <LSM303.h>
#include <L3G.h>
#include "Kalman.h"
#include <SoftwareSerial.h>
#include <ResponsiveAnalogRead.h>

//RX DO BT LIGA AO D2 (D2 É TX)
//TX DO BT LIGA AO D3 (D3 É RX)
SoftwareSerial BT(3, 2); //RX,TX

//ACELEROMETRO E MAGNETOMETRO
//ACELEROMETRO GERA RUIDO
LSM303 compass;

//GIROSCOPIO
//EXISTENCIA DE DRIFT. COMPLEMENTAR COM ACELEROMETRO E GIROSCOPIO
L3G gyro;

//KALMAN FILTER
Kalman kalmanX;
Kalman kalmanY;
Kalman kalmanZ;

float accelXraw, accelYraw, accelZraw, accelXmg, accelYmg, accelZmg, accelXg, accelYg, accelZg, accelXms2, accelYms2
```

Figure 1.6 – SmartCrutch configuration file

Uploading the sketch to the Arduino is done the same way as before, as seen in Figure 1.5. After the upload has been successfully done, all the connections regarding the whole system used must be done, as seen in Figure 1.7.

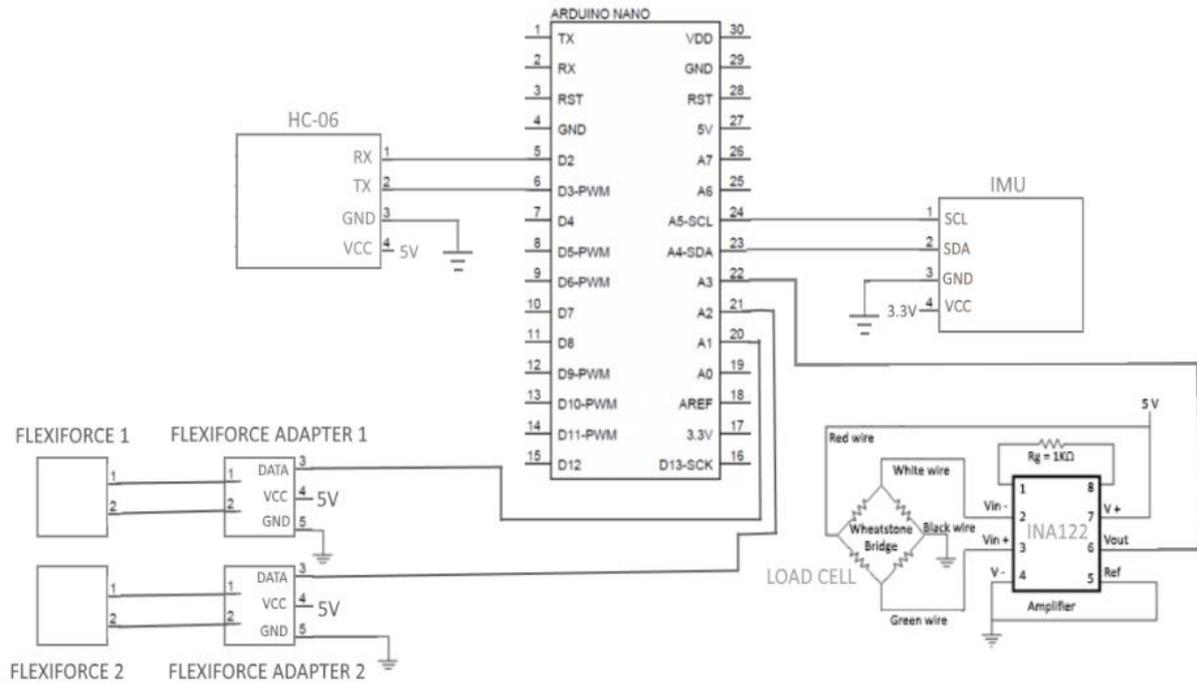


Figure 1.7 – SmartCrutch system circuit

The Arduino is now ready to transmit data to the mobile devices that connect to it.

Chapter 2 – Mobile application installation

Installing the mobile application on a mobile device can be done in two ways:

- Installing the apk file provided through the use of an APP that manages apk files;
- Installing the application through Android Studio.

1. Installing the application using the apk file provided

- 1.1. Connect the mobile device to the computer through USB and select the “Transfer files” option (see Figure 2.1);



Figure 2.1 – USB options

1.2. Create a folder in the mobile device folder and name it “apk” and then copy the provided .apk file into the folder (see Figure 2.2);

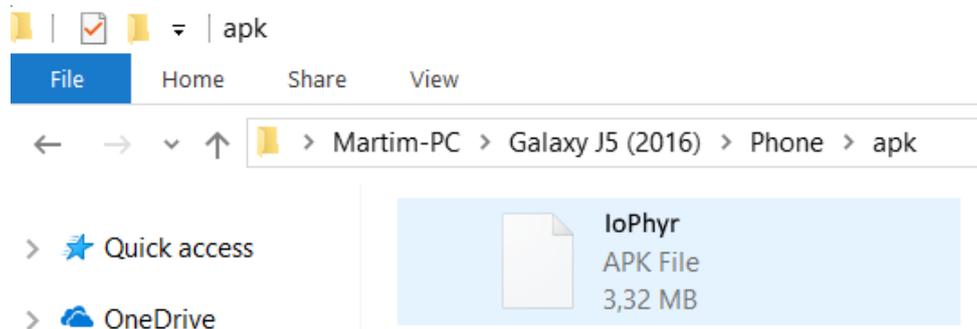


Figure 2.2 – Provided .apk file copied into the created folder

1.3. Applications that are not installed through the *Play Store* must be manually allowed to be installed by the user. In the mobile device, go to Settings (Definições) -> Lock screen and Security (Ecrã bloqueio e Segurança) -> Unknown sources (Origens Desconhecidas) and turn on that option (see Figure 2.3);



Figure 2.3 – Turning on unknown sources option on the mobile device

1.4. Download and install “*APK Manager Pro*” software from the *Play Store* in order to install the application (see Figure 2.4);

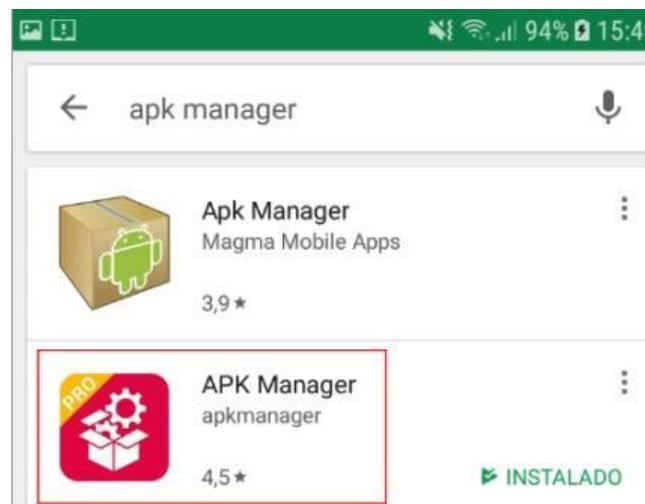


Figure 2.4 – APK Manager Pro software

1.5. Install the mobile application (see Figure 2.5) by running APK Manager PRO -> Local (1) -> apk (2) -> Press and hold IoPhyr.apk (3) -> Install (4);

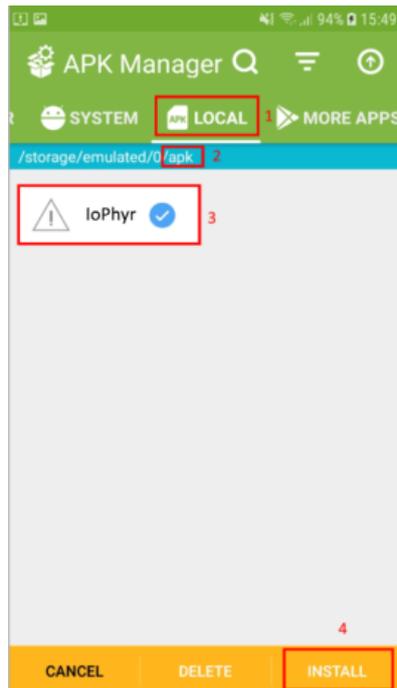


Figure 2.5 – Installation of IoPhyr APP using APK Manager

1.6. Execute IoPhyr APP (see Figure 2.6).



Figure 2.6 – IoPhyr APP

2. Installing the application using Android Studio

- 1) Extract the .rar file provided that contains all the code from the APP (see Figure 2.7);

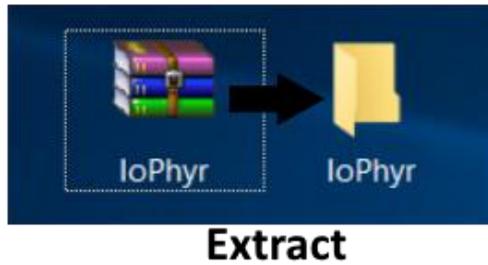


Figure 2.7 – Extracted .rar file provided

- 2) Open Android Studio and go to File -> Open -> Select the folder extracted -> Click Ok (see Figure 2.8);

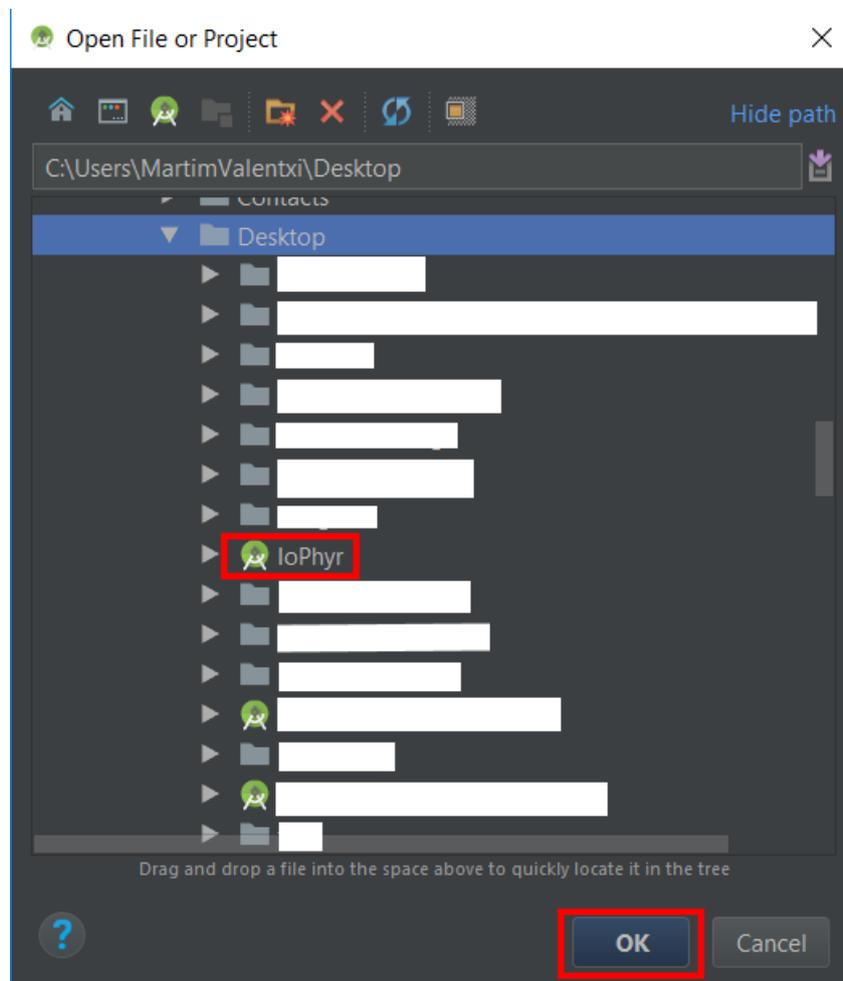


Figure 2.8 – Open IoPhyr project using Android Studio

- 3) Run the APP on the mobile device (see Figure 2.9) by clicking the Run icon (1) -> Selecting the mobile device connected (2) -> Clicking Ok (3);

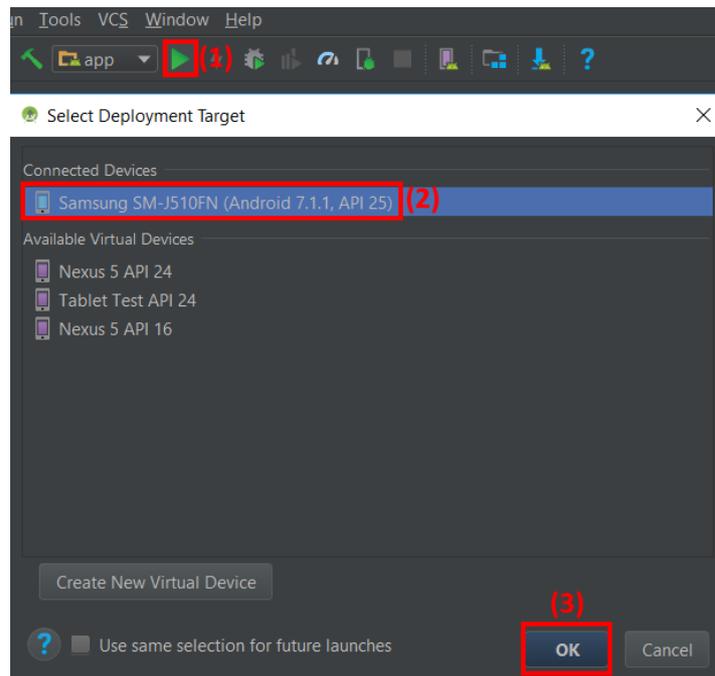


Figure 2.9 – Running the application in Android Studio

- 4) The application IoPhyr will now be installed on the mobile device and doesn't require Android Studio anymore to be runned.

Chapter 3 – IoPhyr Mobile application

This chapter focuses on explaining how IoPhyr works and how the application should be managed in order to obtain the best possible outcome. The physical therapists managing the application should be aware of the full possibilities of the application, therefore reading of this manual is a must. In order to start the application, simply clicking on the IoPhyr icon starts the APP (see Figure 3.1);

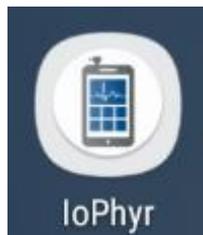


Figure 3.1 – IoPhyr APP icon

The application then starts and displays a login window (see Figure 3.2).



Figure 3.2 – IoPhyr login activity

- If the user is not registered in the APP he should click the “Register Here” text (4). Information such as first name, last name, username, password, email, age and phone number must be filled (see Figure 3.2). Failure to fill out all the data will return in a popup message to the user, preventing him to successfully register while also telling the user what needs to be changed/fixed to properly register (see Figure 3.4).



Figure 3.3 – Physiotherapist register

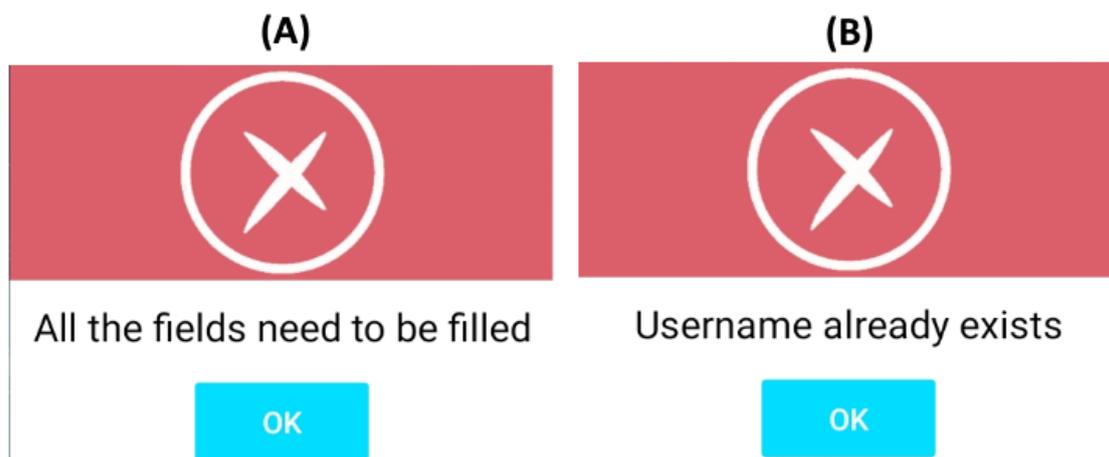


Figure 3.4 – Errors displayed to the user when registering a physiotherapist

- In Figure 3.4, (A) will be displayed to the user when he tries to click the Register button and a field of information in Figure 3.3 is not filled. When a username is already registered in the database with the same username that the user filled, (B) is shown indicating that the user must choose a different username.
- If the user successfully registers himself into the application, another window pops up indicating a successful register (see Figure 3.5);

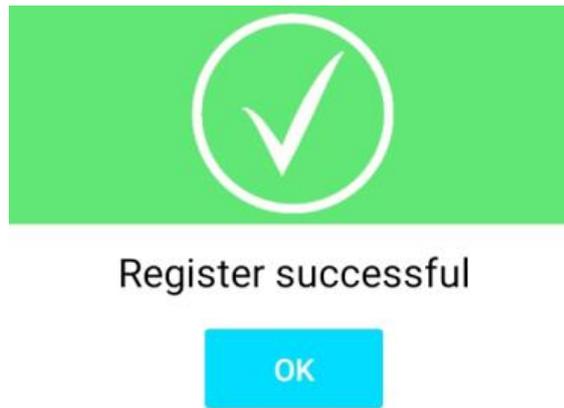


Figure 3.5 – Successful register of a physiotherapist

- Going back to the login window in Figure 3.2, if the user is registered but his data is not on the mobile device he/she's currently using, he/she should turn on the mobile device's Internet connectivity and press the synchronize button (5) in order to synchronize all data. The user is notified that synchronization is now in process while displaying an animated progress icon (see Figure 3.6);

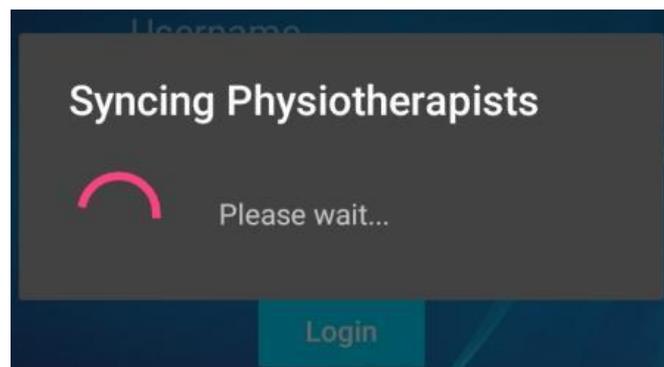


Figure 3.6 – Synchronization of physiotherapists

- If the user is already registered, simply inserting his username (1) and password (2) in the correct fields and then clicking the Login button (3) will guide the user to a new window, the User Area (see Figure 3.7);



Figure 3.7 – User Area

- 1) Logout icon – goes back to the login window;
 - 2) Synchronize button – Synchronizes all patient and session data related to the physiotherapist that is logged in;
 - 3) My Profile icon – view and edit any data related to the physiotherapist that is logged in;
 - 4) Patients icon – view and edit any data of patients related to the physiotherapist that is logged in;
 - 5) Sessions icon – view any session data of patients related to the physiotherapist that is logged in.
- It is possible to view and edit the physiotherapist's own profile by clicking on the My Profile icon (3), displaying a new window seen in Figure 3.8;



Figure 3.8 – Physiotherapist profile view

- When the Edit button is pressed, all the values become editable (see Figure 3.9) and when trying to save them by clicking the Save button, the same notifications seen in Figure 3.4 and 3.5 indicating if the Edit was successful or not are displayed;



Figure 3.9 – Physiotherapist profile edit

- Going back into the User Area, clicking on the Patients icon displays a new window providing the possibility of viewing or adding new patients (see Figure 3.10);

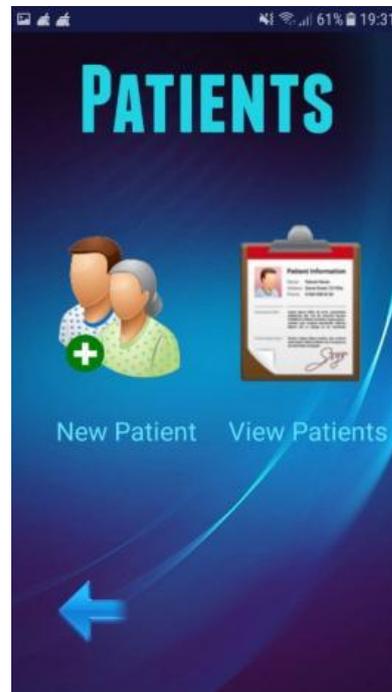


Figure 3.10 – Patients activity

- When the New Patient icon is clicked, a new activity is displayed to the user (see Figure 3.11). By clicking the photo icon (1) it is possible to take a picture of the patient, which can be edited any time.

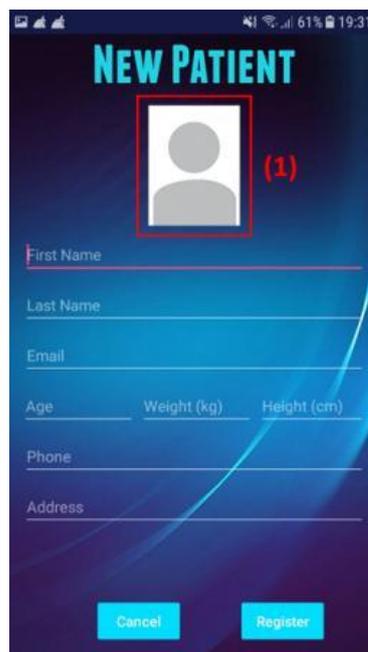


Figure 3.11 – Adding a new patient

- Unreal values inserted, such as height and weight are displayed (see Figure 3.12) in addition to the error displayed when trying to register a patient without filling all the fields (see Figure 3.4). A photo of the patient is not required and can be left in blank;

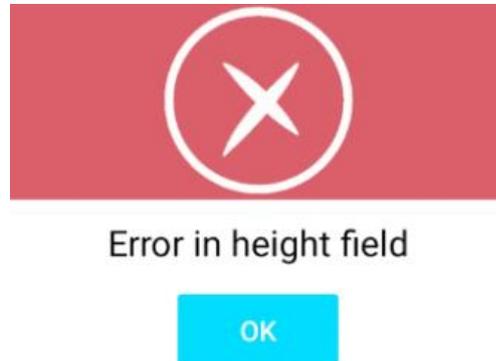


Figure 3.12 – Error notification when creating a patient

- If the patient is successfully created, a new notification pops up (see Figure 3.13);

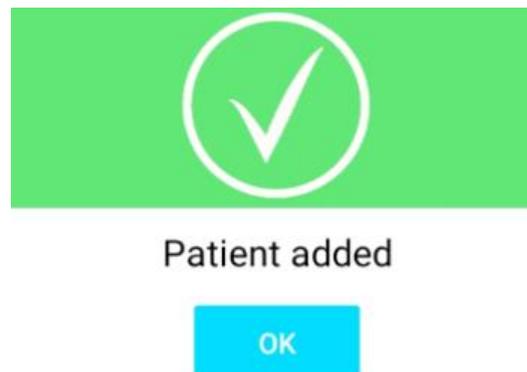


Figure 3.13 – Success notification when creating a patient

- When the user wants to view or edit patient information, simply clicking the View Patients icon in Figure 3.10 opens a patient form (see Figure 3.14);



Figure 3.14 – Patient form

- The user can browse through all the patients by simply clicking the tab (1) that displays the first and last name of all the patients that the physiotherapist has associated with him (see Figure 3.15);

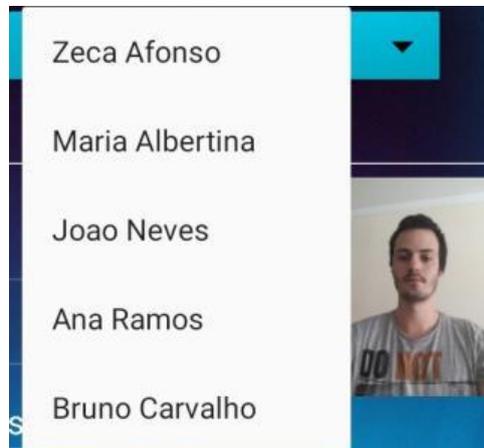


Figure 3.15 – Patient list

- When clicking the edit button, all the patient data, except the date registered can be changed, including the picture taken (see Figure 3.16);



Figure 3.16 – Patient data edit

- When trying to save patient data, if unsuccessful, the same errors displayed when trying to create a patient will be displayed. If saving new data is successful, a new notification is displayed (see Figure 3.17);

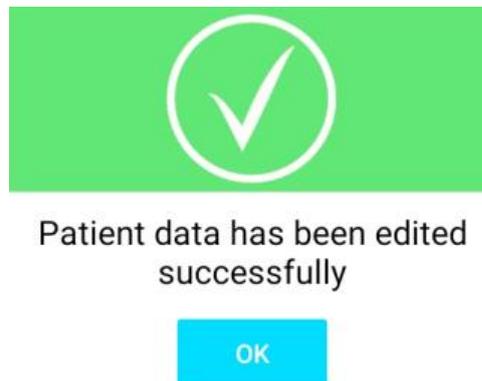


Figure 3.17 – Success notification when editing patient data

- It is also possible for the physiotherapist to start and view previous sessions of certain patients. When clicking the Sessions icon seen in the user area activity (see Figure 3.7), a similar layout to the patient view activity is displayed, where the physiotherapist must choose the patient that he wishes to start a new session or view old session data. This is done by simply selecting the right patient and clicking the Next button, as seen in Figure 3.18.



Figure 3.18 – Selecting patient for session activity

- After the patient has been chosen, its name is displayed in the top region of the screen, so the physiotherapist always knows which patient he/she is dealing with. Two new icons are displayed, one for starting a new session and another for viewing previous sessions (see Figure 3.19).



Figure 3.19 – Sessions activity

- Starting a new session can be done by clicking the New Session icon visible in Figure 3.19. A new activity is displayed, where Bluetooth connection can be turned on and off by clicking a button (see Figure 3.20). If Bluetooth is available, a new button is shown that will discover all Bluetooth devices nearby and display them in a list. Pairing with such devices is done by simply clicking the right device. In this case, BTWalker1 was the name given to the Bluetooth Adapter connected to the SmartCrutch. If it is the first time connecting to the Bluetooth Adapter, a password “1234” must be inserted. After pairing, a connect button is displayed and when clicked it starts receiving sensor information regarding the patient’s training session and displays them in charts.



Figure 3.20 – New session activity

- By scrolling down the screen, charts displaying FSR, load cell and IMU data are visible to the physiotherapist for live data analysis. Each live chart provides a good flow of data, ensuring that the flow of data shown to the physiotherapist is acceptable by indicating a fixed number of visible samples before the chart moves. Charts Y values are expressed when viewing single session data and X values are displayed in minutes. After the session has started, data from the SmartCrutch is transmitted to the mobile device and is visible by scrolling down (see Figure 3.21);



Figure 3.21 – Live FSR sensor data

- By scrolling down again the live session activity, a load cell data chart is available. Figure 3.22 shows an example of a step taken. The load cell reached a maximum value of 24 kilograms, and when not in contact with the floor the value dropped to zero;

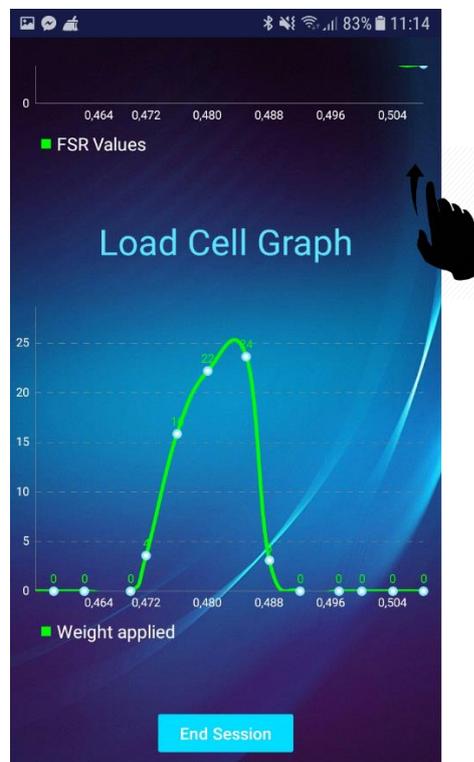


Figure 3.22 – Live load cell sensor data

- Finally, the data displayed the IMU sensor is visible in the last chart (see Figure 3.23) where more relevance is given to the roll and pitch angle, which indicate horizontal and vertical inclination of the crutch, respectively. Positive and negative values are possible. Values from 0° to 180° indicate that the crutch is being tilted forward in case of the pitch angle, and towards the right in case of the roll angle. Values from -180° to 0° indicate that the crutch is being tilted backwards in case of the pitch angle, and towards the left in case of the roll angle. This information is visible when the physiotherapist is viewing single data sessions. Finally, yaw angle ranges from 0° to 360° and is simply to indicate when the patient moves the crutch's orientation, which is useful to understand when the patient turned his body in the session;

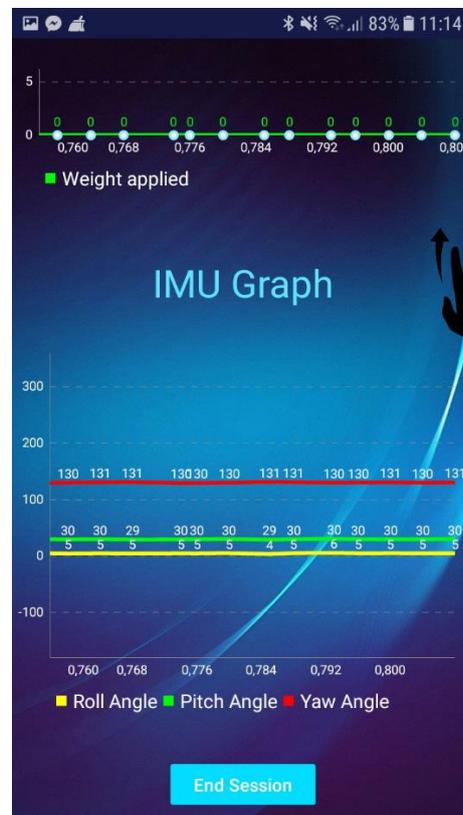


Figure 3.23 – Live IMU sensor data

- The session ends when the physiotherapist clicks the End Session button that is always visible during the session, and immediately a popup window shows up asking for the physiotherapist to name the session created and afterwards notes regarding the session (see Figure 3.24);

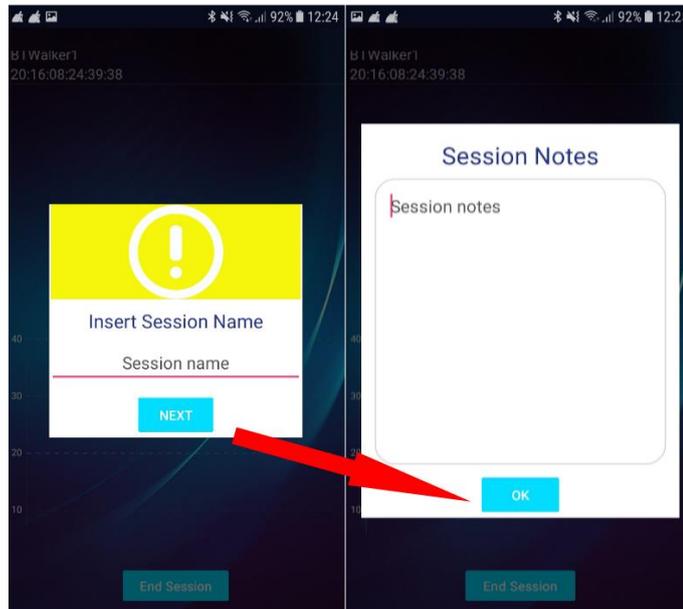


Figure 3.24 – Session name and notes

- After a session has ended, the application automatically enters the Viewing Sessions activity (see Figure 3.25), which can be accessed by clicking the View Sessions icon visible in Figure 3.19. This activity presents the user with two new icons which provide the possibility of viewing single session data, including but not excluding the new session that was created, and the data regarding all the sessions the patient has done;



Figure 3.25 – View session activity

- If the patient decides to analyze a single session, a list of all the selected patient's session show up (see Figure 3.26), providing information such as session number, name, date and time created. Session duration time is visible once the physiotherapist selects a session. Deleting sessions is also possible by clicking the X icon on the top right of screen;



Figure 3.26 – View single session

- When a session has been selected a new activity appears. Notes taken from the session can be seen in this activity by simply clicking the notepad icon (see Figure 3.27);

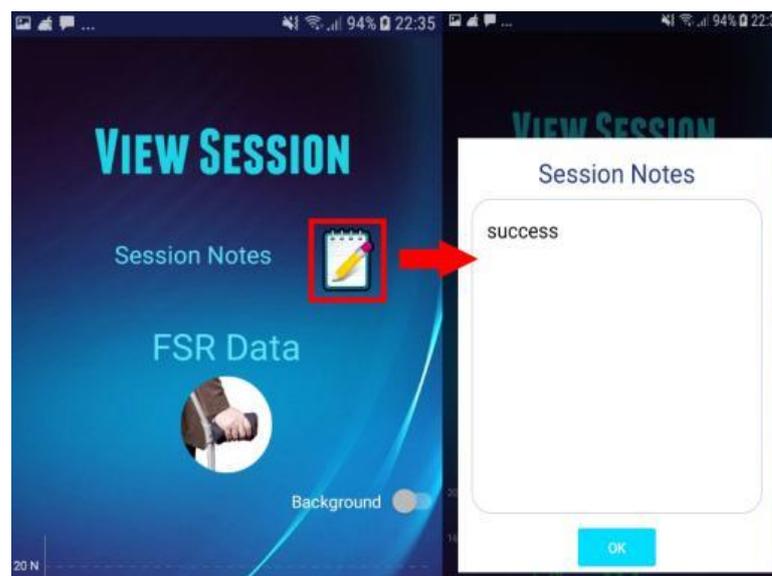


Figure 3.27 – View notes taken

- Charts containing sensor information and other important information, such as maximum force and median force applied, in the case of the FSR sensor, as seen in Figure 3.28. A transparent background is set as default but if the physiotherapist wants to change it to a white background, simply clicking the toggle Background switch button enables this. Clicking and zooming on the charts is also possible, with the first one providing information about the sample clicked, such as the value and the time at which that value was taken.

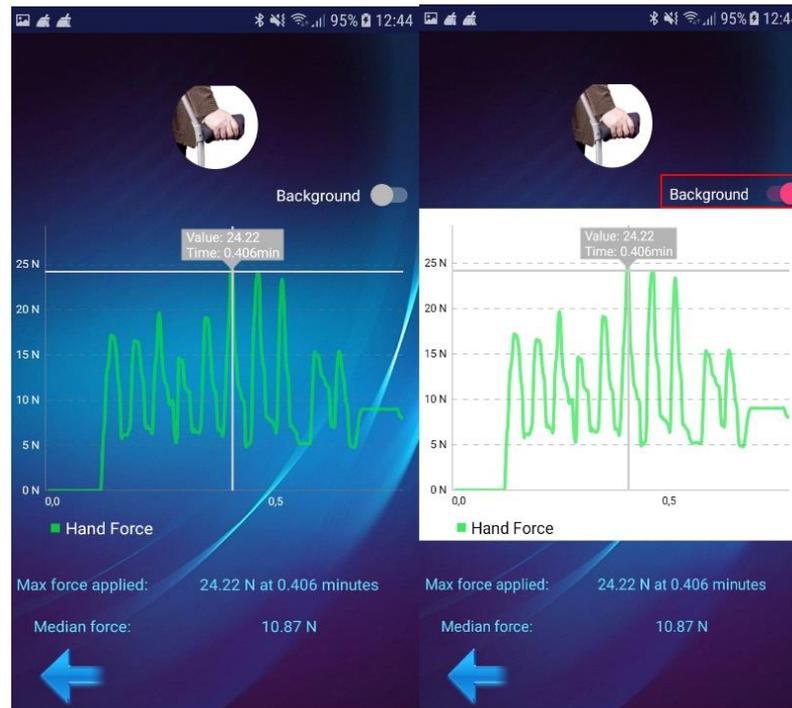


Figure 3.28 – FSR data analysis

- Similar to the live session activity, this activity can also be scrolled down for viewing more information. Figure 3.29 shows the chart of the load cell data acquired, while also indicating the maximum and medium weight applied, as well as the number of steps taken;

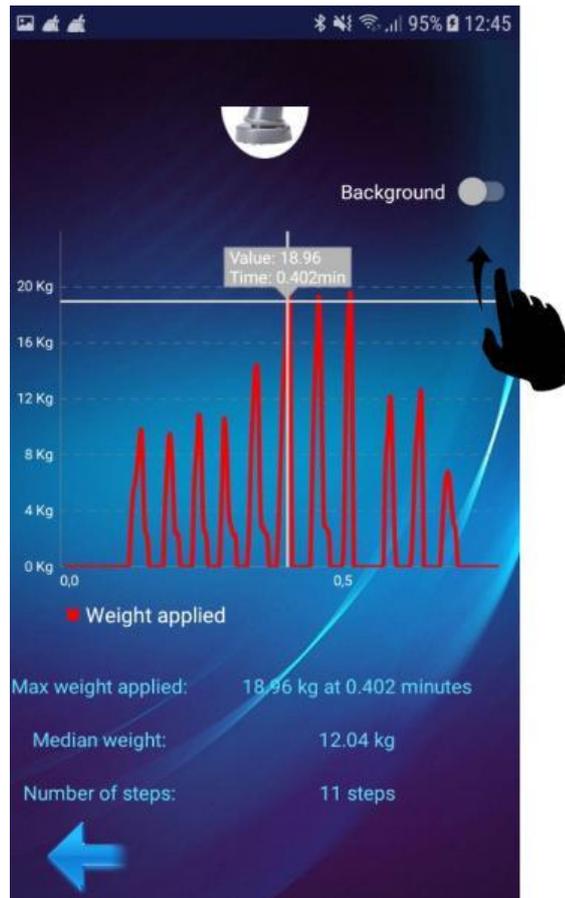


Figure 3.29 – Load cell data analysis

- Finally, data collected from the IMU is visible in the last chart (see Figure 3.29) where multiple information is also displayed. As mentioned before, values from 0° to 180° indicate that the crutch is being tilted forward in case of the pitch angle, and towards the right in case of the roll angle. Values from -180° to 0° indicate that the crutch is being tilted backwards in case of the pitch angle, and towards the left in case of the roll angle. Maximum and minimum values of both angles are displayed, as well as the median angles, which are a great indicator in the patient's gait;

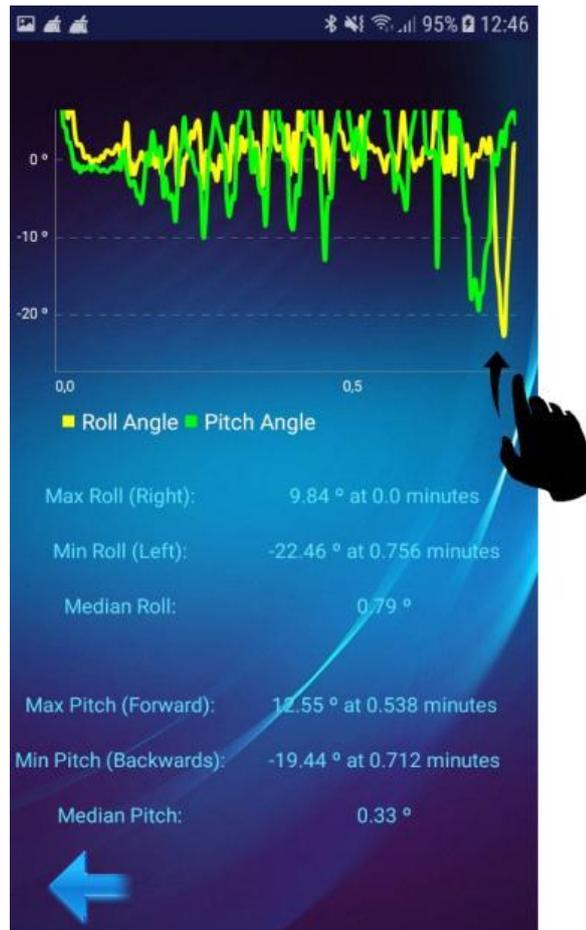


Figure 3.30 – IMU data analysis

- Instead of viewing a single session, the physiotherapist may choose to view the overall progress of the patient by clicking the All Sessions icon in Figure 3.25. Figure 3.29 indicates only one of the bar charts visible to the user, which considers the median FSR force of each session. Charts with other sensor information are visible by scrolling down the activity, as before. Date and time are displayed in the X axis while the mean value is displayed in the Y axis. Only 5 bars were made visible at once to avoid cluttering. By scrolling the chart to the right or left, more sessions will become visible. When clicking a session bar, it will indicate the difference of value between the session clicked and the last session, in percentage. A negative difference means there was less force applied, therefore leading to patient improvement, while a positive difference means the force applied was higher, leading to a negative improvement. This figure indicates a negative patient progress, since after the second visible session the mean force applied kept increasing;

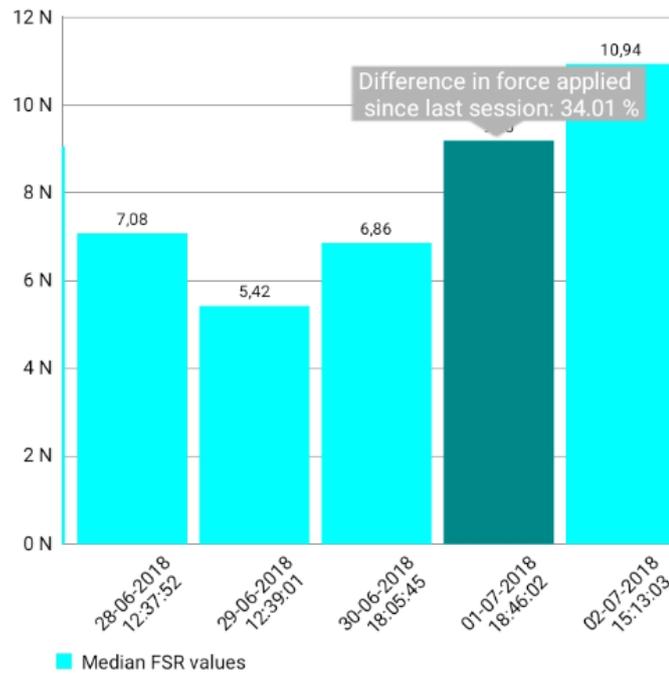


Figure 3.31 – Negative patient progress considering FSR data

- On the other hand, a positive patient progress is visible in Figure 3.30, where a negative difference in force applied since the last session is a good indicator of overall progress by the patient.

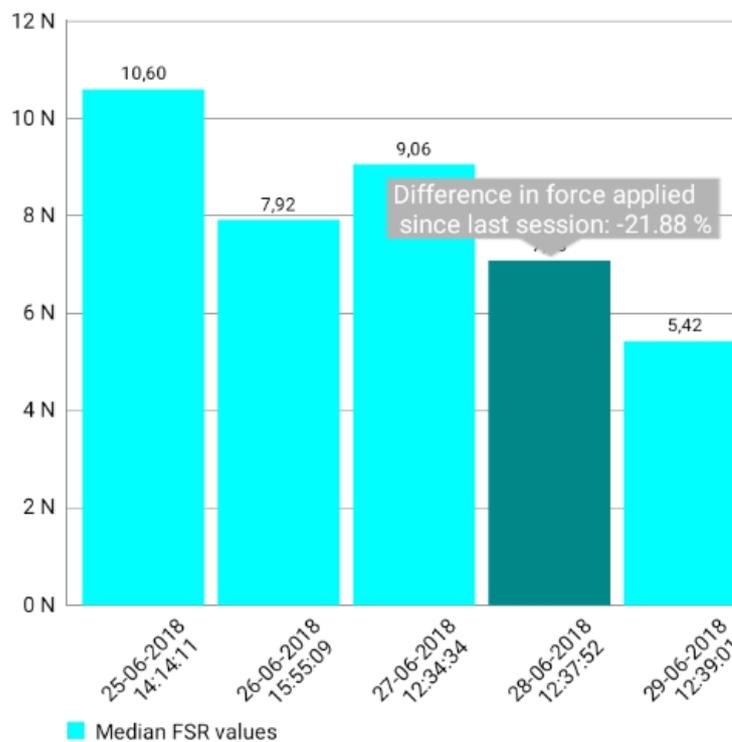


Figure 3.32 – Positive patient progress considering FSR data

Annex C – Technical Manual



Technical Manual

Smart Object for Physical Rehabilitation Assessment

Pedro Martim Valente Lima Frango

Supervisor:
Dr. Octavian Postolache, Assistant Professor,
ISCTE-IUL

Co-Supervisor:
Dr. Vítor Viegas, Assistant Professor,
Escola Naval

October 2018

Contents

Figures	149
Tables.....	150
Technical Manual	151
Chapter 1 – Arduino application	152
1.1. Libraries	152
1.2. Objects and variables	153
1.3. Setup method	153
1.4. Loop method	Error! Bookmark not defined.
1.5. IMU method	Error! Bookmark not defined.
1.5.1. Magnetometer	155
1.5.2. Gyroscope	156
1.5.3. Accelerometer.....	157
1.5.4. Kalman Filter	157
1.6. FSR method.....	Error! Bookmark not defined.
1.7. Load Cell method	Error! Bookmark not defined.
1.8. Result	Error! Bookmark not defined.
Chapter 2 – Mobile Application	160
2.1. JAVA Classes	161
2.2. Application Main Features.....	170
Chapter 3 – Remote MySQL Database	172
Chapter 4 – PHP Files	174
References.....	176

Figures

Figure 1.1 – Libraries used in SmartCrutch system	152
Figure 1.2 – Objects and variables creation	153
Figure 1.3 – Setup method.....	154
Figure 1.4 – Loop method	155
Figure 1.5 – IMU method - Magnetometer	155
Figure 1.6 – LSM303D datasheet magnetic sensitivity.....	156
Figure 1.7 – IMU method – Gyroscope.....	156
Figure 1.8 – L3GD20H datasheet gyroscope sensitivity.....	156
Figure 1.9 – IMU method – Accelerometer	157
Figure 1.10 – IMU method – Kalman Filter.....	157
Figure 1.11 – Kalman Filter library getAngle function.....	158
Figure 1.12 – Transmission of final IMU values	158
Figure 1.13 – FSR method.....	159
Figure 1.14 – Load Cell method.....	160
Figure 2.1 – Physiotherapist Register JAVA class.....	163
Figure 2.2 – Physiotherapist register button	164
Figure 2.3 – Physiotherapist local database data insert	165
Figure 2.4 – ConnectedThread class.....	166
Figure 2.5 – ConnectedThread run method	167
Figure 2.6 – BackgroundWorkerLogin class.....	168
Figure 2.7 – BackgroundWorkerLogin output and input stream.....	169
Figure 2.8 – BackgroundWorkerLogin response	169
Figure 3.1 – MySQL Session table.....	172
Figure 3.2 – MySQL Physiotherapist table	172
Figure 3.3 – MySQL Patient table.....	173
Figure 3.4 – Database diagram	173
Figure 4.1 – PhysioSync PHP script.....	174
Figure 4.2 – PhysioSync PHP script follow up	175

Tables

Table 1 – Libraries description.....	152
Table 2 – Activity package classes.....	161
Table 3 – Bluetooth package classes.....	165
Table 4 – Connection package classes.....	168
Table 5 – Extras package classes.....	170
Table 6 – Model package classes.....	170
Table 7 – SQLite package classes.....	170
Table 8 – PHP files.....	174

Technical Manual

This manual aims to present and explain the existing different technical features of the project developed. Chapter 1 focuses on explaining the Arduino code developed that will be used in the SmartCrutch system. Chapter 2 explains the main functions of the mobile application developed, IoPhyr. Chapter 3 explains the database created in order to store the remote information in the server. Chapter 4 explains the PHP files used and stored in the server to provide data synchronization.

Chapter 1 – Arduino application

This chapter explains the libraries used and the code written in the Arduino application that was used in the SmartCrutch system. The version of the Arduino IDE used to develop the application was the Arduino IDE 1.6.9.

1.1. Libraries

Libraries are commonly used in code development, which provide extra functionalities through built in functions and methods. In Arduino IDE, an include statement must be made in order for the project to recognize the installed library. The libraries used in the project are visible in Figure 1.1.

```
#include <Wire.h>  
#include <LSM303.h>  
#include <L3G.h>  
#include "Kalman.h"  
#include <SoftwareSerial.h>  
#include <ResponsiveAnalogRead.h>
```

Figure 1.1 – Libraries used in SmartCrutch system

Table 1 indicates the description of each library used in the project. Wire.h and SoftwareSerial.h are the only libraries that are included in the Arduino IDE and do not need to be downloaded.

Table 1 – Libraries description

Library	Description
Wire.h	Allows I2C communication.
LSM303.h	Interfaces with the compass and accelerometer sensors in the IMU.
L3G.h	Interfaces with the gyroscope in the IMU.
Kalman.h	Provides the use of the Kalman filter by using the values provided by the IMU.
SoftwareSerial.h	Allows serial communication on other digital pins of the Arduino (besides pin 0 and 1).
ResponsiveAnalogRead.h	Reduces large amounts of noise when reading signals without decreasing responsiveness.

1.2. Objects and variables

After including the libraries in the project, several objects and variables must be created, as shown in Figure 1.2. All these objects and variables are set before the setup() method.

```
SoftwareSerial BT(3, 2); //RX, TX 1

//IMU SENSORS 2
LSM303 compass;
L3G gyro;

//KALMAN FILTER 3
Kalman kalmanX;
Kalman kalmanY;
Kalman kalmanZ;

//IMU 4
float accelXraw, accelYraw, accelZraw, accelXmg, accelYmg, accelZmg, accelXg, accelYg, accelZg, accelXms2, accelYms2, accelZms2;
float magXraw, magYraw, magZraw, magXmgauss, magYmgauss, magZmgauss, magXgauss, magYgauss, magZgauss;
float gyroXraw, gyroYraw, gyroZraw, gyroXmdps, gyroYmdps, gyroZmdps, gyroXdps, gyroYdps, gyroZdps;

//IMU FINAL RESULT VALUES
float accelXroll, accelYpitch, angleXroll, angleYpitch;

//FSR 5
int fsrPin1 = 1;
int fsrPin2 = 2;
int fsrReading1, fsrReading2;
float fsrVoltage1, fsrVoltage2, fsrWeight1, fsrWeight2, fsrForce1, fsrForce2, fsrForceMed;

//LOAD CELL 6
int loadCellPin3 = 3;
int loadCellReading;
float loadCellVoltage, loadCellWeight;

unsigned long timer; 7

ResponsiveAnalogRead analog1(A1, true); //true = sleep enable 8
ResponsiveAnalogRead analog2(A2, true);
ResponsiveAnalogRead analog3(A3, true);
```

Figure 1.2 – Objects and variables creation

1. A Software Serial object named BT is created, with the first value being the pin (3) which will receive the serial data and the second value being the pin (2) that will transmit the serial data;
2. Objects of the accelerometer and magnetometer LSM303 and the gyroscope L3G are created;
3. Kalman objects regarding the three axis are created;
4. Many variables regarding the IMU values are created;
5. Several variables are needed for receiving and calculation values of the FSR sensors;
6. Just like the FSR, the load cell sensor also needs variables to be created for further calculations;
7. A timer variable is created that will be used when using the Kalman filter;
8. Three Responsive Analog Read objects are created with the sole purpose of reading the analog signals from the sensors while reducing the noise as much as possible.

1.3. Setup method

This method is only called once after the Arduino has been powered on or has been reset. Several objects and variables that were created before have to be instantiated in this method. This method is visible in Figure 1.3.

```
void setup()
{
  Serial.begin(9600);
  BT.begin(9600);
  Wire.begin();
  compass.init();
  compass.enableDefault();
  gyro.enableDefault();

  kalmanX.setAngle(180); // Set starting angle
  kalmanY.setAngle(180);

  timer = micros();

  compass.m_min = (LSM303::vector<int16_t>){-2617, -2666, -2121};
  compass.m_max = (LSM303::vector<int16_t>){+1626, +1831, +1877};

  if (!gyro.init())
  {
    Serial.println("Failed to autodetect gyro type!");
    while (1);
  }
}
```

Figure 1.3 – Setup method

1. Two objects are instantiated for serial communication at a 9600 baud rate. The first one is for debugging purposes on the computer, while the second was an object created before through the SoftwareSerial library that will be in charge of communication with the Bluetooth adapter;
2. Several objects created before are now instantiated;
3. Values of the software calibration of the accelerometer LSM303;
4. Simple verification that checks if the IMU is working and connected properly. If not, the program doesn't start.

1.4. Loop method

This method will start just after the setup method and will run continuously until the Arduino is powered off or reset.

```
void loop() {  
  startIMU();  
  startFSR();  
  startLoadCell();  
  BT.println();  
  Serial.println();  
  delay(200);  
}
```

Figure 1.4 – Loop method

Figure 1.4 shows three methods that process the values received by the three sensors used are called. After that a break line is used and a delay of 200 is called, meaning information will be transmitted every 200 milliseconds.

1.5. IMU method

This method reads the raw values of the IMU sensor, converts them into values that are recognizable and then uses those values with the help of a Kalman filter function to reach the final values of the roll and pitch angles to determine the correct horizontal and vertical inclination of the SmartCrutch.

1.5.1. Magnetometer

Calculations regarding the magnetometer can be seen in Figure 1.5.

```
void startIMU() {  
  compass.read();  
  gyro.read();  
  
  magXraw = compass.m.x;  
  magYraw = compass.m.y;  
  magZraw = compass.m.z;  
  
  if(magXraw < 0) {  
    magXraw = -magXraw;  
  }  
  magXmgauss = magXraw * 0.160;  
  
  if(magYraw < 0) {  
    magYraw = -magYraw;  
  }  
  magYmgauss = magYraw * 0.160;  
  
  if(magZraw < 0) {  
    magZraw = -magZraw;  
  }  
  magZmgauss = magZraw * 0.160;  
  
  magXgauss = magXmgauss / 1000;  
  magYgauss = magYmgauss / 1000;  
  magZgauss = magZmgauss / 1000;  
}
```

Figure 1.5 – IMU method - Magnetometer

1. Read methods are called in order to read the values from the accelerometer, magnetometer and gyroscope;
2. Previously created variables are now instantiated and will store raw values of the magnetometer;

3. Normalizing and converting raw values to milli Gauss. According to the LSM303D datasheet [1] the M_GN specification states a conversion factor of 0.160 mgauss/LSB with its default full scale setting of +/- 4 gauss (see Figure 1.6);
4. Conversion from milli Gauss to Gauss.

M_GN	Magnetic sensitivity	Magnetic FS=±2gauss	0.080	mgauss/ LSB
		Magnetic FS=±4gauss	0.160	
		Magnetic FS=±8gauss	0.320	
		Magnetic FS=±12gauss	0.479	

Figure 1.6 – LSM303D datasheet magnetic sensitivity [1]

1.5.2. Gyroscope

Gyroscope values also need to be converted to real values for later use (See Figure 1.7).

```
//GYROSCOPE
gyroXraw = (int) gyro.g.x;
gyroYraw = (int) gyro.g.y;
gyroZraw = (int) gyro.g.z;

gyroXmdps = gyroXraw * 8.75;
gyroYmdps = gyroYraw * 8.75;
gyroZmdps = gyroZraw * 8.75;

gyroXdps = gyroXmdps / 1000;
gyroYdps = gyroYmdps / 1000;
gyroZdps = gyroZmdps / 1000;
```

Figure 1.7 – IMU method – Gyroscope

1. Casting the raw values read by the gyroscope to integers and storing them into variables created;
2. Converting raw values to degrees per second. According to the L3GD20H datasheet [2] the So specification states a conversion factor of 8.75 mdps/LSB with its default full scale setting of +/- 245 dps (see Figure 1.8);
3. Converting milli degrees per second to degrees per second.

Symbol	Parameter	Test condition	Min.	Typ. ⁽¹⁾	Max.	Unit
FS	Measurement range	User selectable		±245 ±500 ±2000		dps
So	Sensitivity			8.75 17.50 70.00		mdps/digit

Figure 1.8 – L3GD20H datasheet gyroscope sensitivity [2]

1.5.3. Accelerometer

Finally, values retrieved from the accelerometer must be processed as seen in Figure 1.9.

```
//ACCELEROMETER
accelXraw = compass.a.x;
accelYraw = compass.a.y;
accelZraw = compass.a.z;

accelXmg = accelXraw / 16;
accelYmg = accelYraw / 16;
accelZmg = accelZraw / 16;

accelXg = accelXmg / 1000;
accelYg = accelYmg / 1000;
accelZg = accelZmg / 1000;

accelXms2 = 9.8 * accelXg;
accelYms2 = 9.8 * accelYg;
accelZms2 = 9.8 * accelZg;

accelYpitch = (atan2(accelXg, sqrt(accelYg*accelYg + accelZg*accelZg))*180.0)/M_PI;
accelXroll = (atan2(-accelYg, accelZg)*180.0)/M_PI;
```

Figure 1.9 – IMU method – Accelerometer

1. Previously created variables are now instantiated and will store raw values of the accelerometer;
2. The acceleration data registers contain a left aligned 12-bit number, meaning the lowest 4 bits are always 0. The values should be shifted right by 4 bits (divided by 16) to be consistent with the conversion factors specified in the datasheets;
3. According to the LSM303D datasheet [1] the So specification states a conversion factor of 1 mg/digit with its default full scale setting of +/- 2 g. Conversion from g to milli g is done here;
4. Conversion from milli g to milliseconds;
5. Calculating the acceleration of the pitch and roll inclination in angles without the Kalman filter.

1.5.4. Kalman Filter

Previous results are now used in order to calculate the pitch and roll angle with the help of the Kalman filter library (see Figure 1.10).

```
//FINAL RESULTS - KALMAN FILTER
angleYpitch = kalmanY.getAngle(accelYpitch, gyroYdps, (double)(micros() - timer));
angleXroll = kalmanX.getAngle(accelXroll, gyroXdps, (double)(micros() - timer));
float heading = compass.heading();

timer = micros();
```

Figure 1.10 – IMU method – Kalman Filter

By using the function available in the Kalman Filter library (see Figure 1.11) it is possible to reach a final value for the angles with reduced error, removing noise and drift over time.

```
// The angle should be in degrees and the rate should be in degrees per second  
float getAngle(float newAngle, float newRate, float dt);
```

Figure 1.11 – Kalman Filter library getAngle function

Besides the values calculated before using the L3G and LSM303 library, a delta time is also needed, by calculating the difference between readings in micro seconds and casting that difference to double for more precision. The yaw angle is simply provided by the magnetometer, providing values from 0° to 360° simply for understanding the changes of direction to the SmartCrutch.

Finally, these values are transmitted to the Bluetooth adapter and to the console for debugging purposes, with each value being split by a semi colon (see Figure 1.12).

```
BT.print(angleXroll);  
BT.print(";");  
BT.print(angleYpitch);  
BT.print(";");  
BT.print(heading);  
BT.print(";");  
Serial.print(angleXroll);  
Serial.print(";");  
Serial.print(angleYpitch);  
Serial.print(";");  
Serial.print(heading);  
Serial.print(";");  
}
```

Figure 1.12 – Transmission of final IMU values

1.6. FSR method

This method will process the values received by both FlexiForce sensors and send them to the Bluetooth adapter (see Figure 1.13).

```
void startFSR(){  
  
  //ANALOG 1  
  analogRead(fsrPin1);  
  analog1.update();  
  fsrReading1 = analog1.getValue();  
  fsrVoltage1 = fsrReading1 * (5000 / 1023.0);  
  
  //ANALOG 2  
  analogRead(fsrPin2);  
  analog2.update();  
  fsrReading2 = analog2.getValue();  
  fsrVoltage2 = fsrReading2 * (5000 / 1023.0);  
  
  //CALIBRATION  
  fsrWeight1 = ((fsrVoltage1 + 60.42) / 75.279);  
  fsrWeight2 = ((fsrVoltage2 + 34.574) / 82.849);  
  
  //OFFSET  
  fsrWeight1 = fsrWeight1 - 0.9;  
  fsrWeight2 = fsrWeight2 - 0.7;  
  if(fsrWeight1 < 0){  
    fsrWeight1 = 0;  
  }  
  if(fsrWeight2 < 0){  
    fsrWeight2 = 0;  
  }  
  
  //FORCE CALC  
  fsrForce1 = fsrWeight1 * 9.8;  
  fsrForce2 = fsrWeight2 * 9.8;  
  
  //MEDIAN FORCE CALC  
  fsrForceMed = fsrForce1 + fsrForce2;  
  fsrForceMed = fsrForceMed/2;  
  
  BT.print(fsrForceMed);  
  BT.print(";");  
  Serial.print(fsrForceMed);  
  Serial.print(";");  
}
```

Figure 1.13 – FSR method

1. Each force sensor will read the value of its pin through the analogRead method and the object created previously will be updated in each loop while the value of the object will be set to a reading value. The analog reading value is then converted to a voltage value. (ADC of the Arduino is 10 bits; $2^{10} = 1024$, but we don't count zero, so 1023. Arduino works at a voltage of $5V = 5000mV$;
2. Calibration previously done is now calculated into a new value that provides weight value in kilograms;
3. Offset value of each sensor is removed while also making sure that no values below 0 are sent;
4. Weight is transformed into force by using Newton's second law;
5. Median force of both forces is calculated;
6. Median force value is sent to the Bluetooth adapter and to the console with a semi colon added for splitting purposes.

1.7. Load Cell method

This method will process the value received by the load cell sensor and send it to the Bluetooth adapter (see Figure 1.14).

```
void startLoadCell(){  
    analogRead(loadCellPin3);  
    analog3.update();  
    loadCellReading = analog3.getValue();  
    loadCellVoltage = map(loadCellReading, 0, 1023, 0, 5000);  
    //CALIBRATION  
    loadCellWeight = ((loadCellVoltage + 1.5156) / 30.189);  
    //OFFSET  
    loadCellWeight = loadCellWeight - 2.0;  
    if(loadCellWeight < 0){  
        loadCellWeight = 0;  
    }  
    BT.print(loadCellWeight);  
    Serial.print(loadCellWeight);  
}
```



Figure 1.14 – Load Cell method

1. This part of code works just like the FSR method. The map method however, reads the analog values and maps them from 0 to 1023. The result is the same, just done differently by using a built-in function;
2. Calibration previously done is used, providing a value in kilograms;
3. Offset values are removed, ensuring that a negative value is never transmitted;
4. The result is transmitted to the Bluetooth adapter and sent to the console.

1.8. Result

The resulting string contains the value of the roll angle, pitch angle, yaw angle, median force by both FSR sensors and the load cell sensor. All the values are split by semi colons, with exception of the last inserted value, which breaks to a new line after it has been sent. An example of the final string sent to the Bluetooth adapter is seen below:

18.98;25.75;76.45;15.65;19.14
(roll, pitch, yaw, FSR, load cell)

Chapter 2 – Mobile Application

This chapter explains the main functions of the IoPhyr application developed for Android mobile devices, in terms of programming. It also explains the description of all

the classes used in the application. The application was developed in Android Studio version 3.1.4.

2.1. JAVA Classes

Several classes were created in order to successfully code the application. Groups of classes, also known as packages, were created to provide an organized project. In this topic we briefly go through all the JAVA classes that are contained in different packages by using the following tables and figures. Six packages were created in total. An activity package, containing all the activity classes of the application, which allow user interaction by connecting the class to a layout XML file; a Bluetooth package that contains the classes used in the live session data transmission; a connection package that contains the classes used when synchronization with the remote server is done; an extra package containing classes that are used to support the application itself with extra features; a model package containing the classes used to indicate which data should be stored where; and finally a SQLite package containing the class of the local database used.

Table 2 – Activity package classes

Class Name (.java)	Layout (.xml)	Description
LoginActivity	activity_login	Allows the physiotherapist to login in the application.
PhysioRegisterActivity	activity_register	Allows a physiotherapist to register in the application.
UserAreaActivity	activity_user_area	First activity after the login operation. Allows the physiotherapist to navigate in the application.
PhysioProfileActivity	activity_physio_profile	Allows the physiotherapist to view and edit his data.
PatientsActivity	activity_patients	Allows the physiotherapist to either create or view a patient.
PatientsRegisterActivity	activity_patients_register	Allows the physiotherapist to create a patient.
PatientsViewActivity	activity_patients_view	Allows the physiotherapist to view any of his patients.

SessionsViewActivity	activity_sessions_view	Allows the physiotherapist to select a patient regarding sessions training.
SessionsActivity	activity_sessions	Allows the physiotherapist to either view or start a session.
SessionsViewMenuActivity	activity_sessions_view_menu	Allows the physiotherapist to view a single session or the overall of all the sessions.
SessionsViewAllActivity	activity_sessions_view_all	Allows the physiotherapist to view data regarding all the sessions of a patient.
SessionsViewTableActivity	activity_sessions_view_table	Allows the physiotherapist to choose a single session from a table of previous sessions done by the patient.
SessionsViewSingleActivity	activity_sessions_view_single	Allows the physiotherapist to view data regarding a single session of a patient.

```

20
21 public class PhysioRegisterActivity extends AppCompatActivity {
22     SQLiteOpenHelper openHelper;
23     SQLiteDatabase db;
24
25     EditText etFirstName,etLastName,etUsername,etPassword,etEmail,etAge,etPhone;
26     Button bRegister,bCancel;
27     AlertDialog alert;
28
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_register);
34
35         openHelper = new DatabaseHelper( context: this);
36
37         alert = new AlertDialog();
38
39         etFirstName = (EditText) findViewById(R.id.etFirstName);
40         etLastName = (EditText) findViewById(R.id.etLastName);
41         etUsername = (EditText) findViewById(R.id.etUsername);
42         etPassword = (EditText) findViewById(R.id.etPassword);
43         etEmail = (EditText) findViewById(R.id.etEmail);
44         etAge = (EditText) findViewById(R.id.etAge);
45         etPhone = (EditText) findViewById(R.id.etPhone);
46         bRegister = (Button) findViewById(R.id.bView);
47         bCancel = (Button) findViewById(R.id.bCancel);
48         bCancel.setOnClickListener(new View.OnClickListener() {
49             @Override
50             public void onClick(View v) {
51                 finish();
52             }
53         });
54     }
55

```

Figure 2.1 – Physiotherapist Register JAVA class

In Figure 1.15 an example of an activity where the physiotherapist must register is shown. The onCreate method runs only once when the activity starts. This is where the connection with the XML file is done, by calling the setContentView method. The database must also be initialized in order to store the data inserted regarding the physiotherapist register, while also initializing several variables, such as buttons and edit texts that are used as an input to the physiotherapist. These variables must be connected to components present in the corresponding XML file by its respective ID, by using the findViewById method.

```
57 public void OnReg(View view){
58     db=openHelper.getWritableDatabase();
59
60     String str_FirstName = etFirstName.getText().toString();
61     String str_LastName = etLastName.getText().toString();
62     String str_Username = etUsername.getText().toString();
63     String str_Password = etPassword.getText().toString();
64     String str_Email = etEmail.getText().toString();
65     String str_Age = etAge.getText().toString();
66     String str_Phone = etPhone.getText().toString();
67     String type = "register";
68
69     String selectQuery = "SELECT * FROM " + DatabaseHelper.TABLE_PHYSIO + " WHERE " + DatabaseHelper.PHYSIO_USERNAME + "=?";
70     Cursor cursor = db.rawQuery(selectQuery, new String [] {str_Username});
71
72     if (cursor.getCount() > 0) {
73         cursor.moveToFirst();
74         showAlert(test:"Username already exists");
75         return;
76     }
77     if(str_Email.isEmpty() || str_Password.isEmpty() || str_Age.isEmpty() || str_FirstName.isEmpty() ||
78         str_LastName.isEmpty() || str_Username.isEmpty() || str_Phone.isEmpty()){
79         showAlert(test:"All the fields need to be filled");
80         return;
81     }
82     else{
83         String uniqueId;
84         uniqueId = UUID.randomUUID().toString();
85         insertData(uniqueId, str_FirstName, str_LastName, str_Username, str_Password, str_Email, str_Age, str_Phone);
86
87         Intent registerIntent = new Intent(packageContext: PhysioRegisterActivity.this, LoginActivity.class);
88         registerIntent.putExtra(name:"FROM_ACTIVITY", value:"register");
89         PhysioRegisterActivity.this.startActivity(registerIntent);
90         finish();
91     }
92 }
```

Figure 2.2 – Physiotherapist register button

The method in charge of making sure the physiotherapist is properly registered in the application is the OnReg method (see Figure 1.16). This method starts by indicating the database that it can be editable and then reads all the values inserted by the physiotherapist in the edit texts and converts them to strings. A query to the database is then made, where it verifies that the data inserted in the username field in the table PHYSIO doesn't already exist. This is done with the help of a cursor object that points to data existing in a database. Several verifications are made, providing the physiotherapist with different error types, such as "Username already exists", indicating that the username written already exists in the database, and "All the fields need to be filled", in case the physiotherapist left a blank field. If everything is done correctly, a random UUID is attributed to the new physiotherapist. All the data is then stored in the local database by using the insertData method that is shown in Figure 1.17. Intents are used to travel from one activity to another, so when the register is successfully done the physiotherapist is sent back to the login activity.

```

94     public boolean insertData(String id, String str_FirstName, String str_LastName, String str_Username,
95                             String str_Password, String str_Email, String str_Age, String str_Phone){
96
97         ContentValues contentValues = new ContentValues();
98
99         contentValues.put(DatabaseHelper.PHYSIO_ID, id);
100        contentValues.put(DatabaseHelper.PHYSIO_FIRSTNAME, str_FirstName);
101        contentValues.put(DatabaseHelper.PHYSIO_LASTNAME, str_LastName);
102        contentValues.put(DatabaseHelper.PHYSIO_USERNAME, str_Username);
103        contentValues.put(DatabaseHelper.PHYSIO_PASSWORD, str_Password);
104        contentValues.put(DatabaseHelper.PHYSIO_EMAIL, str_Email);
105        contentValues.put(DatabaseHelper.PHYSIO_AGE, str_Age);
106        contentValues.put(DatabaseHelper.PHYSIO_PHONE, str_Phone);
107
108        long result = db.insert(DatabaseHelper.TABLE_PHYSIO, nullColumnHack, contentValues);
109
110        if(result == -1)
111        {
112            alert.showDialogFail( activity: this, msg: "Register failed");
113            return false;
114        } else
115            return true;
116        }
117
118
119    public void failAlert(String text){
120        alert.showDialogFail( activity: this, text);
121    }
122 }
    
```

Figure 2.3 – Physiotherapist local database data insert

The insertData method is responsible for storing the data inserted by the physiotherapist in the local database so it can be accessed by the application. A content value is created, which allows information to be stored inside an object in the form of key-value. This object can then be passed to the insert method of an instance of the local database which will either return true (successfully inserted the data in the database) or false (an error occurred inserting the data in the database). An instance of a custom dialog class created is called in the failAlert method, showing a pop-up window to the user with a custom message.

Table 3 – Bluetooth package classes

Class Name (.java)	Layout (.xml)	Description
BluetoothActivity	activity_bluetooth	Allows the physiotherapist to start a new session by connecting the mobile device to the SmartCrutch system.
BluetoothConnectionService	-	Responsible for the creation of threads, sockets and data streams. Information sent by the Bluetooth adapter is also decoded in this class.
DeviceListAdapter	-	Responsible for creating a list that shows the discoverable devices nearby with Bluetooth enabled.

```
217 public class ConnectedThread extends Thread{
218     private final BluetoothSocket mmSocket;
219     private final InputStream mmInStream;
220     private final OutputStream mmOutStream;
221
222     public ConnectedThread(BluetoothSocket socket) {
223         Log.d(TAG, msg: "ConnectedThread starting");
224         mmSocket = socket;
225         InputStream tmpIn = null;
226         OutputStream tmpOut = null;
227
228         try {
229             mProgressDialog.dismiss();
230         } catch (NullPointerException e){
231             e.printStackTrace();
232         }
233         try {
234             tmpIn=mmSocket.getInputStream();
235             tmpOut=mmSocket.getOutputStream();
236         } catch (IOException e) {
237             e.printStackTrace();
238         }
239         mmInStream=tmpIn;
240         mmOutStream=tmpOut;
241     }
242 }
```

Figure 2.4 – ConnectedThread class

The BluetoothConnectionService class is in charge of creating the threads, sockets and stream inputs in order to receive data from the SmartCrutch system. An inner class, ConnectedThread, is in charge of receiving the data, storing it in variables, and sending it to an activity class through the use of intents, in order to have real time data analysis. In Figure 1.18 the ConnectedThread constructor is visible, where it uses local sockets and local input and output streams. A progress dialog is shown when trying to establish a connection, but as soon as the connection is established and data starts to flow, the progress dialog is dismissed.

```
public void run(){
    byte[] buffer = new byte[1024];
    int bytes=0;
    long startTime = System.currentTimeMillis();
    Intent startingIntent = new Intent( action: "incomingData");
    startingIntent.putExtra( name: "inc", value: "ok");
    LocalBroadcastManager.getInstance(mContext).sendBroadcast(startingIntent);

    while(true){
        //read from InputStream
        try {
            String incomingMessage = "";
            buffer[bytes] = (byte) mmInStream.read();
            if(buffer[bytes] == '\n') {
                incomingMessage = new String(buffer, offset: 0, bytes);
                Timestamp timestamp = new Timestamp(System.currentTimeMillis());
                String timestampString = timestamp.toString();

                long elapsedTime = (System.currentTimeMillis() - startTime) ;
                long elapsedSeconds = elapsedTime / 1000;
                long timeSeconds = TimeUnit.MILLISECONDS.toSeconds(elapsedTime);
                double elapsedTime1 = (double) elapsedTime;
                double timeseconds = elapsedTime1 /1000.0;
                double timeminutes = timeseconds / 60;
                double timeminutesrounded = Math.round(timeminutes*1e3)/1e3;

                Intent incomingMsgIntent = new Intent( action: "incomingMessage");
                incomingMsgIntent.putExtra( name: "timestamp", timestampString);
                incomingMsgIntent.putExtra( name: "sensorValues", incomingMessage);
                incomingMsgIntent.putExtra( name: "time", timeminutesrounded);
                incomingMsgIntent.putExtra( name: "inc", value: "ok");
                LocalBroadcastManager.getInstance(mContext).sendBroadcast(incomingMsgIntent);
                bytes=0;
            }else{
                bytes++;
            }
        } catch (IOException e) {
```

Figure 2.5 – ConnectedThread run method

Every time a thread is started the run method is called. A buffer of bytes is created in order to store the information received by the SmartCrutch and an intent is sent to the BluetoothActivity class which indicates that data is now being received. Data transmission is done with the help of a LocalBroadcastManager object. The information coming from the input stream will now be stored in the buffer, making sure each data set is properly received by including a new line verification. Several information is then sent through the LocalBroadcastManager, such as the timestamp of the data received, the string sent by the SmartCrutch system (see Chapter 1, topic 1.8) and time in minutes with three decimal cases. After each data set is sent to the BluetoothActivity class, the bytes in the buffer are reset to zero.

Table 4 – Connection package classes

Class Name (.java)	Description
BackgroundWorker	Responsible for establishing a HTTP connection with the PHP files located in the remote server in order to synchronize patients related to the current physiotherapist and all the sessions related to his patients.
BackgroundWorkerLogin	Responsible for establishing a HTTP connection with the PHP files located in the remote server in order to synchronize the physiotherapists.

```

38 public class BackgroundWorkerLogin extends AsyncTask<String,Void,String> {
39     Context context;
40     SQLiteOpenHelper openHelper;
41     SQLiteDatabase db;
42     AlertDialog mProgressDialog;
43
44     public BackgroundWorkerLogin (Context ctx) {
45         context = ctx;
46         openHelper = new DatabaseHelper(context);
47     }
48
49     @Override
50     protected String doInBackground(String... params) {
51         String type = params[0];
52         String physioSync_url = "http://51.255.38.88/SmartCrutch/physioSync.php";
53         if(type.equals("teste")) {
54             try {
55                 URL url = new URL(physioSync_url);
56                 HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
57                 httpURLConnection.setRequestMethod("POST");
58                 httpURLConnection.setDoOutput(true);
59                 httpURLConnection.setDoInput(true);
60                 OutputStream outputStream = httpURLConnection.getOutputStream();
61                 BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, charsetName: "UTF-8"));
62                 String post_data, id, first_name, last_name, username, password, email, age, phone;
63                 post_data = "";
64                 for(Physiotherapist physio : getAllPhysios()){
65                     id = physio.getId();
66                     first_name = physio.getFirstName();
67                     last_name = physio.getLastName();
68                     username = physio.getUsername();
69                     password = physio.getPassword();
70                     email = physio.getEmail();
71                     age = physio.getAge();
72                     phone = physio.getPhone();

```

Figure 2.6 – BackgroundWorkerLogin class

In figure 1.20 the connection with the PHP files is done for data synchronization. This class is defined as an AsyncTask which enables proper use of the UI thread, allowing background operations to be made. Each time the synchronizing button in the login activity is clicked the doInBackground method is called. A HTTP connection is then established with the respective PHP file URL. An OutputStream is created in order to send information to the server. A for loop is called in order to get information regarding all the physiotherapists that exist in the local database.

```

72     phone = physio.getPhone();
73     post_data += URLEncoder.encode( id, "UTF-8")+"&"+URLEncoder.encode(id, enc: "UTF-8")+"&"
74     +URLEncoder.encode( id, "first_name[]", enc: "UTF-8")+"&"+URLEncoder.encode(first_name, enc: "UTF-8")+"&"
75     +URLEncoder.encode( id, "last_name[]", enc: "UTF-8")+"&"+URLEncoder.encode(last_name, enc: "UTF-8")+"&"
76     +URLEncoder.encode( id, "username[]", enc: "UTF-8")+"&"+URLEncoder.encode(username, enc: "UTF-8")+"&"
77     +URLEncoder.encode( id, "password[]", enc: "UTF-8")+"&"+URLEncoder.encode(password, enc: "UTF-8")+"&"
78     +URLEncoder.encode( id, "email[]", enc: "UTF-8")+"&"+URLEncoder.encode(email, enc: "UTF-8")+"&"
79     +URLEncoder.encode( id, "age[]", enc: "UTF-8")+"&"+URLEncoder.encode(age, enc: "UTF-8")+"&"
80     +URLEncoder.encode( id, "phone[]", enc: "UTF-8")+"&"+URLEncoder.encode(phone, enc: "UTF-8")+"&";
81
82     bufferedWriter.write(post_data);
83     bufferedWriter.flush();
84     bufferedWriter.close();
85     outputStream.close();
86     InputStream inputStream = httpURLConnection.getInputStream();
87     BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream, charsetName: "iso-8859-1"));
88     String result="";
89     String line="";
90     while((line = bufferedReader.readLine())!= null) {
91         result += line;
92     }

```

Figure 2.7 – BackgroundWorkerLogin output and input stream

Local physiotherapist data is then encoded via UTF-8 and sent to the PHP file through the use of a `BufferedWriter`. A `BufferedReader` will be responsible for receiving the response of the PHP file located in the server, which comes in JSON format.

```

94     try {
95         db = openHelper.getWritableDatabase();
96         db.delete(DatabaseHelper.TABLE_PHYSIO, whereClause: null, whereArgs: null);
97
98         String id_physio, firstname_physio, lastname_physio, username_physio, password_physio,
99             email_physio, age_physio, phone_physio;
100        JSONArray jsonArray = new JSONArray(result);
101
102        for(int i = 0 ; i < jsonArray.length(); i++){
103            JSONObject jsonObject = jsonArray.getJSONObject(i);
104
105            id_physio = jsonObject.getString( name: "physio_id");
106            firstname_physio = jsonObject.getString( name: "physio_firstname");
107            lastname_physio = jsonObject.getString( name: "physio_lastname");
108            username_physio = jsonObject.getString( name: "physio_username");
109            password_physio = jsonObject.getString( name: "physio_password");
110            email_physio = jsonObject.getString( name: "physio_email");
111            age_physio = jsonObject.getString( name: "physio_age");
112            phone_physio = jsonObject.getString( name: "physio_phone");
113
114            insertData(id_physio,firstname_physio,lastname_physio,username_physio,password_physio,
115                email_physio,age_physio, phone_physio);
116        }
117    } catch (JSONException e) {
118        e.printStackTrace();
119    }
120    bufferedReader.close();
121    inputStream.close();
122    httpURLConnection.disconnect();
123    db.close();
124    return result;
125 } catch (MalformedURLException e) {
126     e.printStackTrace();
127 } catch (IOException e) {
128     e.printStackTrace();
129 }
130 }
131 return null;

```

Figure 2.8 – BackgroundWorkerLogin response

In this case, since a physical therapy clinic shouldn't have too many physiotherapists, a different approach was made. Unlike the other class responsible for patient and sessions synchronization, this one will delete all the data in the physiotherapist table in the local database and replace them with new data that was stored in the server's database. The information is decoded in a JSON array and is quickly inserted correctly in the local database. After the whole process is done all the connections are closed.

Table 5 – Extras package classes

Class Name (.java)	Description
MyTouchListener	Shows a black overlay with transparency each time an icon is pressed.
NetworkChangeReceiver	Responsible for indicating the user when Internet connection is available. Synchronizing buttons will either be visible or not with the help of this class.
TimerFormatter	Represents the time visible in the session charts.
ViewDialog	Responsible for indicating the user with success, alert or fail pop-up windows.

Table 6 – Model package classes

Class Name (.java)	Description
Patient	Contains all the data related to a patient.
Physiotherapist	Contains all the data related to a physiotherapist.
Session	Contains all the data related to a session.

Table 7 – SQLite package classes

Class Name (.java)	Description
DatabaseHelper	Contains all the information that is stored in the local database.

2.2. Application Main Features

Several important features for the development of the mobile application are listed below:

- **Authentication** – In order to successfully login in the application, the physiotherapist must be registered in the local database. If the physiotherapist is already registered but the local database has no information related to him, synchronization must be made from the remote database to the local database simply by turning on Wi-Fi and clicking on the synchronize button.
- **HTTP Connection** – Communication from the mobile application to the database is done through HTTP requests. For this purpose, an AsyncTask class was used, allowing the user to perform background operations and publish results on the UI

thread. This class is ideally used for short operations, a few seconds at most, which makes it ideal for database synchronization.

- **Threads and Sockets** – Threads are commonly used in JAVA, which is known as being a multi-threaded programming language. A multi-threaded program consists on two or more parts of the program that run concurrently, while each part is assigned a different task. Threads are commonly used with sockets for communication purposes, such as client-server applications. Communication is done via TCP, which provides a reliable point-to-point communication channel between client-server applications. Both server and client must establish a connection with each other by binding a socket to the end of the connection. Socket classes (*ServerSocket* and *Socket*) are used in order to represent the connection between a server and a client. Creation of sockets and data stream information is done inside the created threads.

Chapter 3 – Remote MySQL Database

In order to have data synchronization in the system a remote database must be held in a remote server, which will communicate with the mobile application through PHP scripts with the help of HTTP protocol. Figures 3.1, 3.2 and 3.3 indicate the SQL code used to create the tables used in the system, Session, Physiotherapist and Patient, respectively.

```
CREATE TABLE IF NOT EXISTS `Session` (  
  `session_id` char(40) NOT NULL,  
  `session_patientid` text NOT NULL,  
  `session_physioid` text NOT NULL,  
  `session_name` tinytext NOT NULL,  
  `session_date` text NOT NULL,  
  `session_starttime` text NOT NULL,  
  `session_endtime` text NOT NULL,  
  `session_timer` text NOT NULL,  
  `session_fsrdata` longtext NOT NULL,  
  `session_rollxdata` longtext NOT NULL,  
  `session_pitchydata` longtext NOT NULL,  
  `session_yawzdata` longtext NOT NULL,  
  `session_loadcelldata` longtext NOT NULL,  
  `session_notes` text NOT NULL,  
  `session_median_fsr` text NOT NULL,  
  `session_median_load` text NOT NULL,  
  `session_median_roll` text NOT NULL,  
  `session_median_pitch` text NOT NULL,  
  `session_steps` text NOT NULL,  
  `session_timestamp` datetime NOT NULL,  
  PRIMARY KEY (`session_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 3.1 – MySQL Session table

```
CREATE TABLE IF NOT EXISTS `Physiotherapist` (  
  `physio_id` varchar(250) NOT NULL,  
  `physio_firstname` text NOT NULL,  
  `physio_lastname` text NOT NULL,  
  `physio_username` text NOT NULL,  
  `physio_password` text NOT NULL,  
  `physio_email` text NOT NULL,  
  `physio_age` text NOT NULL,  
  `physio_phone` text NOT NULL,  
  `patient_update` timestamp NULL DEFAULT NULL,  
  `session_update` timestamp NULL DEFAULT NULL,  
  PRIMARY KEY (`physio_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 3.2 – MySQL Physiotherapist table

```
CREATE TABLE IF NOT EXISTS `Patient` (  
  `patient_id` varchar(250) CHARACTER SET latin1 NOT NULL,  
  `patient_firstname` text NOT NULL,  
  `patient_lastname` text NOT NULL,  
  `patient_email` text CHARACTER SET latin1 NOT NULL,  
  `patient_age` text CHARACTER SET latin1 NOT NULL,  
  `patient_phone` text CHARACTER SET latin1 NOT NULL,  
  `patient_address` text CHARACTER SET latin1 NOT NULL,  
  `patient_datereg` datetime NOT NULL,  
  `patient_idphysio` text CHARACTER SET latin1 NOT NULL,  
  `patient_photo` longtext CHARACTER SET latin1,  
  `patient_weight` text CHARACTER SET latin1,  
  `patient_height` text CHARACTER SET latin1,  
  `patient_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP  
  ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (`patient_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figure 3.3 – MySQL Patient table

The resulting diagram of the database used can be seen in Figure 3.4.

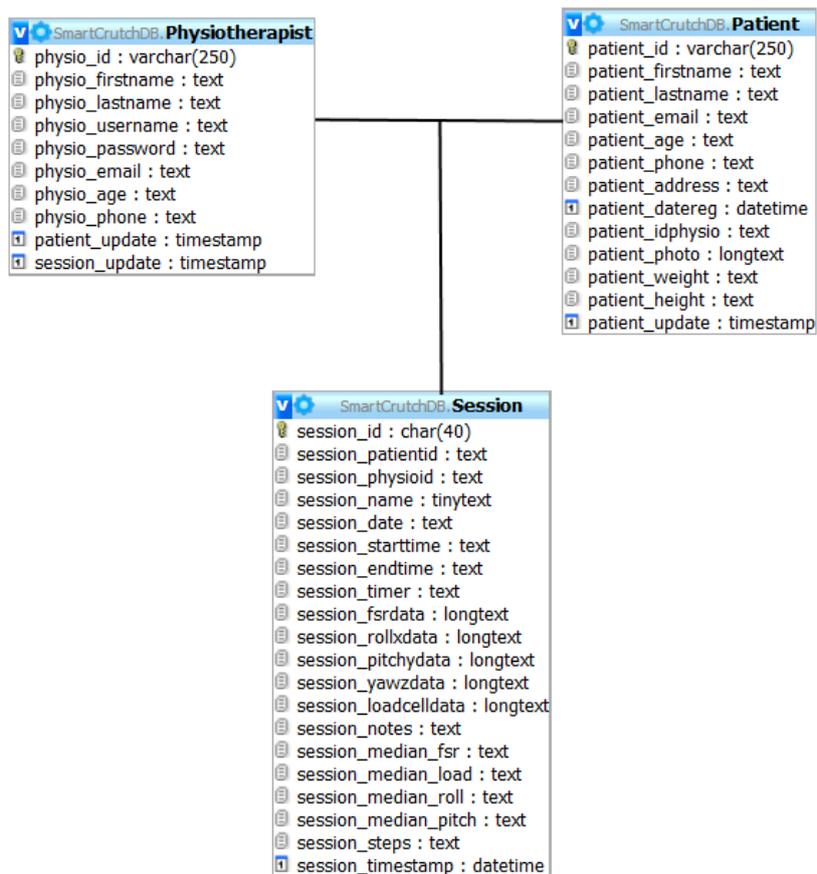


Figure 3.4 – Database diagram

Chapter 4 – PHP Files

In order to have communication between the server and the mobile application, PHP files are used. These files are stored in the server and every time a physiotherapist synchronizes his application, an HTTP connection is established to the server, through the use of the PHP files. Three files were used in total for this purpose, which can be seen in Table 8.

Table 8 – PHP files

File Name (.php)	Description
physioSync	Synchronizes remote and local database data related to all existing physiotherapists.
patientsSync	Synchronizes remote and local database data related to all existing patients that are related to the physiotherapist that is currently logged in.
sessionSync	Synchronizes remote and local database data related to all existing sessions that are related to the patients of the physiotherapis that is currently logged in.

```

1 <?php
2 ini_set('display_errors', 1);
3 ini_set('display_startup_errors', 1);
4 error_reporting(E_ALL);
5
6 $con = mysqli_connect(" Server IP ", "User", "Password", "SmartCrutchDB");
7 if(isset($_POST['id'][0])){
8
9     foreach($_POST["id"] as $key => $physio){
10         $physio_id = $_POST["id"][$key];
11         $physio_firstname = $_POST["first_name"][$key];
12         $physio_lastname = $_POST["last_name"][$key];
13         $physio_username = $_POST["username"][$key];
14         $physio_password = $_POST["password"][$key];
15         $physio_email = $_POST["email"][$key];
16         $physio_age = $_POST["age"][$key];
17         $physio_phone = $_POST["phone"][$key];
18
19         $statement = mysqli_prepare($con, "INSERT INTO Physiotherapist (physio_id, physio_firstname, physio_lastname,
20 physio_username, physio_password, physio_email, physio_age, physio_phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
21
22         mysqli_stmt_bind_param($statement, "sssssss", $physio_id, $physio_firstname, $physio_lastname,
23 $physio_username, $physio_password, $physio_email, $physio_age, $physio_phone);
24
25         mysqli_stmt_execute($statement);
26     }
27
28     mysqli_stmt_close($statement);
29

```

Figure 4.1 – PhysioSync PHP script

In figure 4.1 and 4.2, the PHP file in charge of synchronizing the physiotherapists is shown. Initial code, from line 2 to line 4 is used for debugging errors. A connection is then started, providing the IP of the server, username, password and database name, using

the *mysql_connect* method. Method in line 7 is used to check whether a variable is set or not, in this case the physiotherapist's ID, while also making sure they are not empty. Then, a *foreach* method is used to cycle through all the physiotherapists, while receiving the values sent from the mobile phone with the help of POST method. Prepared statements are then used for storing data into the remote database. In line 19, *mysqli_prepare* method receives the connection as first argument, and the SQL query as second argument, preparing it and returning a statement handle to be used for further operations on the statement. Variables must be bound to the prepared statement as parameters, using the *mysqli_stmt_bind_param* which receives the previous statement, type of variables sent (in this case Strings, 's') and the variables. Finally, the statement is executed in line 25 with the use of *mysqli_stmt_execute*, storing all variables in the server. When the loop has gone through all the physiotherapists, the statement is finally closed.

```
30
31     $sql = "SELECT * FROM Physiotherapist";
32     $result = $con->query($sql);
33
34     if ($result->num_rows > 0) {
35         // output data of each row
36         while($row = $result->fetch_assoc()) {
37             $users[] = $row;
38         }
39         echo json_encode($users);
40     } else {
41         $response = array();
42         $response["success"] = true;
43
44         echo json_encode($response);
45     }
46     $con->close();
47 ?>
```

Figure 4.2 – PhysioSync PHP script follow up

Figure 4.2 indicates how to retrieve data from the remote server in order to synchronize the local SQLite database. A query is made to retrieve all the data from the physiotherapist table (line 31) and if there is physiotherapist data in the table, the information will be encoded as a JSON message and sent to the mobile application for decoding. If there are no physiotherapists, it simply sends a message to the mobile phone, also encoded in JSON.

References

- 62- *Pololu.com*, 2018. [Online]. Available:
<https://www.pololu.com/file/0J703/LSM303D.pdf>. [Accessed: 14- Sep- 2018].
- 63- *Pololu.com*, 2018. [Online]. Available:
<https://www.pololu.com/file/0J731/L3GD20H.pdf>. [Accessed: 14- Sep- 2018].