



MOBILE HEALTH SYSTEM FOR
EVALUATION OF BREAST CANCER
PATIENTS DURING TREATMENT AND
RECOVERY PHASES

REALIZADO POR:
JOAQUÍN OLLERO GARCÍA

DIRIGIDO POR:
*ORESTI BAÑOS LEGRÁN, JOSÉ ANTONIO MORAL-MUÑOZ
E IGNACIO ROJAS RUÍZ*

DEPARTAMENTO:
ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

9 de septiembre de 2016

Mobile health system for evaluation of breast cancer patients during treatment and recovery phases

por

Joaquín Ollero García

PALABRAS CLAVE: Android, Android Wear, Servidor local, Servidor remoto, Reloj inteligente, Teléfono inteligente, Aplicaciones para dispositivos móviles, Aplicación web, Sensor de pulso cardíaco, Sensores cinemáticos, Reconocimiento de actividades, Rehabilitación de cáncer de mama.

RESUMEN: El cáncer de mama es el tumor más frecuente en las mujeres occidentales y estadísticamente 1 de cada 8 mujeres tendrá cáncer de mama a lo largo de su vida. Una vez superado, la etapa de rehabilitación que el paciente debe seguir es fundamental para su recuperación. En este trabajo se ha desarrollado un sistema compuesto de 3 aplicaciones, una para relojes inteligentes, otra para teléfonos inteligentes y una aplicación web. Las aplicaciones para dispositivos móviles están dirigidas al paciente que se encuentra en proceso de rehabilitación y le permitirá monitorizar parámetros de interés que indicarán si el proceso de rehabilitación que está siguiendo está haciendo que mejore su condición con respecto a la superación de la enfermedad. La aplicación web está dirigida a un experto médico con el objetivo de que pueda hacer un seguimiento de la rehabilitación que están llevando a cabo sus pacientes.

KEYWORDS: Android, Android Wear, Local server, Remote server, Smartwatch, Smartphone, Handheld devices applications, Web application, Heart rate sensor, Kinematics sensors, Activity recognition, Breast cancer rehabilitation.

ABSTRACT: Breast cancer is the most common tumor in western women and statistically 1 out of 8 women will develop breast cancer over their lifetime. Once overcome it, the stage of rehabilitation that the patient should follow is critical to recover from the suffered illness. In this paper, a system composed of 3 applications, one for smartwatches, one for smartphones and a web application, is presented. Applications for handheld devices are directed to the patient who is undergoing rehabilitation and allow to monitor parameters of interest that will indicate whether the rehabilitation process being followed is improving the health of the patient. The web application is directed to a medical expert with the objective of tracking rehabilitation conducted by the patients.

D. Oresti Baños Legrán, Profesor de la Universidad de Twente (Holanda) y D. Ignacio Rojas Ruiz, Profesor del departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada (España), como directores del Proyecto Fin de Carrera de D. Joaquín Ollero García

Informan:

que el presente trabajo, titulado:

Mobile health system for evaluation of breast cancer patients during treatment and recovery phases

Ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizamos a su presentación.

Granada, a 9 de septiembre de 2016

Fdo. _____

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Carrera se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 9 de septiembre de 2016

Oresti Baños Legrán (cotutor)

DNI:

Firma:

José Antonio Moral-Muñoz (cotutor)

DNI:

Firma:

Ignacio Rojas Ruíz (cotutor)

DNI:

Firma:

Joaquín Ollero García (autor)

DNI:

Firma:

A Emma y a ti, que todavía no lo sabes.

Acknowledgements

Me gustaría agradecer a todas las personas que me han apoyado en el desarrollo de este proyecto. A Ignacio Rojas por darme la ayuda necesaria y mostrarme su amabilidad. A Oresti Baños por su sabiduría, paciencia y ser todo un ejemplo como persona y tutor. A José Antonio Moral-Muñoz por aconsejarme y estar ahí en todo momento. A mis amigos, Luis, Laura, Rafa, Víctor y Javi y todos los demás que me han dado cariño y me han hecho seguir adelante. A mi madre, mi padre, mi hermano, mi hermana, mi sobrina, mis abuelas y mi tío por ser una familia maravillosa. A mi "familia" del Carmen de la Victoria por aguantarme cada día y finalmente a todas esas personas que han arrojado un rayo de luz en mi vida.

Contents

List of Figures	III
List of Tables	V
List of Listings	VI
1. Introduction	1
1.1. Motivation, Context and Problem Statement	1
1.2. Objectives	4
1.3. Thesis and Structure	4
2. State of the Art	6
2.1. mHealth Overview	6
2.2. mHealth Devices	8
2.3. mHealth Applications	12
2.4. mHealth Frameworks	14
3. System Design	16
3.1. Top Level View	16
3.2. Requirements	16
3.3. Methods	18
3.3.1. Heart Rate	18
3.3.2. Energy Expenditure	19
3.3.3. Arm Mobility	20
4. System Implementation	22
4.1. Overview	22
4.2. Smartwatch Application	23
4.2.1. System	23
4.2.2. Implementation	24
4.3. Smartphone Application	41
4.3.1. System	41
4.3.2. Implementation	42
4.4. Web Server Application	55
4.4.1. System	55
4.4.2. Implementation	58
4.5. System usage	71

5. System Test	73
5.1. Heart rate tests	73
5.2. Energy expenditure tests	74
5.3. Arm mobility tests	75
6. Conclusions	76
6.1. Aims achieved	76
6.2. Future work	76
References	79

List of Figures

1. Estimated new cancer cases and deaths in 2016 by cancer type, both sexes combined. American Cancer Society	1
2. Estimated worldwide smartwatch shipments. Strategy Analytics	2
3. The Magic Ring device	10
4. Jawbone UP and accompanying app	11
5. Jolt clip	11
6. DrawMD App	13
7. Top level view of the system	16
8. Smartwatch application waiting for configuration parameters with progress bar	27
9. Smartwatch application with configuration parameters received .	27
10. Heart rate frequency experiment with SENSOR DELAY FASTEST	29
11. Heart rate frequency experiment with SENSOR DELAY GAME	29
12. Heart rate frequency experiment with SENSOR DELAY UI . . .	30
13. Heart rate frequency experiment with SENSOR DELAY NORMAL	30
14. Heart rate frequency experiment with 5000000 microseconds . . .	31
15. Heart rate frequency experiment with 15000000 microseconds . .	31
16. Heart rate frequency experiment with 30000000 microseconds . .	32
17. Smartwatch application Heart Rate while monitoring	33
18. Smartwatch application Heart Rate when monitoring time has concluded	33
19. Smartwatch application Energy Expenditure while monitoring . .	35
20. Smartwatch application Energy Expenditure when monitoring time has concluded	35
21. Smartwatch application Arm Mobility while monitoring	37
22. Smartwatch application Arm Mobility when monitoring time has concluded	37
23. Smartphone application home screen	44
24. Smartphone application configuring parameters	45
25. Smartphone application receiving heart rate data	46
26. Smartphone application communication with web server application	52
27. Structure of the Web Server application website	60
28. Displaying all the patients of the database of patients	61
29. Formulary to add new patients on the web server application . .	63
30. Deleting patients through an intuitive button	64
31. Deployment of patients to select by clicking on picture	66

32.	Heart Rate, Energy Expenditure and Arm Mobility tabs with date selector	67
33.	Heart rate (average) chart	73
34.	Standard deviation chart	73
35.	Activity Recognition Heart Rate chart	74
36.	Activity counts (average) chart	74
37.	METs (average) chart	74
38.	Activity Recognition Energy Expenditure chart	75
39.	Arm mobility chart	75

List of Tables

1.	Patient database	61
2.	Heart Rate Test database	65
3.	Energy Expenditure Test database	65
4.	Arm Mobility Test database	66

List of Listings

1.	Reception of configuration parameters on smartwatch	25
2.	Heart Rate SensorManager	32
3.	Linear Acceleration SensorManager	34
4.	Linear Acceleration and Gyroscope SensorManager	36
5.	Sending heart rate data with the Wearable Data Layer API . . .	38
6.	Use of the Google API Client	39
7.	Requests and removes activity updates of the Activity Recognition API	49
8.	Activity Recognition pending intent and broadcast receiver . . .	49
9.	Builds a Map object to be sent with the Volley library	53
10.	Sending an object with the Volley library	53
11.	Flask basic example	56
12.	Connection to localhost and OpenShift and to the MongoDB database	58
13.	Git commands to upload changes to OpenShift	59
14.	MongoDB find instruction	61
15.	MongoDB insert instruction	62
16.	MongoDB remove instruction	63
17.	Request of data sent from the mobile application to the server .	64
18.	Retrieving test data from the database	68
19.	Building a lineChart with nvD3	69

1. Introduction

1.1. Motivation, Context and Problem Statement

Breast cancer (BC) is one of the most common cancers in the world and by far the most frequent cancer among women. It ranks as the fifth leading cause of death from cancer overall. Complications of BC therapies are common, like cardiovascular disease, which is fairly frequent and leads to morbidity, poor quality of life or premature mortality. [1]. There has been an appreciable reduction in the breast cancer mortality rate over the past two decades, especially among women younger than 50 years of age (3.1% per year), attributable to improvements in early detection and treatment [2].

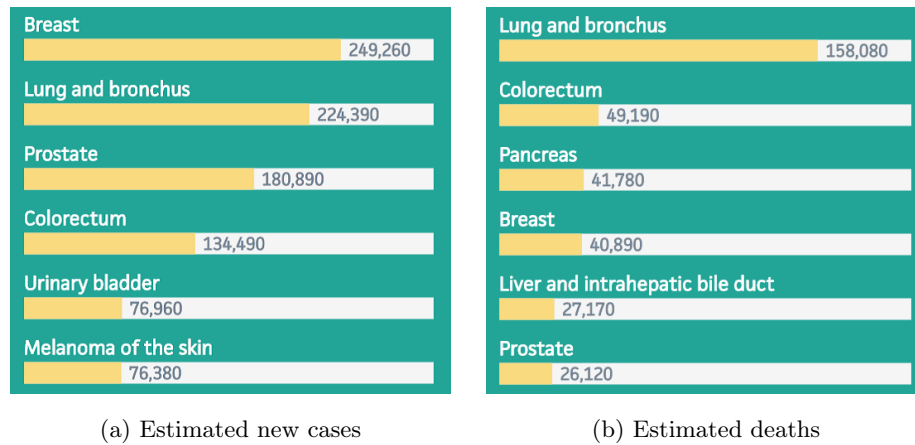


Figure 1: Estimated new cancer cases and deaths in 2016 by cancer type, both sexes combined. American Cancer Society

After treatment for breast cancer, follow-up care is important to help maintain good health, manage any side effects from treatment, watch for signs that the cancer has come back after treatment and screen for other types of cancer. A follow-up care plan may include regular physical examinations and other medical tests to monitor the recovery during the coming months and years.

Women recovering from breast cancer are encouraged to follow established guidelines for good health, such as reaching and maintaining a healthy weight, exercising, not smoking, eating a balanced diet, and following cancer screening recommendations. For example, it is highly recommended being physically active for at least 150 minutes of moderate or 75 minutes of vigorous activity each week.

Technology can greatly help those women who have had breast cancer and have overcome it. For example, Fitbit, an American company known for products like activity trackers, wireless-enabled wearable technology devices that measure data such as the number of steps walked, heart rate, quality of sleep, steps climbed, and other personal metrics, have announced a collaboration by partnering with the Dana-Farber Cancer Institute for breast cancer research. The study will attempt to figure out if exercise helps prevent breast cancer from recurring.

Smartwatches are one of the most popular and powerful devices in wearable technology. These devices are effectively wearable computers, running applications using a mobile operating system, as Android Wear. The smartwatch industry is fast growing, from USD 1.3 billion in 2014 to expected 117 billion in 2020. New generations of watches feature continuous measurement of physiological parameters, such as heart rate. Smartwatches connect with accompanying smartphones and receive messages and notifications that are potentially very useful for ubiquitous monitoring applications.

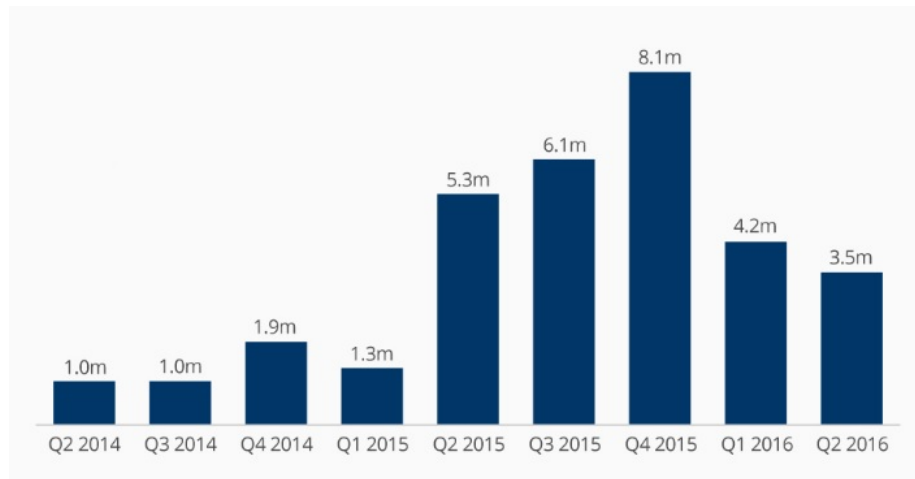


Figure 2: Estimated worldwide smartwatch shipments. Strategy Analytics

Smartwatches are equipped with smart sensors, facilitating one of the main trends in big data science-the quantified self (QS). QS community is engaged in self tracking or group tracking of physiological, behavioral, or environmental information. The community shares insights, approaches, and algorithms.

New sensors and systems enable seamless collection of records and integration in databases that can facilitate data mining and new insights. Several companies created health kit toolsets, such as Google Fit.

Introduction of continuous physiological monitoring on smartwatches may revolutionize the field of mHealth and longitudinal monitoring. Smartwatches provide an unprecedented opportunity for collection of large data sets that can be used for individual monitoring as guidance and monitoring of specific patient populations.

The smartwatch is worn more than any other wearable sensor or device. Due to the constant contact with the skin it may collect biological, environmental, and behavioral information about user's activity, and even identify the user. The most important features of the smartwatch based physiological monitoring systems include: wearability, sampling frequency, personalization of monitoring, seamless data integration, connectivity, multisensory integration and context awareness [3].

The use of wearable sensors has made it possible to have the necessary treatment at home for patients after sudden attacks of diseases such as heart attacks, sleep apnea, Parkinson disease, cancer and so on. Patients after an operation usually go through the recovery/rehabilitation process where they follow a strict routine. All the physiological signals as well as physical activities of the patient are possible to be monitored with the help of wearable sensors. During the rehabilitation stage the wearable sensors may provide audio feedback, virtual reality images and other rehabilitative services. The system can be tuned to the requirement of individual patient. The whole activity can be monitored remotely by doctors, nurses or caregivers [4].

Breast cancer rehabilitation can be linked to wearable technology, specifically smartwatches and smartphones, in order to facilitate the recovery from the disease. Automatic and remote monitoring will enable the patient to follow the instructions of the medical expert and perform and send useful tests from the home environment while the only devices required are the mentioned handheld devices. The medical expert will have access to the tests of patients automatically and thus will be able to follow closely the evolution of the recovery from breast cancer.

1.2. Objectives

The objective of this thesis is the design and development of a system to improve the health of patients who have suffered of breast cancer and find themselves in the rehabilitation process. The *mobile health system for evaluation of breast cancer patients during treatment and recovery phases* comprises 3 different technologies and applications: an application for a smartwatch running on the Android Wear operating system, an application for a smartphone running on the Android operating system and a web server application that can be accessed from any device with an Internet connection. The patient, through the handheld applications, will be able to monitor parameters of importance and send them to the server so the medical expert can visualize them.

More precisely, the objectives of the system can be schematized as it follows:

- Define mechanisms to obtain heart rate, energy expenditure and arm mobility data from the patient.
- Communication between the devices and technologies involved so they become a single and unique system.
- Define storage procedures to store patients personal information and performed health tests on a remote server.
- Visualization tools to show the received data from the handheld devices to the medical expert.
- Present intuitive user interfaces since the potential users may not be used to use the involved devices.
- Build a compact and flexible system that can be extended in the future.

1.3. Thesis and Structure

This thesis is divided into the following chapters:

- *Chapter 1. Introduction.* It is the current chapter which is composed by three sections. The first section titled Motivation, Context and Problem Statement describes the importance of building a tool to help patients who have overcome breast cancer, which is the subject of this thesis. The second section, Objectives, contains an overall summary of the proposal objectives at the beginning of the thesis. The third and last section, Thesis and Structure, presents the structure and composition of the thesis.

- *Chapter 2. State of the Art.* This chapter presents a review of four fields related to the mobile health area: mHealth Overview, mHealth Devices, mHealth Applications and mHealth Frameworks.
- *Chapter 3. System design.* In this chapter, an overview of the requirements of the system is presented first, followed by the top level view of the system and the description of the theoretical methods followed to obtain heart rate, energy expenditure and arm mobility of the patient.
- *Chapter 4. System implementation.* A detailed description of the whole system, which is composed by the smartwatch application, the smartphone application and the web application, can be found in this chapter. The system usage section closes this chapter describing the use of the system.
- *Chapter 5. System test.* The global system and conducted tests to prove the functionality of the system are evaluated in this chapter, along with information about the monitoring time, durability and data consumption of the system.
- *Chapter 6. Conclusions.* This chapter contains the aims achieved in this thesis and a proposal of future work.

2. State of the Art

This section aims to cover four different areas: mHealth overview, devices, applications and frameworks.

2.1. mHealth Overview

The Foundation for the National Institutes of Health defined mobile health or mHealth as the delivery of healthcare services via mobile communications devices [5]. The term is conceived as a conglomeration of hardware and software components, see mobile devices, a software platform that serves basic functionality and applications that provide health services to the user [6].

The idea has been in development for years but lacked the technology required to make it possible. Thanks to the exhuberant growing of ICT, soon it became a feasible via to approach an area of extreme importance as it is to improve health. In 1965, Y. Hatano, a Japanese pedometer manufacturer, in order to improve well-being, determined that people should walk 10.000 steps per day [7]. The philosophy and traditions were wrong placed: instead of focusing on "healthcare", countries and behaviors were centered in "sick care" [8]. This is shifting to a more preventive attitude. For example, health insurance in the United States is drifting to a direction in which companies are tending to reward people who exercise more and take care of their well-being [9].

Nowadays, since fitness and wellness technology (devices and applications) are widely expanding within the mobile market, the industry (companies, developers and users) is ready to achieve several goals that should conform an integrable system to be fitted on the actual backbone. Next, objectives and usefulness of mHealth are listed:

- To serve efficiency to an overbooked system in favor to save time for both patients and health care organizations [10]. For example, medical records could be accessed by mHealth systems [5].
- Inform, motivate and enable users to manage their own health information [11]. By the realization of this, the early diagnosis of potential diseases could be under control.
- Provide education and awareness in order to achieve a more preventive posture while attracting and keeping users to create impact on their adop-

tion behavior. Theoretically, the only fact of enhance monitoring will plant a seed of motivation in every user.

With nearly 7 billions of phone subscribers in the world it is believed that between the 30% and 50% of smartphones owners use their devices to get health information and about one fifth of them have installed at least one health app [12] [13]. The potential of the field is out of doubt. The affordability of smartphones, the widespread use and acceptance from the average consumer offer numerous opportunities and facilitates the approach of human health services. High data connectivity, low-cost and energy-efficient wireless electronics, implementation research initiatives, social awareness (from developers to average people), backing from tech giants and massive possibilities in plausible applications demonstrate the impact of mHealth competence. The practice of medicine and public health supported by mobile devices promises that the user will be capable of easily and realibly self-diagnose symptoms and measure data by themselves without the need of a psysician to evaluate the results or to send the information if that is the desired purpose. From general healthcare to specific diseases, tracking heart, muscle and brain electrical activity, blood oxygen saturation and pressure, breathing rate, glucose level and body temperature, among others, is no longer a task to manage only by specialists with specific instruments [14]. The point is to capture biofeedback and be able to track physiological activity that are indispensable for health supervision.

In contemplation of this, multiple considerations and approximation strategies must be established to achieve success:

- The system needs to be useful, easy to use (friendly user interface), interactive, personalized and to provide connectivity. Realibility and accuracy on data acquisition, processing, storage (local or in the cloud), transmission and sharing are imperative [15] [16].
- Flexibility, sustainable business models and private treatment (security) of data are fundamental ingredients.
- Mobility and wearability are crucial concepts in terms of location of the device on the body, form, design and weight. Since devices are body adornments with both expressive and referential characteristics [17] that must visibly wear, they relate with the social factor, hence impact and behavior on people are significative aspects.

- Spread a philosophy that wearables are not about replacing reality but about augmenting it [18]. Systems should not interfere too much or be invasive, instead they need to provide the exact services that are required.
- Health researchers, technology developers, software designers and clinicians must collaborate together in pursuance of getting more committed users [19]. Considering novel designers without sufficient budget to bring to reality their ideas, crowdfunding is a useful tool to take advantage of as a new way of raising capital.
- It may be too early for a comprehensive adoption of these systems and their implantation on everyday life [20], although people were not used to carry computers in their pockets and they got used to it, as it eases the way they live [21].

Two of the areas that are related to mHealth are commonly known as the Quantified Self (QS) and the Internet of Things (IoT). The philosophy behind the QS movement is that using wearable technology which provides quantifiable data it is possible to significantly improve the understanding of well-being and enhance it in many different ways [22]. This is mHealth in its maximum exponent, as it has been described as a system that helps the users reach their own goals. In relation to the IoT, it is perceived as the connection between everyday objects (from lamps to cars) and Internet in order to have them under supervision and be able to control and manage them as pleased. This extends directly to mHealth as we can include wearable tech in this area, more directly fitness and activity trackers. The steps involved in the creation of an IoT system can be visualized as data generation, treatment and the correspondent action-taking of our choice based on the results of that generation of information [20]. Having this applied to the medical field results in massive opportunities to explore. These two concepts are mandatory to have in mind as they are strictly connected with the essence of mHealth.

2.2. mHealth Devices

The hardware part of mHealth is defined by the device that is going to be worn or used. At present, many examples of wearable technology devices can be found in the market, as it expands in multiple directions. The designs have varied from bracelets or wristbands (the most preferred ones) to smartwatches, contemplating clip-on styles, clothing (textiles), necklaces, rings, glasses, pieces of jewelry, different sensors attached to sport instruments or patches fixed in

precise parts of the human body (e.g. thigh or waist bands). All of them are essentially multi-sensors mobile platforms that measure a physical action and converts it into a countable signal that can be combined with other signals in order to be evaluated by an observer [23].

As for creating a wearable gear several aspects should be considered, being wearability the most crucial one and the most challenging problem: low-weight, small size, noninvasive, stable and comfortable placement, consistent and security are essential elements. Also, it should incorporate interactivity, sensing capabilities, effectiveness, real-time processing, wireless connectivity, constant access to information and appropriate attention on power management. Some of these components represent the assets of wearable technology since it should be an entity that facilitates the work done. On the other hand, demands include that it will be obligatory to avoid systems that could be obtrusive, frustrating, heavy or which give a feeling of discomfort. Beyond these technical details it has been found a more profound controversy regarding the awareness of the area and the usage of it. Studies show that the growth in devices sales is going to be hugely flourishing in the next years, shipping to the market more than 55 million of gadgets [9]. The popularity of these kind of devices is constantly growing but there's a significative gap until the average user decides to purchase and start exploring the possibilities they offer. In terms of marketing, Kyoungwhan Oh's thesis *"The effects of brand, design and price on intent to purchase and activity tracker"* [24] demonstrates that while brand and design had a positive impact on perceived quality, price (among 10\$ and 250\$) had no significant effect on willingness to buy.

The device's design depend on the manufacturer and the data they aspire to capture. A band affixed to the chest will be able to receive data more faithfully from the heart than a wristband, therefore the location in which the gadget is placed determines the purpose and vice versa. Then, it exists a strict relationship between the device's possibilities and the activity to be performed, not only because of the placement and the form, but also for the kind of signals it expects to receive. Concerning motions, when these are dominated by translations, acceleration features are used, and when led by rotations, the gyroscope provides information that is invariant to body part displacement [25].

The most common tasks carried out by activity trackers are measuring steps,

calories and tracking sleep. While steps are fairly accurate measured (despite that smaller or quicker steps are difficult recorded by the device since it indicates less arm movement, which is the way it receives the information required), calories are less faithfully computed. The truthfulness of the devices depends again on the exercise modality [7]. A lack of sleep is linked to many risks regarding health, thus it marks a subject of high importance. Sleep trackers seek to combat those exposures in order to provide the user with valuable data about their bedtime habits. Sleeping patterns are complex and since this is an area in current improvement, accuracy of sleep trackers still do not match the quality of information that can be received in a sleep lab [26]. All this material collected by motion and visual sensors, different types of signals, meaning blood volume pulse, respiratory rate or a wide range of moves (from simple exercises to complete physical activities), requires to be processed adequately in a variety of stages [20]. Here, the "big data" challenge takes leadership and needs to be solved, because of the tremendous processing power demand, in order to support automatic operations [8].

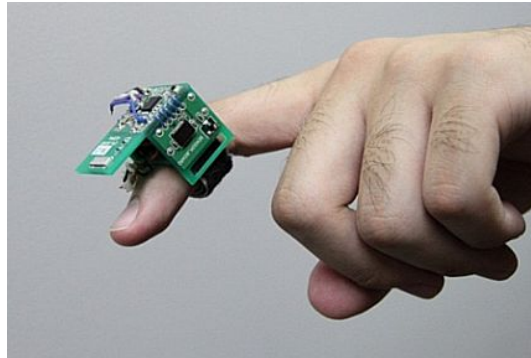


Figure 3: The Magic Ring device

Giving some unique representative examples of wearable technology, Lei Jing and his team developed the *Magic Ring*, a finger ring shape input device using inertial sensor to detect the subtle finger gestures, routine daily activities and various hand manipulations [27]. Tomoya Tanaka from the University of Hyogo built the *button system*, which is attached over the chest for monitoring electrocardiogram (ECG), heart rate, 3 axis acceleration and temperature [28]. Speaking of wearable technology products that are being commercialized, popular enterprises like Nike [29], Fitbit [30], Jawbone [31], Samsung [32] or Misfit Wearables [33] have created bracelet devices accompanied with a proper smart-



Figure 4: Jawbone UP and accompanying app

phone app that generally tracks activity (steps taken, calories burned, distance travelled, hours slept) and conforms a personal digital trainer. Other instances are specifically sport or gym oriented and some of them deliver intuitive notifications (calls, emails, message) by LED lights or installed screens. Apart from



Figure 5: Jolt clip

wristband products, Athos [34] conceived a textile kit that reads muscle effort and gives insight to exercise correctly. Fin [35] designed a wearable ring worn on the thumb, which is defined as "a real life buddy for every individual to do their digital interactions". Jolt [36], a company from Washington, have developed a clip sensor that tracks athletes' head impacts in real-time and enables them to act decisively in the critical moments following a dangerous impact. These examples show the power and potential of wearable technology and this is only the beginning since there is a multitude of different visions trying to help users in various other ways.

Apple plans for mHealth are made by the combination of the HealthKit platform, the OS 8 Health App and mainly by the Apple Watch, a smartwatch

equipped with an accelerometer for body movement measures, a heart rate sensor and pedometer linked to GPS, an operating system with the named Health App that uses a user-friendly dashboard to display health data and numerous connections with other devices and apps [37]. Another sight of a smartwatch is Pebble [38], which received the majority of its initial funding via the crowdfunding platform Kickstarter [39]. Compatible with Android and iOS devices, it features a black and white e-paper display, a vibrating motor, a magnetometer, an accelerometer, among others, enabling its use as an activity tracker.

2.3. mHealth Applications

As mentioned, mHealth is the summation of the hardware component covered in the previous section and the software segment, meaning the programs and mobile applications with medical purposes. These health ends can be categorized in diverse groups depending on the target which is meant to be achieved. A set of these areas is presented next covering a wide range of mHealth's applications [5] [10] [12]:

- Personal (healthy living, fitness training, disease treatment, tracking tools).
- Social (gaming, sports).
- Home health monitoring (medication adherence, sedentary behavior, disabilities).
- Public and professional education (teaching, behavior change communication).
- Electronic medical records and communication (access to health information).
- Professional information (medical databases, data collections and reports).
- Professional consultation (decision support, health publications).

Highlighting some illustrative examples of apps, *Visible Body* gives medical students, doctors and patients an up-close look at all the systems of the human body, *VaxNation* is an online vaccination tracker, *Calorie Tracker* tracks a user's diet, weight change and workout frequency, and *DrawMD* is a free iPad app that lets physicians show their patients exactly what a surgical or other procedure will entail [10]. At the University of Alabama in Huntsville, a computing infrastructure for mobile health was developed by Mladen Milosevic and

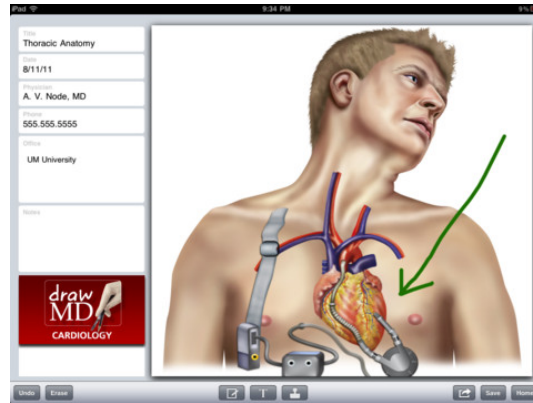


Figure 6: DrawMD App

their research group, including two applications, one which quantifies and automates a standard test used to assess mobility of individuals and *mWheelness* that monitors physical activity of people who rely on wheelchairs for displacement [14]. Giving strictly medicinal uses, monitorization of cardiac arrhythmia or congestion in heart failure and managing stress, obesity or smoking cessation are model enforcements. Going further, investigators and companies have struggled with the brain to obtain a greater understanding of it. Mental performance optimization techniques and emotion reading programs are being examined because of the utility of this area [20]. Human activity recognition (often expressed as HAR) is a distinct prominent iteration of mobile health brought to reality in terms of software. Classification of body postures and movements with wearable accelerometers represent one of the most relevant topics of the field [40].

Apps should offer an ample collection of features in order to being exploited and conform effective products that help people reach their goals thanks to portability, immediacy and comfort. Apps' success relay on content quality, usability, device connectivity standards, attractive interfaces, social sharing, security, data analysis tools, display of convenient 2D/3D graphics, charts and animations, taking advantage of unique device's characteristics (context awareness) and web support. An innovative strategy, commonly labeled as gamification, is being increasingly applied. Through simple game mechanics and the use of video game elements (avatars, feedback, trophies, rankings...) in non-gaming systems, the user experience of applications can be enhanced, since the process of information recollection becomes more enjoyable and fun for users [41]. Ease of use is crucial since people of all ages and different levels of dedication and

experience regarding computer devices will make use of the applications. To minimize costs in relation with development, support and maintenance will enforce the dedication of developers in pursuance of virtue. Widespread adoption and usage of mobile health apps continue to be a challenge despite the large amount of downloads occurring nowadays, predicted to grow from 154 millions in 2010 to 908 millions by 2016 [42].

2.4. mHealth Frameworks

This last chapter covers mHealth frameworks and how these help to carry the necessary tasks and facilitate developers in achieving their objectives. An application framework consists of a software framework used by software developers to implement the standard structure of an application [43]. Therefore, the goal of a mobile health framework is to provide a set of core classes to accomplish basic functionalities needed in mHealth apps [6]. Producing applications for wearable devices can be a difficult work since depends on the exact characteristics of the devices, then to soothe the efforts of computer science designers suppose a essential interest. Commercialized systems are presented with related software but it is usually very specific and closed source. An ideal solution for this issue would be to be able to develop software apps that can run along multiple gadgets without the need of build specific apps for each one, meaning a common one relaying on a practical framework. Constructed and used as an integrable library, frameworks are generally built to cover some basic purposes, like data visualization and processing, communication, storage or some low-level contact with the medical periferic's core in order to establish a connection in which the development of the application can be conveniently be taken care of. Advantages of the development of a framework can be listed as componetization (modularity), reusability of code and design, extensibility, simplicity and easier maintenance of the platform. The dependency of a unique device type or communication protocol does not exist since every particular feature is filtered through the framework and translated to a common language that will act as the skeletal support to build different applications. Although the great potential of frameworks development for the biomedical field, except some of them, there are not abundant precedent examples of this kind of software frameworks already developed.

Samsung is opening the ecosystem with the Samsung Architecture Multimodal Interaction system, S.A.M.I. [44], that will enable wearable devices to

upload information to the cloud, so the data can be worked and visualized in different contexts. The Apple HealthKit [45] it has been built to allow mHealth apps to share their data (stored in a centralized and secure location) with the Apple Health app, so the whole system gets globalized. Open mHealth's [46] approach makes digital health data useful and easily collectionable, in such a way that their API lets developers work comfortably with the information they need. mHealthDroid [47] is a novel framework for agile development of mobile health applications. Created in the University of Granada, serves as a base for this project, since its structure based in different managers lets a easy way of build Android applications in relation with wearable devices.

Implementations of this kind will tremendously help the grow of the mHealth field. It eases the communication between all the elements that are part of this conglomerate, people (developers and users), devices and applications. Efforts are being made in favor of a consistent and normalized structure that permits the evolution and expansion of this crucial area of technology.

3. System Design

3.1. Top Level View

Development of an automatic monitoring system for patients who have suffered from breast cancer, have overcome and are undergoing rehabilitation. The system obtains useful data (heart rate, energy expenditure and mobility of the affected arm) through which an expert can reach conclusions based on their knowledge of the rehabilitation that the patient is carrying out.

These data of interest are obtained by heart rate, accelerometer and gyroscope sensors embedded into a smartwatch running on the Android Wear operating system. This information is transmitted to a smartphone running on the Android operating system where data is processed into useful information. Finally, these data are sent to a remote server where they are stored in a database (which cohabits with personal patient data). Subsequently, the data can be displayed in the form of charts in an accessible web interface to the expert from any mobile device or computer with Internet connection.

A diagram of the system comprising 3 different applications is shown next:

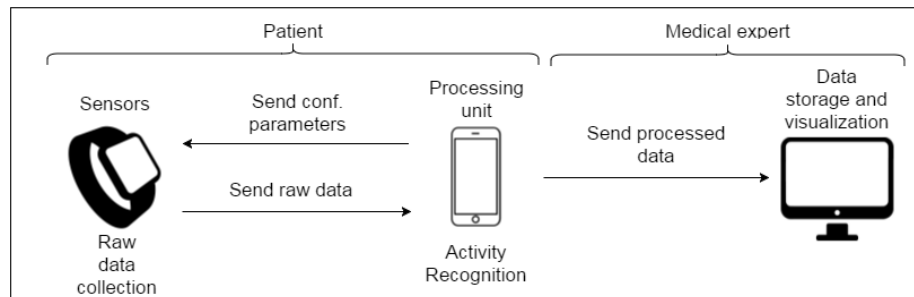


Figure 7: Top level view of the system

3.2. Requirements

Each actor of the system has its own set of requirements. Patients need to monitor specific parameters data to provide the doctor with treated information that can be easily interpreted in order to return intellectual feedback. Due to this reason, information regarding monitoring is displayed to the patient and it is the medical expert the one who will be able to visualize the performed tests.

A list of functional requirements that specifies functions and particular results of the system is presented next:

- Set configuration parameters, monitoring and frequency times, on the smartphone application for heart rate, energy expenditure and arm mobility readings.
- Send configuration parameters of heart rate, energy expenditure and arm mobility readings from the smartphone application to the smartwatch application.
- Receive configuration parameters of heart rate, energy expenditure and arm mobility readings on the smartwatch application from the smartphone application.
- Perform readings of heart rate, energy expenditure and arm mobility through the sensors of the smartwatch.
- Send sensor data of heart rate, energy expenditure and arm mobility from the smartwatch application to the smartphone application.
- Receive sensor data of heart rate, energy expenditure and arm mobility on the smartphone application from the smartwatch application.
- Process received sensor data according to specific methods on the smartphone application to obtain tests objects.
- Send processed data, tests objects, from the smartphone application to the web server application.
- Store on databases on the web server application the data received, tests objects, from the smartphone application.
- Log in to the web server application.
- View, insert and remove patients from the patients database on the web server application.
- View and delete tests from the tests database on the web server application.
- Visualize tests related to a specific patient, test type and date in line charts and bar charts.

3.3. Methods

3.3.1. Heart Rate

Breast cancer therapy alters autonomic function and contributes to cardiovascular disorders. According to the American College of Cardiology and American Heart Association, oncology patients who have received chemotherapy represent a high-risk for heart failure.

Heart rate variability (HRV) is an important noninvasive index of vagal-nerve response and a potential stress marker which can be a useful test for autonomic imbalance. It represents the time differences between beat-to-beat intervals, synonymous with RR variability (RR interval in the electrocardiogram). Analysis of the time differences between successive heartbeats can be accomplished with reference to time (time-domain analysis) or frequency (frequency-domain analysis). The parasympathetic and sympathetic nervous systems, which are primarily responsible for changes in the inter beat interval, modify the heart rate and can be quantified by using HRV time- and frequency-domain parameters [2].

The heart rate variability analysis could be used as a complementary non-invasive tool for the early diagnosis and better prognosis of autonomic dysfunction, and survival in BC patients. There are many potential clinical applications of heart rate variability analysis in BC patients, and the use of such approaches could lead to lower impairment of autonomic function in this individuals. [1]

Heart rate variability analysis may be used to evaluate autonomic changes in breast cancer patients and shines a light on understanding some breast cancer outcomes. Prognosis of autonomic dysfunction in breast cancer patients may be assessed by HRV analysis.

The initial object of the system was to calculate heart rate variability due to its crucial importance but it was not possible to attempt because of the device used, the LG G Watch R smartwatch, equipped with a PPG heart rate sensor but unable to return the raw signal which is needed to calculate HRV. This device running on the Android Wear operating system only returns the heart rate values expressed in beats per second.

Instead of this, the system retrieves heart rate values according to a fre-

quency time and calculates the average of this set of data while at the same time computes the standard deviation, to quantify the amount of variation or dispersion of this set of data values, to approach the initial objective.

3.3.2. Energy Expenditure

Accelerometers, as the one included on smartwatches, satisfy many of the requirements for physical activity assessment, such as the possibility of measuring it in free-living conditions with minimal discomfort for the subject and in a representative time frame for the average activity level.

The metabolic equivalent of task (MET) is currently the most used indicator for measuring the energy expenditure (EE) of a physical activity (PA) and has become an important measure for determining and supervising a person's state of health.

A physical activity can be measured with accelerometers through calculation of counts. Each count is a value that indicates the strength of a movement and can be used in conjunction with other parameters (height, weight, age, gender) to determine the related METs to a PA and thus the EE. The number of METs can be calculated through the number of counts obtained from information from the accelerometers and the aforementioned physiological parameters.

The system has presented the algorithm expressed in [48], which can be resumed in the next steps:

- Get raw accelerometer values (x, y, z) for a certain period of time with a sampling rate of 15 Hz.
- Calculate the linear acceleration of those values (with low-pass and high-pass filters).
- Normalize linear acceleration: $\sqrt{(linear_x)^2 + (linear_y)^2 + (linear_z)^2}$
- Apply an integration process to calculate the area under the curve. The sum of the areas returns the total number of counts:

$$\int_a^b f(x) dx = (b - a) \frac{f(a) + f(b)}{2}$$
- Once obtained the counts, calculate METs through the next mathematical formula (for all age-groups combined) with the patient physiological information (height, weight, age and gender):

$$\text{EE (METs)} = 2.7406 + 0.00056 \times \text{VM activity counts (counts} \times \text{min}^{-1}) - 0.008542 \times \text{age (years)} - 0.01380 \times \text{body mass (kg)}$$

The developed system calculates activity counts of a linear accelerometer reading and transform them into METs. Both data are displayed on charts with the referential time values of the readings.

3.3.3. Arm Mobility

Surgery is usually the first line of attack against breast cancer. After breast cancer surgery, some women experience numbness, swelling, weakness, or tingling in the arm and shoulder area on the same side of the body on which surgery was done. Several nerves run through the chest and underarm area and may be affected by breast cancer surgery, so muscles can get weaker compared with how they were before surgery. Exercises that improve shoulder and arm mobility usually can be started a few days after surgery while exercises to improve arm strength come later.

Wearable sensor technology has enabled unobtrusive monitoring of arm movements of different diseases survivors, like stroke survivors, in the home environment, with accelerometry representing the most established approach. Activity counts derived from the acceleration signals provide quantitative information about arm activity, such as total duration and intensity of movements. In conjunction with a gyroscope sensor it is possible to measure angular displacement during highly dynamic activities.

Following the method proposed by [49], the system qualitatively assess functional arm use in the home environment, relying on only a single wrist-worn sensor module, like a smartwatch equipped with both accelerometer and gyroscope sensors.

This is achieved through the following schema:

- Get raw accelerometer and gyroscope values (x, y, z) for a certain period of time with a sampling rate of 50 Hz.
- Calculate the forearm orientation relative to the earth referential frame using the gradient descent orientation filter proposed by Madgwick [50] which fuses sensor measurements of gravity and angular rate into an optimal orientation estimate. The filter outputs orientation in a quaternion

representation, $q = [q_0, q_1, q_2, q_3]$, which has to be transformed into a 3 x 3 direction cosine matrix R . To calculate the forearm elevation, the forearm vector expressed in the earth fixed referential is needed, $a_e = R^T[1, 0, 0]^T$. The elevation Θ of the forearm vector $a_e = [a_{ex}, a_{ey}, a_{ez}]$ can be computed as:

$$\Theta = \arctan\left(\frac{a_{ez}}{\sqrt{a_{ex}^2 + a_{ey}^2}}\right) \text{ and then represented in a polar representation.}$$

The system calculates the elevation Θ of the forearm activity measuring between -90 and 90 degrees on the vertical axis. This data is displayed on a chart with the referential time values of the readings.

4. System Implementation

4.1. Overview

The standalone system comprises different devices and technologies. It is design involves 3 different applications that are connected and synchronized to offer the desired functionality. Through the use of these applications, the patient will be able to monitor automatically valuable information about the rehabilitation that is in current progress and the medical expert will have an adaptive environment to check the data generated by the patient.

In the following subsections, a detailed vision of every technology and application is offered in order to provide insight and proper knowledge about the behaviour and performance of the whole system.

1. *Smartwatch Application.*

A wearable application has been developed to take advantage of the possibilities of this kind of devices. First, a general vision of the operating system is offered, highlighting the capabilities that make smartwatches powerful devices. Then, particular details of the application's implementation are presented.

2. *Smartphone Application.*

The Android operating system is quite known by now since an immense number of applications has been developed in the past years. Also, some interesting features have been implemented in the system that are worth mentioning, such as the connection between the smartphone and the server. This and a complete view of the smartphone application will be presented in this section.

3. *Web Server Application.*

The Web Server Application that is available for the medical expert constitutes a complete tool that is accesible from any device with Internet connection. The user will manage the databases of patients and tests and will visualize the statistics that the related patients have generated thanks to the handheld applications. A mix of different programming languages have been used in order to create a useful and compelling application which will be detailed in this category.

4.2. Smartwatch Application

4.2.1. System

Android Wear is a version of Google's Android operating system designed for smartwatches and other wearables. By pairing with mobile phones running Android version 4.3 or newer (by the accompanying Android Wear app that should be downloaded and configured on the phone), Android Wear integrates mobile notifications and other features. Also, it adds the ability to download applications from the Google Play Store.

Android Wear supports both Bluetooth and Wi-Fi connectivity, as well as a range of features and applications. Watch face styles include round, square and rectangular. Released devices include the LG G Watch, the Motorola Moto 360, or the Samsung Gear Live. Other hardware manufacturing partners include ASUS, Broadcom, Fossil, HTC, Intel, Huawei and many others.

The most interesting features that these devices offer are the following:

- Users can orient themselves by voice from the phone, choose transport mode, including bike, and start a journey. While traveling, the watch shows directions, and vibrates to indicate turns by feel.
- Via Google Fit and similar applications, Android Wear supports ride, run tracking, step-counting, calorie expenditure and general body activity.
- Users can use their Android Wear Watch to control their phone, for example, music can be requested to be played.
- The notification system between the watch and the phone is compatible with a large number of existing applications, showing to the user notifications the phone generates by displaying them on the screen of the smartwatch, so the user can check them directly.
- If the phone's camera app is activated, the screen is relayed to the watch, and the user can control the shutter, and view photos on the watch.

A differentiating factor over other devices is the large number of sensors included in such devices. For example, a heart rate monitor is a usually included working sensor that can only be found in specific devices prepared to calculate this information. The intrinsic design of smartwatches makes the use of such sensor in these devices ideal.

4.2.2. Implementation

The main objectives of the smartwatch application can be summarized in 4 ordered points:

- Receive, from the mobile phone application, the configuration parameters, monitoring time and frequency time, of the desired reading or readings to be performed.
- Monitor the desired physiological information (heart rate, energy expenditure or arm mobility) of the patient through the integrated sensors of the device.
- Send this data to the mobile phone application.
- Show a friendly and intuitive user interface (UI), since the potential user which will use this application may not have experience with the use of a smartwatch. The UI is divided in 3 different fragments dedicated to heart rate, energy expenditure and arm mobility respectively.

Reception of configuration parameters (from the mobile phone application)

To start monitorization on the smartwatch application, two essential configuration parameters are required to initiate the desired readings. These parameters are sent from the mobile phone application to the smartwatch application and once they are properly received on the smartwatch application, the patient will be able to press the button displayed on the screen to start calculating the desired data.

More precisely, the two configuration parameters that are received are:

- *Monitoring time*: an integer value expressed in milliseconds that indicates the amount of time one sensor specifically will be generating data. For example, if the mobile phone application sends a monitoring time of 600000 milliseconds for the heart rate calculation, this value will be received on the smartwatch and when the patient presses the heart rate button to start the monitorization, this functionality will be running during 10 minutes.
- *Frequency time*: an integer value expressed in milliseconds that represents how often a value is sampled and stored from the generated data. For

example, for a frequency time of 30 seconds for the heart rate, this can be understood as while the sensor is generating data during those 30 seconds, at the end of this amount of time, the system will calculate the average of those values and will store it. In conclusion, the system does not register every single value (whatever this is), but only every frequency time.

To receive these parameters the smartwatch application make use of a *WearableListenerService* with a related path, depending if the parameters that are being sent are for heart rate, energy expenditure or arm mobility.

An example of how the smartwatch application receive the configuration parameters through a *WearableListenerService* is shown as it follows, applied to the heart rate calculation (the same could be applied to energy expenditure and arm mobility):

Listing 1: Reception of configuration parameters on smartwatch

```
1  // WearableListenerService
2
3  @Override
4  public void onDataChanged(DataEventBuffer dataEvents) {
5
6      super.onDataChanged(dataEvents);
7      Log.i(TAG, "onDataChanged on Wear is called");
8
9      for (DataEvent dataEvent : dataEvents) {
10
11          Log.e(TAG, "Event type: " + dataEvent.getType() + ", Uri:' " +
12              dataEvent.getDataItem().getUri());
13
14          if (dataEvent.getType() == DataEvent.TYPE_CHANGED &&
15              dataEvent.getDataItem().getUri().getPath().contains("/HRMobileToWear"))
16          {
17              DataMap dataMap =
18                  DataMapItem.fromDataItem(dataEvent.getDataItem()).getDataMap();
19              Log.i(TAG, "dataMap HR received from mobile: " + dataMap);
20
21              Intent i = new Intent();
22              i.setAction(Intent.ACTION_SEND);
23              i.putExtra("dataMapHRFromMobile", dataMap.toBundle());
```

```

20         LocalBroadcastManager.getInstance(this).sendBroadcast(i);
21     }
22 }
23 }
24
25 // HeartRateFragment
26
27 // Register the local broadcast receiver
28 IntentFilter messageFilter = new IntentFilter(Intent.ACTION_SEND);
29 MessageReceiver messageReceiver = new MessageReceiver();
30 LocalBroadcastManager.getInstance(this.getActivity()).registerReceiver(messageReceiver,
    messageFilter);
31
32 public class MessageReceiver extends BroadcastReceiver {
33     @Override
34     public void onReceive(Context context, Intent intent) {
35         Bundle dataMap = intent.getBundleExtra("dataMapHRFromMobile");
36
37         if (dataMap != null) {
38             mMonitoringTime = dataMap.getInt("MonitoringTime");
39             mFrequencyTime = dataMap.getInt("FrequencyTime");
40         }
41     }
42 }

```

If the smartwatch application is started, a progress bar for heart rate, energy expenditure and arm mobility are shown indicating, altogether with an explanatory message, that these parameters are required. This it is shown as it follows:

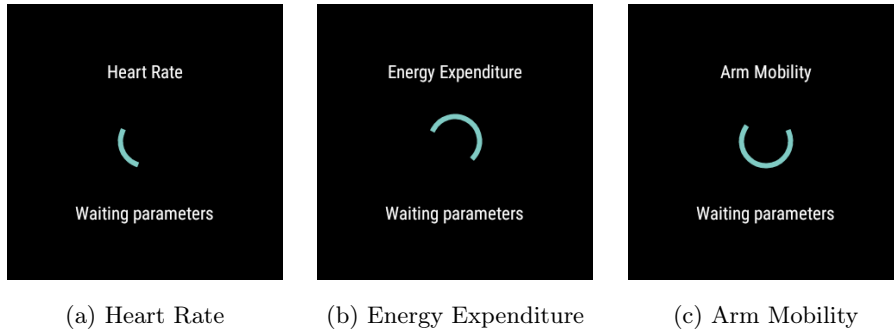


Figure 8: Smartwatch application waiting for configuration parameters with progress bar

And once the parameters are received, the user interface changes to:

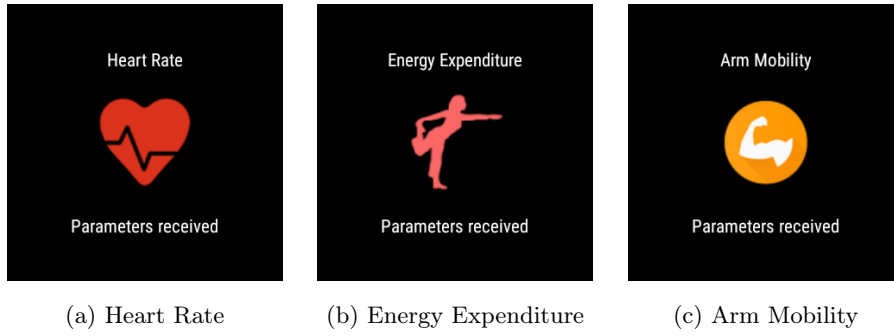


Figure 9: Smartwatch application with configuration parameters received

Monitorization

Heart Rate

The system monitors heart rate through the *SensorManager* of the operating system. To do this, once the sensor is declared we need to use the function *registerListener* on which it is possible to indicate the desired delay between two consecutive events in microseconds.

This is a matter of controversy because the indicated value it is only a hint to the system and it will not deliver results exactly at that rate. This issue only happens with the heart rate sensor and it is caused because of the proper nature of the sensor itself and its low level implementation. This means that this sensor will deliver data at its own rate and the value that is indicated on the *registerListener* function will not make any visual effect. It is possible to

add another value to this instruction, the maximum time in microseconds that events can be delayed before being reported to the application, to force as much as possible the operating system to adjust the rate at the desired one.

An experiment was performed to clarify this topic and to check whether it is possible to control the rate of the heart rate sensor or not.

The experiment was performed with this parameters:

- **Monitorization time:** 5 minutes.
- **Activity performed by the user while collecting data:** still while seated. .
- **Device:** LG G Watch R.
- **Same value for *samplingPeriodUs* and *maxReportLatencyUs*.**
- **Results to be obtained:**
 1. Number of expected samples VS. Number of obtained samples.
 2. Android Plot chart for each experiment where the Y axis will display heart rate values and on the X axis the times in which they were obtained.
- **Sampling periods** (*SensorManager* and custom constants):
 1. `SENSOR_DELAY_FASTEST` (100 Hz) (1 sample every 10000 microseconds = 100 samples every 1 second).
 2. `SENSOR_DELAY_GAME` (50 Hz) (1 sample every 20000 microseconds = 50 samples every 1 second).
 3. `SENSOR_DELAY_UI` (16.7 Hz) (1 sample every 66667 microseconds = 15 samples every 1 second).
 4. `SENSOR_DELAY_NORMAL` (5 Hz) (1 sample every 200000 microseconds = 5 samples every 1 second).
 5. 1 sample every 5000000 microseconds = 1 sample every 5 seconds.
 6. 1 sample every 15000000 microseconds = 1 sample every 15 seconds.
 7. 1 sample every 30000000 microseconds = 1 sample every 30 seconds.

Results of the experiment:

1. **`SENSOR_DELAY_FASTEST`:**

- Number of expected samples (in 5 minutes): 30000 samples.
- Number of obtained samples (in 5 minutes): 151 samples.

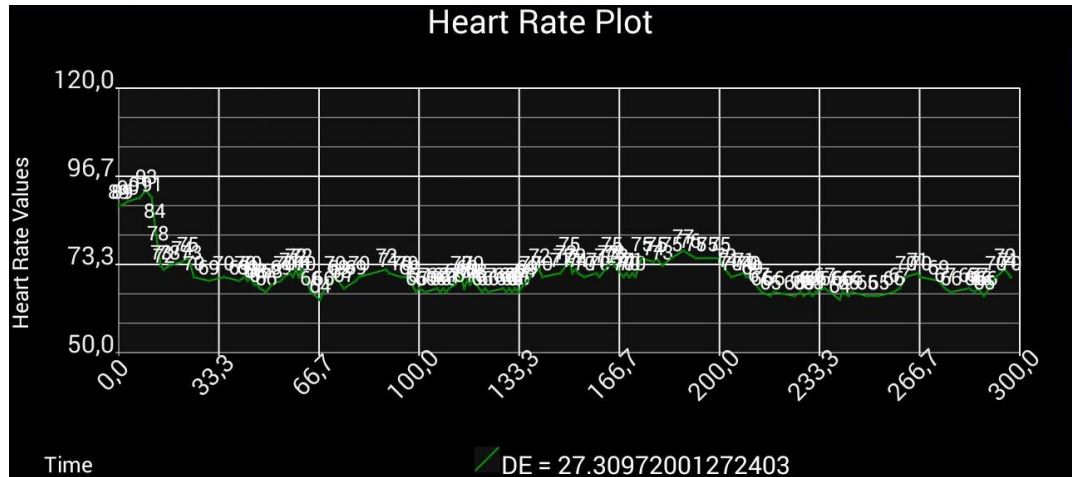


Figure 10: Heart rate frequency experiment with SENSOR DELAY FASTEST

2. SENSOR_DELAY_GAME:

- Number of expected samples (in 5 minutes): 15000 samples.
- Number of obtained samples (in 5 minutes): 103 samples.

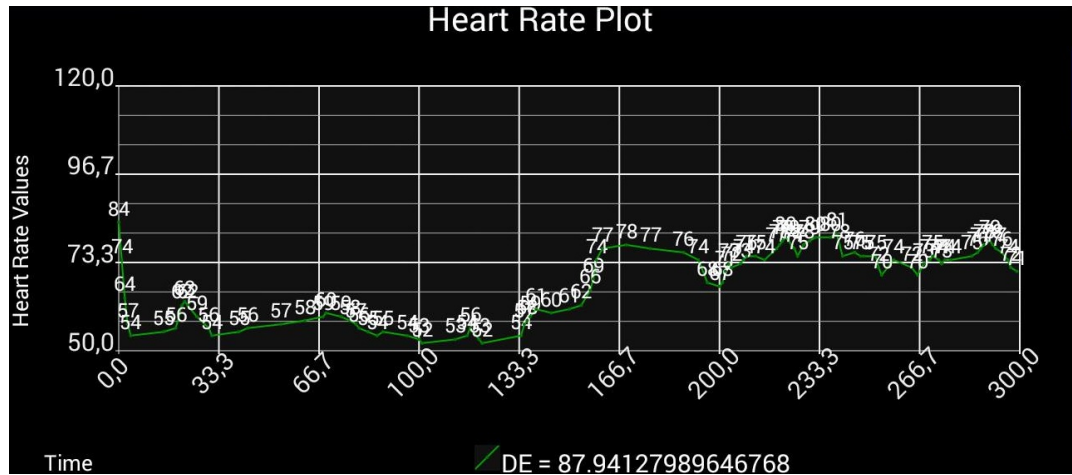


Figure 11: Heart rate frequency experiment with SENSOR_DELAY GAME

3. SENSOR_DELAY_UI:

- Number of expected samples (in 5 minutes): 4500 samples.

- Number of obtained samples (in 5 minutes): 175 samples.

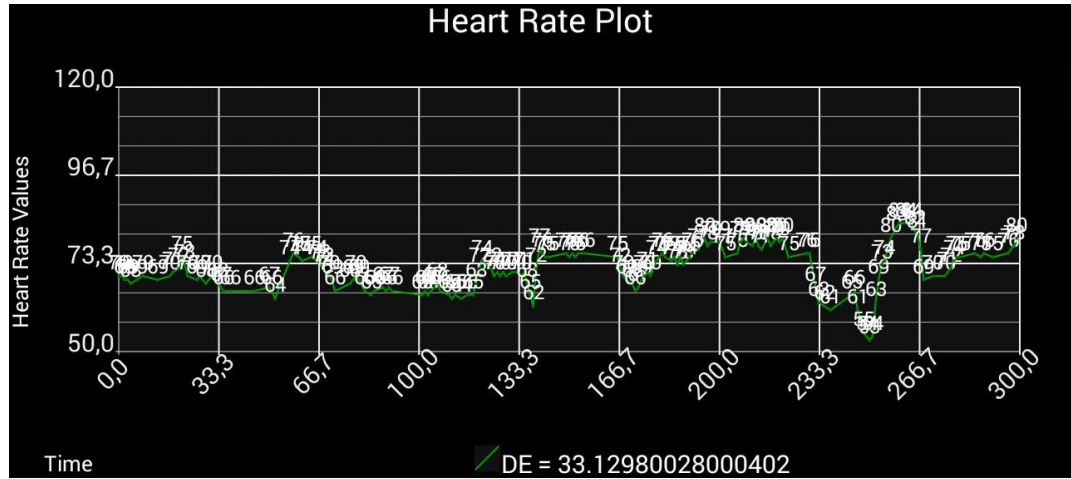


Figure 12: Heart rate frequency experiment with SENSOR DELAY UI

4. SENSOR_DELAY_NORMAL:

- Number of expected samples (in 5 minutes): 1500 samples.
- Number of obtained samples (in 5 minutes): 146 samples.

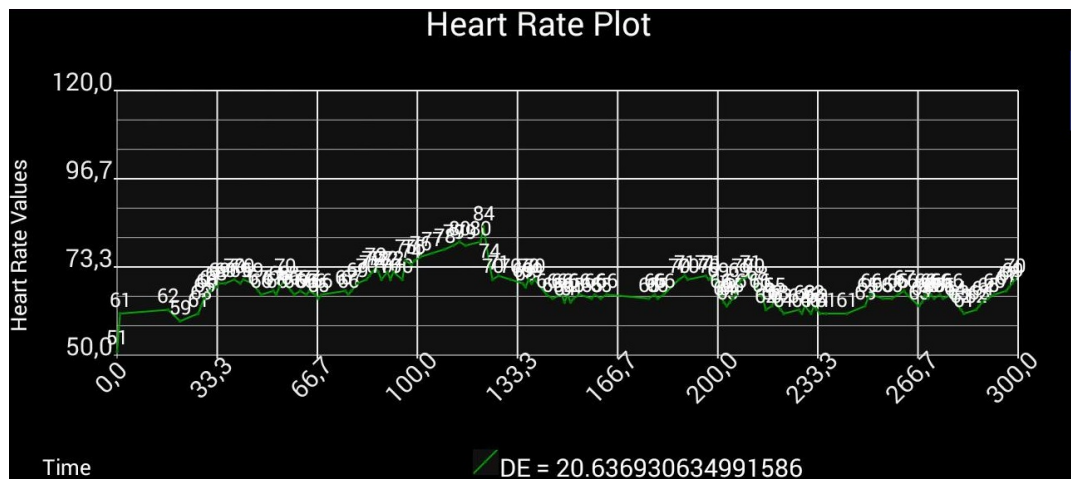


Figure 13: Heart rate frequency experiment with SENSOR DELAY NORMAL

5. 5000000 microseconds:

- Number of expected samples (in 5 minutes): 60 samples.
- Number of obtained samples (in 5 minutes): 181 samples.

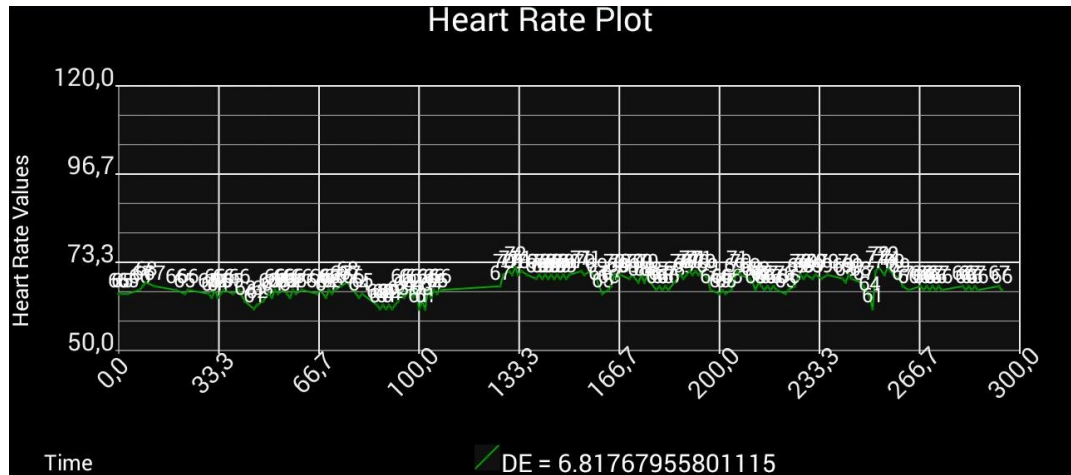


Figure 14: Heart rate frequency experiment with 5000000 microseconds

6. 15000000 **microseconds**:

- Number of expected samples (in 5 minutes): 20 samples.
- Number of obtained samples (in 5 minutes): 169 samples.

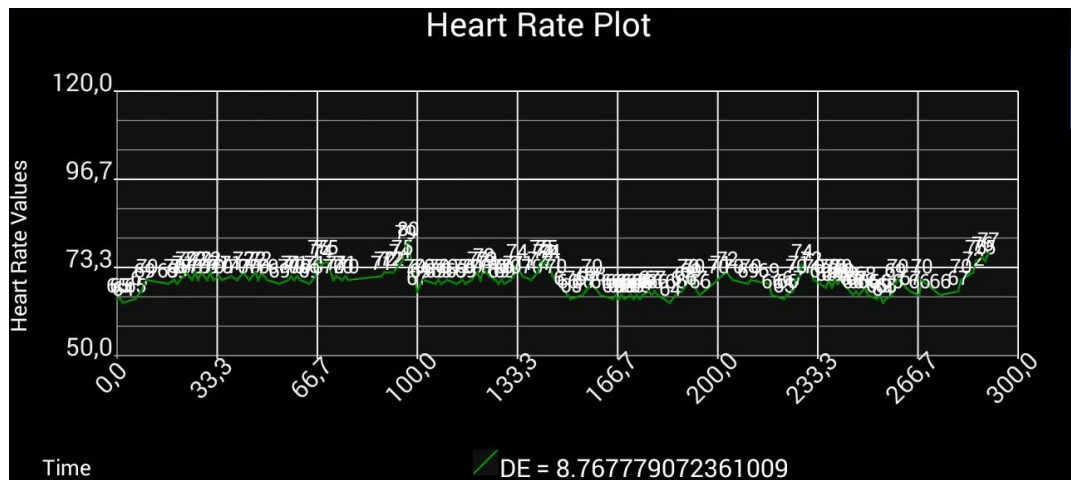


Figure 15: Heart rate frequency experiment with 15000000 microseconds

7. 30000000 **microseconds**:

- Number of expected samples (in 5 minutes): 10 samples.
- Number of obtained samples (in 5 minutes): 147 samples.

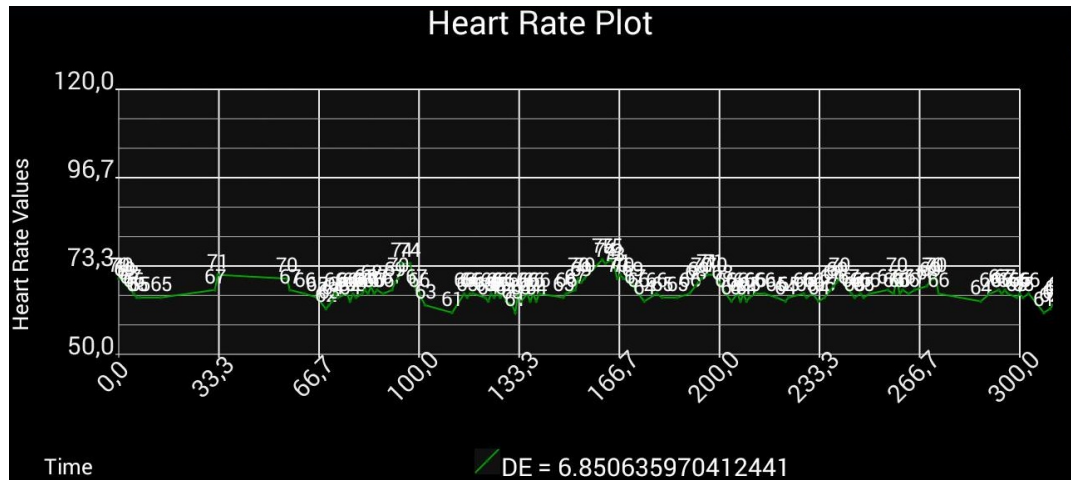


Figure 16: Heart rate frequency experiment with 30000000 microseconds

We can conclude that the heart rate sensor delivers 1 sample every 2 seconds on average (150 samples in 5 minutes). Also, no difference can be seen between the use of one sampling rate or another. Through this experiment, it has been found that a good practice is to specify the same value for these two fields, and make them as large as possible, because the results were more coherent than in other cases.

In the system implementation, a sampling rate of 5 million microseconds (5 seconds) it is set by default, because this would be the ideal rate to receive new heart rate values of the patient.

An example of the *SensorManager* with the heart rate sensor in action is shown below:

Listing 2: Heart Rate SensorManager

```

1 // To get an instance of the SensorManager
2 mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
3 // To get the default sensor for TYPE_HEART_RATE
4 mHRSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_HEART_RATE);
5
6 // Registers a SensorEventListener for the given sensor at the given
   sampling frequency and the given maximum reporting latency
7 mSensorManager.registerListener(this, mHRSensor, 5000000, 5000000);
8

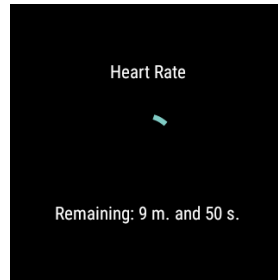
```

```

9  // Raw value of heart rate (bpm, beats per minute)
10 mHRValue = event.values[0];
11
12 // Unregisters a listener for all sensors
13 mSensorManager.unregisterListener(this);

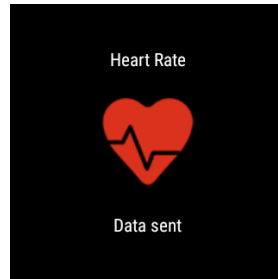
```

The use of this sensor will generate the heart rate of the patient which will be continuously sent to the mobile phone application to be processed.



(a) Heart Rate

Figure 17: Smartwatch application Heart Rate while monitoring



(a) Heart Rate

Figure 18: Smartwatch application Heart Rate when monitoring time has concluded

Energy Expenditure

To calculate the energy expenditure of a patient in a specific time period, we need to take advantage of the accelerometer sensor of the smartwatch. Actually, as the followed algorithm to calculate this data indicates, it is indeed a better idea to use the linear acceleration sensor type because it measures the acceleration force in m/s^2 that is applied to a device on all three physical axis (x, y,

and z), excluding the force of gravity.

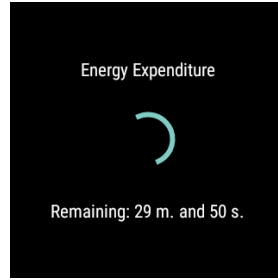
As the methodology to calculate energy expenditure points out, the ideal sampling rate of the accelerometer should be 15 Hz, which is exactly what the already predefined constant of Android, `SENSOR_DELAY_UI`, marks. So the application uses this value on the *registerListener* function for the accelerometer. In this case, it has been proven that the sensor delivers almost exactly 15 values per second, which is what to be obtained.

An example of the *SensorManager* with the linear acceleration sensor running is shown below:

Listing 3: Linear Acceleration SensorManager

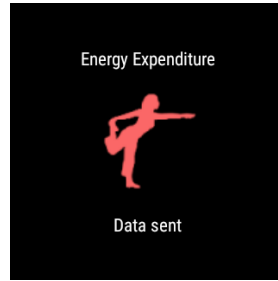
```
1 // To get an instance of the SensorManager
2 mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
3 // To get the default sensor for TYPE_LINEAR_ACCELERATION
4 mLinAccSensor =
    mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
5
6 // Registers a SensorEventListener for the given sensor at the given
    sampling frequency and the given maximum reporting latency
7 mSensorManager.registerListener(this, mLinAccSensor,
    SensorManager.SENSOR_DELAY_UI, SensorManager.SENSOR_DELAY_UI);
8
9 // Acceleration force along the x, y and z axis (excluding gravity).
10 mAccX = event.values[0];
11 mAccY = event.values[1];
12 mAccZ = event.values[2];
13
14 // Unregisters a listener for all sensors
15 mSensorManager.unregisterListener(this);
```

Once this is done, every event value generated by the sensor will be sent synchronously to the mobile phone application to be properly processed and to finally calculate the energy expenditure of the patient.



(a) Energy Expenditure

Figure 19: Smartwatch application Energy Expenditure while monitoring



(a) Energy Expenditure

Figure 20: Smartwatch application Energy Expenditure when monitoring time has concluded

Arm Mobility

To be able to assess arm mobility to the patient, it is needed that the smartwatch delivers data from both the accelerometer and gyroscope sensors at the same time.

The chosen method to calculate arm mobility specifies that the sampling rate of the accelerometer and gyroscope should be 50 Hz. The already predefined constant of Android, `SENSOR_DELAY_GAME`, gives the programmer this exact value. So the developed application use this value while registering both sensors. The indicated sampling frequency generates 50 values per second constantly.

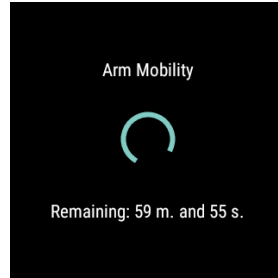
An example of the *SensorManager* with the linear acceleration and gyroscope sensors generating data is shown below:

Listing 4: Linear Acceleration and Gyroscope SensorManager

```
1  // To get an instance of the SensorManager
2  mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
3  // To get the default sensor for TYPE_LINEAR_ACCELERATION
4  mLinAccSensor =
        mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
5  // To get the default sensor for TYPE_GYROSCOPE
6  mGyroSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
7
8  // Registers a SensorEventListener for the given sensor at the given
        sampling frequency and the given maximum reporting latency
9  mSensorManager.registerListener(this, mLinAccSensor,
        SensorManager.SENSOR_DELAY_GAME, SensorManager.SENSOR_DELAY_GAME);
10 mSensorManager.registerListener(this, mGyroSensor,
        SensorManager.SENSOR_DELAY_GAME, SensorManager.SENSOR_DELAY_GAME);
11
12 switch (event.sensor.getType()) {
13     case Sensor.TYPE_LINEAR_ACCELERATION:
14         // Acceleration force along the x, y and z axis (excluding
            gravity).
15         mAccX = event.values[0];
16         mAccY = event.values[1];
17         mAccZ = event.values[2];
18
19         break;
20
21     case Sensor.TYPE_GYROSCOPE:
22         // Rate of rotation around the x, y and z axis.
23         mGyrX = event.values[0];
24         mGyrY = event.values[1];
25         mGyrZ = event.values[2];
26
27         break;
28 }
29
30 // Unregisters a listener for all sensors
31 mSensorManager.unregisterListener(this);
```

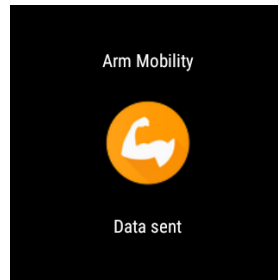
The six defined variables will hold the acceleration force along the axis (with-

out gravity) and the rate of rotation around the 3 axis. Every instance of these six values will be sent to the mobile phone application and will be processed to finally compute the arm mobility of the patient.



(a) Arm Mobility

Figure 21: Smartwatch application Arm Mobility while monitoring



(a) Arm Mobility

Figure 22: Smartwatch application Arm Mobility when monitoring time has concluded

Send data (to the mobile phone application)

Every single data generated by the heart rate, accelerometer or gyroscope of the smartwatch who uses the system has to be sent to the accompanying mobile phone application which acts as the processor unit of the system. The smartwatch does not retain any generated data at all because when one single data is generated it is automatically sent to the mobile phone.

This is implemented in the *onSensorChanged* function of the *SensorManager* of Android, so every time the sensor change its value, this function will be activated and its content processed. Inside of this function a call to the com-

munication system implemented on the application is done.

The communication and transfer of information from the smartwatch application to the smartphone application is done through the *Wearable Data Layer API*, which is part of Google Play services and provides a communication channel for the handheld and wearable apps. The API consists of a set of data objects that the system can send and synchronize over the wire and listeners that notify your apps of important events with the data layer.

To use this API properly we need to follow these steps:

1. Create a *DataMap* object with the data generated of one sensor in particular and send it to the data layer.
2. Create a *PutDataMapRequest* object, setting the path of the data item. The *path* string is a unique identifier for the data item that allows you to access it from either side of the connection and must begin with a forward slash.
3. Call *PutDataMapRequest.getDataMap()* to obtain a data map that you can set values on and set any desired values for the data map using the *put...()* methods.
4. Call *PutDataMapRequest.asPutDataRequest()* to obtain a *PutDataRequest* object. If a delay in syncing would negatively impact user experience, call *setUrgent()*.
5. Finally, call *DataApi.putDataItem()* to request the system to create the data item.

This protocol works for any kind of data the system would like to send to the mobile phone application. As it has been said, the system uses it to transfer data from the smartwatch to the smartphone.

An example of use of the *Wearable Data Layer API* with heart rate data is shown next (the same can be applied to the data generated by the accelerometer and gyroscope sensors in order to calculate energy expenditure and arm mobility):

Listing 5: Sending heart rate data with the Wearable Data Layer API

```

1  DataMap dataMapHRWearToMobile = new DataMap();
2
3  dataMapHRWearToMobile.putDouble("mHRValue", mHRValue);
4
5  String path = "/HRWearToMobile";
6  PutDataMapRequest putDataMapReq = PutDataMapRequest.create(path);
7
8  putDataMapReq.getDataMap().putAll(dataMap);
9
10 PutDataRequest putDataReq = putDataMapReq.asPutDataRequest();
11 putDataReq.setUrgent();
12
13 Wearable.DataApi.putDataItem(mHRGoogleApiClientWear, putDataReq)
14     .setResultCallback(new ResultCallback<DataApi.DataItemResult>() {
15         @Override
16         public void onResult(DataApi.DataItemResult dataItemResult) {
17             Log.i(TAG, "dataItemResult: " + dataItemResult.getStatus() + ",
18                 " + dataItemResult.getDataItem().getUri());
19         }
20     });

```

The *path* variable is essential to differentiate and encapsulate data. It depends if the application is sending heart rate, energy expenditure or arm mobility data and this needs to be perfectly specified so the application can perform work parallelly. In total, there are 3 channels of communication for the smartwatch to send data to the mobile phone application. Each one of them is independant but all of them coexist in the same habitat.

In the fragment of code included above about the Wearable Data Layer API, an instance of the *GoogleApiClient* is used on the *putDataItem* function. This API is part of Google Play services, so it is mandatory to include an own instance of *GoogleApiClient*. This explains why there are 3 channels of communications, because there are 3 instances of *GoogleApiClient* in total. The following code can be encapsulated in a function so we can connect and disconnect properly to the Google Play services (similar instances and functions should be created for energy expenditure and arm mobility):

Listing 6: Use of the Google API Client

```

1  mHRGoogleApiClientWear = new GoogleApiClient.Builder(this)
2      .addConnectionCallbacks(new GoogleApiClient.ConnectionCallbacks() {
3          @Override
4          public void onConnected(Bundle connectionHint) {
5              Log.d(TAG, "onConnected: " + connectionHint);
6              mHRGoogleApiClientWear.connect();
7          }
8          @Override
9          public void onConnectionSuspended(int cause) {
10             Log.d(TAG, "onConnectionSuspended: " + cause);
11             if (null != mHRGoogleApiClientWear &&
12                 mHRGoogleApiClientWear.isConnected()) {
13                 mHRGoogleApiClientWear.disconnect();
14             }
15         })
16     .addOnConnectionFailedListener(new
17         GoogleApiClient.OnConnectionFailedListener() {
18             @Override
19             public void onConnectionFailed(ConnectionResult result) {
20                 Log.d(TAG, "onConnectionFailed: " + result);
21             }
22         })
23     .addApi(Wearable.API)
24     .build();

```

Apart from this, which is used to send the data generated by the sensors, the smartwatch application sends a *DataMap* every frequency time passed so the mobile phone application will be able to know when is the moment to calculate results from the entirety of the data received. The elapsed time is sent too every frequency time so the system has a time reference.

In the system, this protocol is applied to both heart rate, energy expenditure and arm mobility estimations resulting in objects of data of these types and related calculation times.

4.3. Smartphone Application

4.3.1. System

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches (used in this project), each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. Android has the largest installed base of all operating systems of any kind. Android has been the best selling OS on tablets since 2013, and on smartphones it is dominant by any metric.

Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software required for accessing Google services.

The Android architecture has the following layers:

1. **Applications.** Located at the top of the Android software stack. Applications comprise both the native applications provided with the particular Android implementation (for example web browser and email applications) and the third party applications installed by the user after purchasing the device.
2. **Application Framework.** Is a set of services that collectively form the environment in which Android applications run and are managed. This framework implements the concept that Android applications are constructed from reusable, interchangeable and replaceable components.
3. **Libraries.** The Android Core Libraries (also referred to as the Dalvik Libraries) fall into three main categories, each of which merits an individual description: Dalvik VM Specific Libraries, Java Interoperability Libraries and Android Libraries.

4. **Android Runtime.** Consisting of two components, the Dalvik Virtual Machine (DVM) and core libraries responsible for process management.
5. **Linux Kernel.** Positioned at the bottom of the Android software stack, the Linux Kernel provides a level of abstraction between the device hardware and the upper layers of the Android software stack.

Android is a key technology in the system since the Android Wear version is used on the smartwatch and the mobile operating system links the wrist device with the web server application. The mobile phone application is represented as the processing unit.

A compatible Android device that runs Android 2.3 or higher will run the system because it will be able to use the Google Play services APIs to access the Wearable API, the Activity Recognition API and the HTTP library Volley which is used to transmit data to the web server application.

4.3.2. Implementation

The main components of the smartphone application can be summarized in the following ordered points:

- Present a home screen with a formulary to be filled with personal information that is needed to calculate and contextualize heart rate, energy expenditure and arm mobility.
- Send to the smartwatch application the configuration parameters (monitoring time and frequency time) of the heart rate, energy expenditure and arm mobility desired readings.
- Receive, from the smartwatch application, all the data generated by the sensors of this device.
- Process all the information received from the watch according to each methodology and to build valuable objects containing results of those processings of heart rate, energy expenditure and arm mobility.
- Use the Activity Recognition API to detect the activity the patients are performing at the same time they are calculating their heart rate, energy expenditure or arm mobility in order to contextualize this information and make it more valuable.

- Send these resulting objects to the web server application so the medical expert can graphically visualize this information related to each patient.
- Present a configurable and easy-to-use user interface (UI), where the patient can select the configuration parameters and check information about the readings that are in progress. The UI is divided between a home screen with a formulary and 3 different fragments contained in tabs dedicated to both heart rate, energy expenditure and arm mobility respectively.

Home screen displaying a formulary

The first thing when the user or the patient starts the smartphone application is a home screen displaying a formulary which needs to be filled with personal information that is essential to calculate and contextualize heart rate, energy expenditure and arm mobility. Once the formulary is completed, it is saved so the user will not have to fill it again and it can be changed whenever the user wants.

When the patient has filled every field of the formulary, everything will be ready to press on the *START* button and pass to the core of the application.

A representation of these steps are shown next:

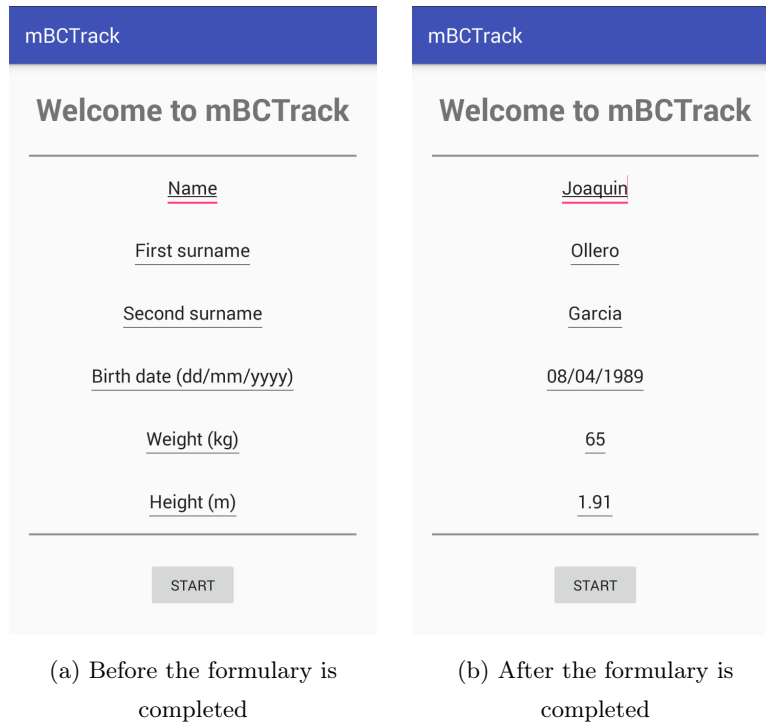


Figure 23: Smartphone application home screen

Send configuration parameters (to the smartwatch application)

Once the user has completed the initial formulary and has accessed to the core of the smartphone application, an interface containing 3 tabs will be displayed. Every tab is named under the information the patient would like to calculate, this is, heart rate, energy expenditure and arm mobility. The user can move through these tabs by just pressing on the name of the tab or sliding among them. Every tab and its content is independant from the others and they coexist altogether without intervening in each others matters. All three have a similar user interface but are dedicated to their subjects of concern.

From now on, the exposed examples in this section will be based on the heart rate tab but the behaviour for the other two data of interest is similar.

The first task the user must carry out to put the system in operation is to set the configuration parameters and send them to the smartwatch application. These parameters are set by default to 1 seconds and if the patient, for example, wants to perform an estimation of its heart rate during 10 minutes and have

results every 30 seconds, these values should be set on the monitoring time and frequency time fields respectively. Once this is done, immediately afterwards, by pressing the *SEND TO SMARTWATCH* button, these 2 parameters will be sent to the smartwatch application and the monitorization will be ready to begin whenever the patient wants.

The parameters are sent using the Wearable Data Layer API explained in the Send data (to the mobile phone application) paragraph.

An example of this task is presented next:

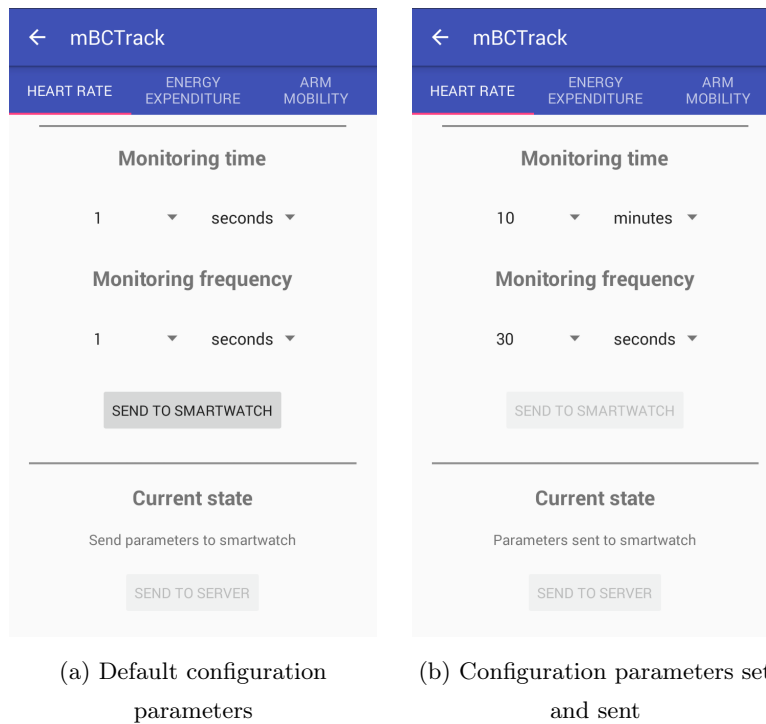


Figure 24: Smartphone application configuring parameters

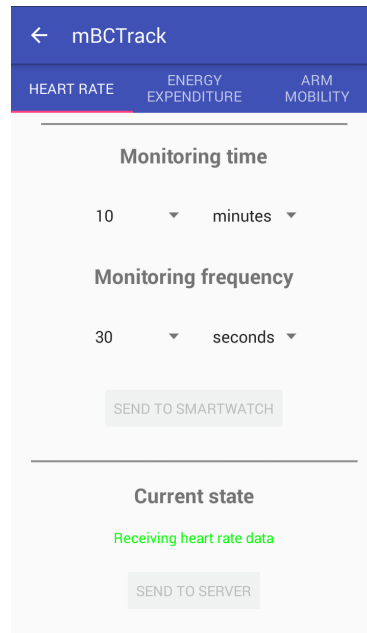
Receive sensor data (from the smartwatch application)

When the smartwatch application has received the configuration parameters the user can press the image button corresponding to heart rate, energy expenditure or arm mobility and the generation of data corresponding to this subjects will start and will be sent to the mobile phone application.

The reception of data is done with the same methods that were implemented

in the smartwatch application to receive the configuration parameters, as seen in Receive configuration parameters (from the mobile phone application).

Each of the three fragments receive data separately from the others and the reception of different data can be done paralelly. Also, when receiving data, the mobile phone application disables the buttons because it is receiving all this data and processing it and no user input is needed or required, like shown next:



(a) User interface while receiving data with buttons disabled

Figure 25: Smartphone application receiving heart rate data

Process sensor data

The processing of data is done separately in each fragment in order to build objects full of processed sensor data that the medical expert will be interested in. These objects will be built in relation with the frequency time indicated before.

Heart Rate

At the end of each frequency time, the mobile phone application will calcu-

late the average and the standard deviation of all heart rate values received in that lapse of time. Jointly, reference times related to those calculations will be obtained too.

The methodology the application follows to obtain these values is explained in System Design → Methods → Heart Rate.

Example of a performed test:

- **Monitorization time:** 1 minute.
- **Frequency time:** 15 seconds.
- **Received heart rate and time values [bpm, seconds from start]:**
[[75,2], [70,9], [65,14], [71,23], [73,27], [75,38], [77,40], [76,43], [69,51], [71,54], [70,59]].

Resulting objects:

- *ArrayList* of average heart rate (bpm): [70, 72, 76, 70].
- *ArrayList* of standard deviation of heart rate: [5, 1.41, 1, 1].
- *ArrayList* of times (s) of measured heart rate, same for average and standard deviation: [0-15, 15-30, 30-45, 45-60].

These 3 *ArrayList* objects will be sent to the web server application along with the recognized activities performed by the patient and the physiological information.

Energy Expenditure

Based on the 3 accelerometer values, at the end of each frequency time, the application will calculate the average activity counts and related METs. As in the heart rate subject, reference times related to those results will be stored too.

Activity counts and METs are determined according to the methodology explained in the subsection System Design → Methods → Energy Expenditure.

Resulting objects:

- *ArrayList* of average activity counts.
- *ArrayList* of average METs.

- *ArrayList* of times of measured energy expenditure, same for average activity counts and average METs.

Arm Mobility

The smartwatch application is sending constantly accelerometer and gyroscope values at 50 Hz.

In order to calculate arm mobility, the application follows the method presented in System Design → Methods → Arm Mobility.

Resulting objects:

- *ArrayList* of angles related to arm activity on the vertical axis.
- *ArrayList* of times of measured angles.

Recognition of activities performed

The Activity Recognition API of Google can be used to detect the activities that the device, and thus the patient, might be undertaking. It returns the most probable activity that the user is performing and the probability of the possibility that this activity is happening in a lapse of time continuously.

List of activities that can be detected (predefined in the Activity Recognition API):

- In vehicle.
- On bicycle.
- On foot.
- Running.
- Still.
- Tilting.
- Unknown.
- Walking.

Once the *GoogleApiClient* instance has connected, a *PendingIntent* is needed to be created that goes to the related *IntentService* and passed to the *ActivityRecognitionApi*. It is also required to set the desired time between activity detections. Larger values result in fewer activity detections while improving battery life. A value of 0 results in activity detections at the fastest possible rate. For this application, we use the frequency time value to try to get activities at the same rate we obtain results from the sensors of the smartwatch application.

Functions to request activity updates and remove activity updates:

Listing 7: Requests and removes activity updates of the Activity Recognition API

```

1  // When starts, Request Activity Updates
2  ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(
3      mHRGoogleApiClientAR,
4      mFrequencyTimeHR,
5      getActivityDetectionPendingIntent()
6  );
7
8  // When ends, Remove Activity Updates
9  ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(
10     mHRGoogleApiClientAR,
11     getActivityDetectionPendingIntent()
12 );

```

The algorithm followed to detect new activities and their confidences values is shown next. The application keeps track of time references so it is possible to contextualize the activities detected in a frame of time and relate them to data from the sensors:

Listing 8: Activity Recognition pending intent and broadcast receiver

```

1  private PendingIntent getActivityDetectionPendingIntent() {
2      Intent intent = new Intent(this,
3          DetectedActivitiesIntentService.class);
4
5      // We use FLAG_UPDATE_CURRENT so that we get the same pending intent

```

```

        back when calling requestActivityUpdates() and
        removeActivityUpdates().
5     return PendingIntent.getService(getActivity(), 0, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
6 }
7
8 public class ActivityDetectionBroadcastReceiverHR extends
    BroadcastReceiver {
9
10    @Override
11    public void onReceive(Context context, Intent intent) {
12
13        // Gets mostProbableActivityType and
14        // mostProbableActivityConfidence from IntentService that
15        // handles incoming intents that are generated as a result of
16        // requesting activity updates
17        String mostProbableActivityType =
18            intent.getStringExtra("mostProbableActivityType");
19        int mostProbableActivityConfidence =
20            intent.getIntExtra("mostProbableActivityConfidence", 0);
21
22        // Calculate the elapsed time in seconds (from the milliseconds
23        // timestamps)
24        mNewTime = (int) ((System.currentTimeMillis() / 1000) -
25            mStartTimestamp);
26
27        mTimeAndActivity.add(mOldTime + "-" + mNewTime + ", " +
28            mostProbableActivityType);
29
30        mOldTime = mNewTime;
31
32        mConfidence.add(mostProbableActivityConfidence);
33    }
34 }

```

Resulting objects:

- *ArrayList* of detected activities.
- *ArrayList* of detected activities confidences values.

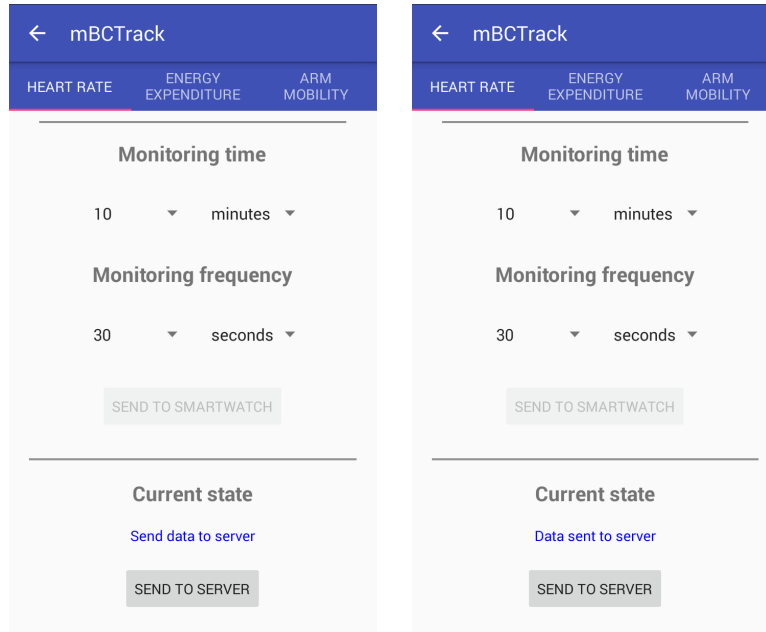
- *ArrayList* of times (s) of detected activities.

Two conclusions about the activity recognition functionality:

1. The Activity Recognition API is used on the mobile phone application rather than in the smartwatch application because is not available on Android Wear. Google has recently announced that it will be available in the coming future. The system imposes that the mobile phone has to accompany at all times the smartwatch, so the activities can be recognized anyway.
2. With the results obtained by using this API and because the recognition occurs at the exactly same time the readings from the sensors, the medical expert will be able to contextualize better the data generated by the sensors. For example, energy expenditure will make more sense if it can be known which physical activity the patient was performing.

Send results (to the web server application)

In the smartphone application, when the monitoring time has already finished, an intuitive message will appear in the user interface indicating that the objects that were built while receiving the data are ready to be sent to the web server application. The *SENT TO SERVER* button becomes able to be pressed and when this is done the data is automatically sent to the server. This is shown as it follows:



(a) Data ready to be sent to the web server application

(b) Data is already sent to the web server application

Figure 26: Smartphone application communication with web server application

To perform this operation, the system makes use of *Volley*, an HTTP library that makes networking for Android apps easier and faster. This library offers the following benefits: automatic scheduling of network requests, multiple concurrent network connections, support for request prioritization, among others.

The general protocol to use the Volley library can be summarized in 3 simple points:

- Instantiate the RequestQueue.
- Request a response from the provided URL.
- Add the request to the RequestQueue.

An independant function to complete this task has been implemented in the application, and the first thing it carries out is to build a $Map<String, Object>$ object with all the processed data obtained during the processing stage, which will be sent to the server.

The *user_id* variable is an identifier with the initials of the name and surnames of the patient and the date of birth. This unique identifier will relate the sent information with the data stored in the database of the server. Every object has a *date* attached because the data will be classified and accessed by days.

Listing 9: Builds a Map object to be sent with the Volley library

```
1  RequestQueue queue = Volley.newRequestQueue(this);
2
3  Calendar calendar = Calendar.getInstance();
4  String date = calendar.get(Calendar.DAY_OF_MONTH) + "/" + (int)
    (calendar.get(Calendar.MONTH) + 1) + "/" +
    calendar.get(Calendar.YEAR);
5
6  Map<String, Object> params = new HashMap<>();
7  params.put("user_id", mUserId);
8  params.put("date", date);
9  params.put("test_type", "HR");
10 params.put("dataHR", mHRArrayList);
11 params.put("dataSD", mSDArrayList);
12 params.put("dataT", mTimeArrayList);
13 params.put("dataTA", mTimeAndActivity);
14 params.put("dataC", mConfidence);
```

Volley provides the `JsonObjectRequest` class which requests for retrieving a `JSONObject` response body at a given URL, allowing for an optional `JSONObject` to be passed in as part of the request body. This is an ideal class to send the data because it will be easier to retrieve on the server.

The `JsonObjectRequest` presents in its class body is the *POST* method to send data, the *URL* of the server and the *Map* that was created before and added at this point to the class.

Listing 10: Sending an object with the Volley library

```
1  JsonObjectRequest request =
2      new JsonObjectRequest(
3          Request.Method.POST,
```

```

4         "http://aplicacion-nahar360.rhcloud.com/mobiletoserverhr",
5         new JSONObject(params),
6         new Response.Listener<JSONObject>() {
7             @Override
8             public void onResponse(JSONObject response) {
9                 Log.i("Response ", response.toString());
10            }
11        },
12        new Response.ErrorListener() {
13            @Override
14            public void onErrorResponse(VolleyError error) {
15                Log.i("Error ", String.valueOf(error));
16            }
17        }
18    )
19    {
20        @Override
21        public Map<String, String> getHeaders() throws AuthFailureError {
22            Map<String, String> headers = new HashMap<String, String>();
23
24            headers.put("Content-Type", "application/json;
25                charset=utf-8");
26
27            return headers;
28        }
29    };
30
31
32    queue.add(request);

```

When this action is performed, the use of the handheld applications has concluded. This means that the patient has generated specific sensor data from the smartwatch application. Then, the data is sent to the mobile phone application and processed according to related algorithms. Finally it is sent to the web server application which will receive and store on the databases this information so the medical expert will be able to visualize it.

The number of readings that the user can perform is limitless. The process

can be repeated anytime it is desired, but medical experts recommend to perform one heart rate test, one energy expenditure test and one arm mobility test per day.

4.4. Web Server Application

A web server application has been implemented in order to give access to the medical expert in a comfortable way so it is possible to check out all the information about its related patients and the tests they have performed from their handheld applications. An intuitive user interface will display charts based on the data received from the mobile phone application. When the data arrives on the server is automatically stored on the associated databases, depending on which test has been sent, and will be consequently showed in the corresponding charts.

The web server application has been implemented in *Python* and *Flask* programming languages for the back-end. *MongoDB* has been used to build the databases and the front-end is being built by *HTML5*, *Bootstrap*, *nvd3* (for charts) and *JavaScript*.

The main goals of the Web Server application can be schematized in the following points:

- Provide a server that can be hosted anywhere (local or cloud) prepared to receive data from the mobile phone application.
- Hold a database system to manage the personal information of patients and their related tests performed on the smartwatch and mobile phone applications.
- Show in an easy-to-understand manner the charts of the performed tests in order to obtain conclusions about the rehabilitation that the patients are following.
- Present a front-end application where the medical expert can log in and manage all the information that is stored in the databases. Show menus, tabs, formularies, tables and charts to ease the use of the platform.

4.4.1. System

Python and Flask

- **Python** is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python can serve as a scripting language for web applications, e.g., via *mod_wsgi* for the Apache web server. Web frameworks like Django or Flask support developers in the design and maintenance of complex applications.

- **Flask** is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. Applications that use the Flask framework include Pinterest, LinkedIn and the community web page for Flask itself.

Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

The following code shows a simple web application that prints "Hello World!":

Listing 11: Flask basic example

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route("/")
5 def hello():
6     return "Hello World!"
7
8 if __name__ == "__main__":
9     app.run()
```

MongoDB

MongoDB is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

MongoDB works on concept of database, collection and document:

- *Database* is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
- *Collection* is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
- *Documents* are a set of key-value pairs and they have dynamic schema, which means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

HTML5, Bootstrap, nvd3 and JavaScript

- **HTML5** is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.
- **Bootstrap** is a free and open-source front-end web framework for designing websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only. Bootstrap is compatible with the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera and Safari browsers. It supports responsive web design which means that the layout of web pages adjusts

dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone).

- **nvd3** builds re-usable charts and chart components for d3.js without taking away the power that d3.js gives.

Examples of charts that can be built with nvd3 are simple line, discrete bar and pie charts among others.

- **JavaScript** is a high-level, dynamic, untyped, and interpreted programming language. Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

4.4.2. Implementation

Connection and configuration of the web server

The web server is allocated in a localhost environment on OpenShift which is a Red Hat's Platform-as-a-Service (PaaS) that allows developers to quickly develop, host, and scale applications in a cloud environment, so the application can be accessed from any device with an Internet connection. OpenShift guides the user to create an account and to configure the Python server and install the MongoDB package.

Both servers must connect to the server itself and to the MongoDB database. This is shown as it follows:

Listing 12: Connection to localhost and OpenShift and to the MongoDB database

```
1 // localhost
2 // connection to Mongo
3 connection = MongoClient('localhost', 27017)
4 // connection to the db 'local_db'
```

```

5 mydb = connection['local_db']
6 host = "http://localhost:5000"
7
8 // openshift
9 // connection to Mongo
10 connection = MongoClient(os.environ['OPENSIFT_MONGODB_DB_URL'])
11 // connection to the db 'openshift_db'
12 mydb = connection['openshift_db']
13 host = "http://aplicacion-nahar360.rhcloud.com"

```

All OpenShift applications are built around a Git source control workflow where the developer codes locally, then upload changes to the server which runs a number of hooks to build and configure the web server application, and finally restarts the application. Once changes are made, the developer will need to *add* and *commit* those changes; *add* tells Git that a file or set of files will become part of a larger check in, and *commit* completes the check in. Git requires that each commit have a message to describe it. Finally, everything is ready to send the changes to the application through the *push* instruction. The usage of the three instruction to make changes is expressed next:

Listing 13: Git commands to upload changes to OpenShift

```

1 $ git add --all
2 $ git commit -m "Description of changes"
3 $ git push

```

Once both connections are done, a structure for the web application can be built and deployed. The application starts with a login page which gives access to the homepage which is divided in 3 main sections: a section to select a patient and visualize its tests, the management of the database of patients and the management of the database of tests.

A mapsite of the application can be seen below:

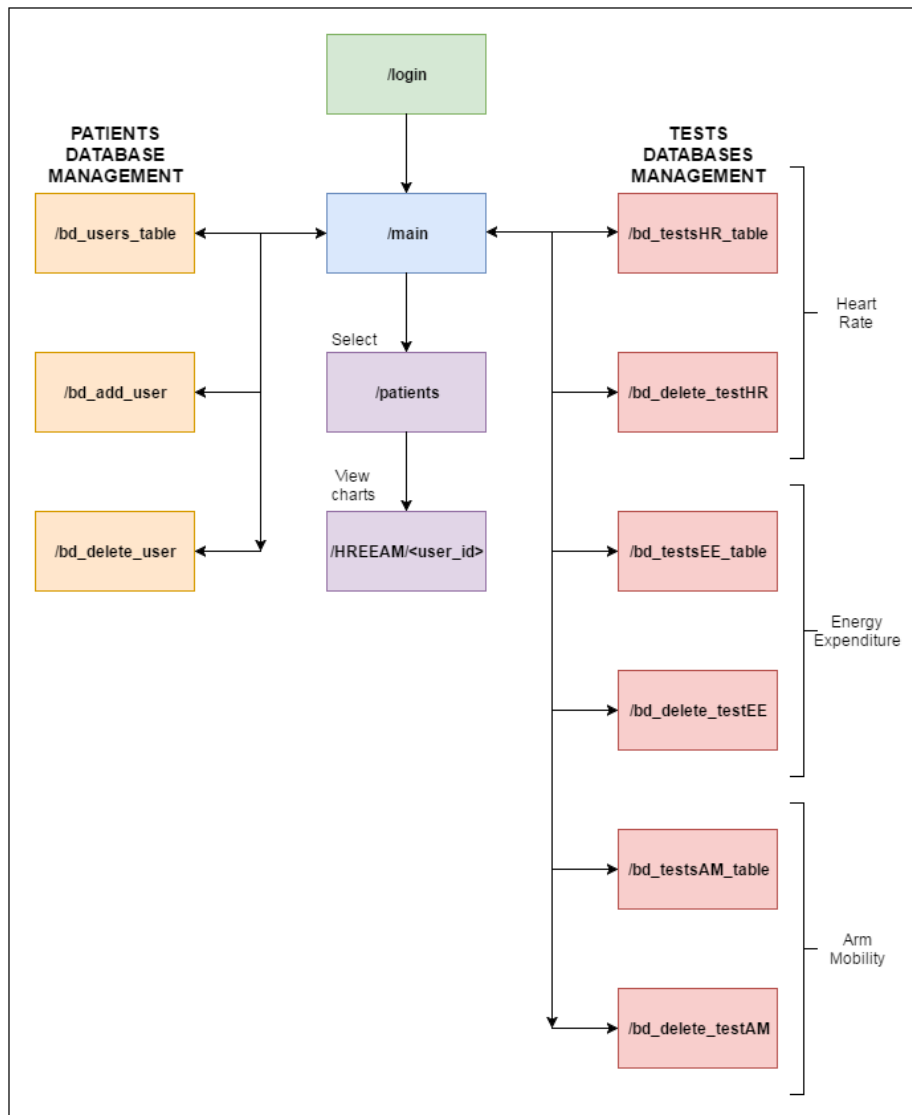


Figure 27: Structure of the Web Server application website

Patients database management

The medical expert will be able to manage the database dedicated to the patients and their personal information adding new patients, deleting the ones that are no longer required or simply checking all the existing patients.

The structure of the database of patients has the following structure:

Patient	
<i>Field</i>	<i>Description</i>
<code>_id</code>	A 12-byte ObjectId value given by MongoDB
<code>user_id</code>	An unique identifier that relates the patient with its tests
Name	Name of the patient
First surname	First surname of the patient
Second surname	Second surname of the patient
Birth date	Birth date of the patient
Height	Height (m) of the patient
Weight	Weight (kg) of the patient
Photo	Path to the <i>static</i> folder where the photo is allocated

Table 1: Patient database

- To see all the existing patients, we use the MongoDB *find* instruction and the next table is displayed so the medical expert can check all the information:

Listing 14: MongoDB find instruction

```

1 // return all patients
2 list(patient.find({}))

```









<code>_id</code>	<code>user_id</code>	Name	First surname	Second surname	Birth date	Height	Weight	Photo
579e4bc0ec17147e9d7e4c27	MGP01011960	Maria	Gonzalez	Perez	01/01/1960	160	60	/static/maria.jpg 
579e4bfbec17147e9d7e4c26	AMP01011961	Adela	Martinez	Perez	01/01/1961	161	61	/static/adela.jpg 
579e4c0bec17147e9d7e4c28	BGP01011962	Beatriz	Garcia	Perez	01/01/1962	162	62	/static/beatriz.jpg 
579e4c22ec17147e9d7e4c27	HGP01011963	Helena	Gomez	Perez	01/01/1963	163	63	/static/helena.jpg 
579e4c37ec17147e9d7e4c29	GTP01011964	Gabriela	Torres	Perez	01/01/1964	164	64	/static/gabriela.jpg 
579e4c48ec17147e9d7e4c28	ICP01011965	Isabel	Cruz	Perez	01/01/1965	165	65	/static/isabel.jpg 
579e4c58ec17147e9d7e4c2a	PFP01011966	Patricia	Ferrer	Perez	01/01/1966	166	66	/static/patricia.jpg 
579e4c69ec17147e9d7e4c29	REP01011967	Rocio	Estevez	Perez	01/01/1967	167	67	/static/rocio.jpg 

Figure 28: Displaying all the patients of the database of patients

- To add a new patient, it is done through the *insert* instruction and the next formulary is displayed so the medical expert can fill the information

of the patients:

Listing 15: MongoDB insert instruction

```
1 // add patient
2 patient.insert({
3     "user_id": name[:1] + surname1[:1] + surname2[:1] +
4         birthdate.replace("/", ""),
5     "name": name,
6     "surname1": surname1,
7     "surname2": surname2,
8     "birthdate": birthdate,
9     "height": int(height),
10    "weight": int(weight),
11    "photo": "/static/" + name.lower() + ".jpg"})
```

Name:

First surname:

Second surname:

Birth date (dd/mm/yyyy):

Height (m):

Weight (kg):

Add

Figure 29: Formulary to add new patients on the web server application

- To delete a patient, a list of all patients is displayed along with a button to execute the *remove* instruction:

Listing 16: MongoDB remove instruction

```
1 // delete a patients with ObjectId
2 patient.remove({"_id": ObjectId(id)})
```

_id	user_id	Name	First surname	Second surname	Birth date	Height	Weight	Photo	Delete
579e4bc0ec17147e9d7e4c27	MGP01011960	Maria	Gonzalez	Perez	01/01/1960	160	60	/static/maria.jpg	
579e4bfbec17147e9cc2cd26	AMP01011961	Adela	Martinez	Perez	01/01/1961	161	61	/static/adela.jpg	
579e4c0bec17147e9d7e4c28	BGP01011962	Beatriz	Garcia	Perez	01/01/1962	162	62	/static/beatriz.jpg	
579e4c22ec17147e9cc2cd27	HGP01011963	Helena	Gomez	Perez	01/01/1963	163	63	/static/helena.jpg	
579e4c37ec17147e9d7e4c29	GTP01011964	Gabriela	Torres	Perez	01/01/1964	164	64	/static/gabriela.jpg	
579e4c48ec17147e9cc2cd28	ICP01011965	Isabel	Cruz	Perez	01/01/1965	165	65	/static/isabel.jpg	
579e4c58ec17147e9d7e4c2a	PFP01011966	Patricia	Ferrer	Perez	01/01/1966	166	66	/static/patricia.jpg	
579e4c69ec17147e9cc2cd29	REP01011967	Rocio	Estevez	Perez	01/01/1967	167	67	/static/rocio.jpg	

Figure 30: Deleting patients through an intuitive button

Tests database management

The server application must have different URLs so the mobile application can send the data related to heart rate, energy expenditure and arm mobility tests. Along with this, once the data is received on the server, must be automatically stored on the corresponding databases.

The request of data sent by the mobile phone application is done on the web server, for example to retrieve the `user_id`, through the following instruction:

Listing 17: Request of data sent from the mobile application to the server

```
1 user_id = request.get_json(force=True)['user_id']
```

It does not matter what type of data the application is trying to obtain because it is encapsulated on a JSON object. So the application proceeds like this and retrieves all values sent. Thereupon, they are inserted on the databases with the MongoDB instruction *insert*. Finally, the implemented method should respond to the mobile application if the data have been received correctly or not. The medical expert will be able to delete tests to with the same functionality patients can be deleted.

There are three different databases related to the tests than can be performed from the handheld applications, this is, heart rate tests, energy expenditure tests and arm mobility tests.

The structure of these databases is presented next:

■ Heart Rate database:

Heart Rate Test	
<i>Field</i>	<i>Description</i>
_id	A 12-byte ObjectId value given by MongoDB
user_id	An unique identifier that relates the patient with its tests
Date	Date of the test
Test type	Heart Rate (HR)
Heart Rate	List of Heart Rate values
Heart Rate Time	List of Heart Rate Time values
Standard Deviation	List of Standard Deviation values
Standard Deviation Time	List of Standard Deviation Time values
Activity and Time	List of Activity and Time values
Confidence	List of Confidence of activities values

Table 2: Heart Rate Test database

■ Energy Expenditure database:

Energy Expenditure Test	
<i>Field</i>	<i>Description</i>
_id	A 12-byte ObjectId value given by MongoDB
user_id	An unique identifier that relates the patient with its tests
Date	Date of the test
Test type	Energy Expenditure (EE)
Activity Counts	List of Activity Counts values
Activity Counts Time	List of Activity Counts Time values
METs	List of values
METs Time	List of METs Time values
Activity and Time	List of Activity and Time values
Confidence	List of Confidence of activities values

Table 3: Energy Expenditure Test database

- Arm Mobility database:

Arm Mobility Test	
<i>Field</i>	<i>Description</i>
_id	A 12-byte ObjectId value given by MongoDB
user_id	An unique identifier that relates the patient with its tests
Date	Date of the test
Test type	Arm Mobility (AM)
Arm activity forearm angles	List of Arm activity forearm angles values
Arm activity forearm angles Time	List of Arm activity forearm angles Time values
Activity and Time	List of Activity and Time values
Confidence	List of Confidence of activities values

Table 4: Arm Mobility Test database

Visualization of the tests of the patients

To visualize the data of the tests sent by the patients, on the *See patients and tests* section, pictures of all patients are displayed so the medical expert can click on the patient of interest like the following figure shows:

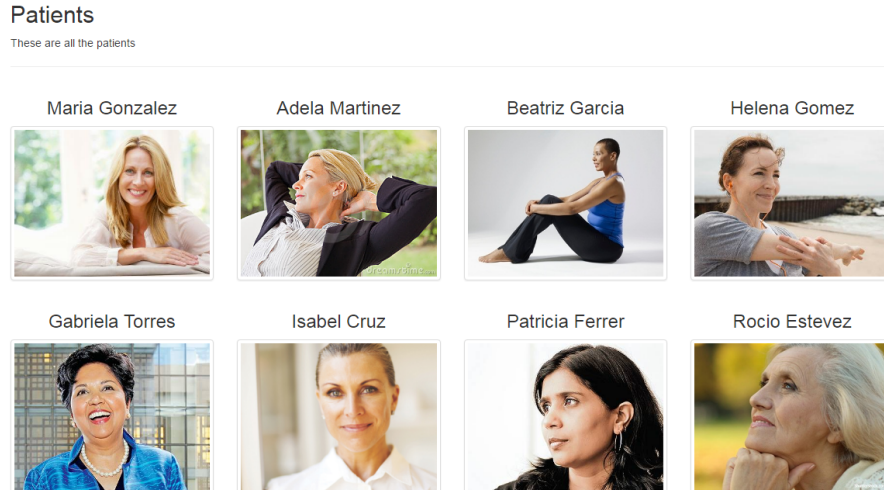


Figure 31: Deployment of patients to select by clicking on picture

When clicked on one, the medical expert access to a section where patients and test are related. This page is dominated by 3 different tabs, one each of

them corresponding to heart rate, energy expenditure or arm mobility tests respectively. It is possible to navigate through them to see related tests when selecting a specific date from the date picker field.

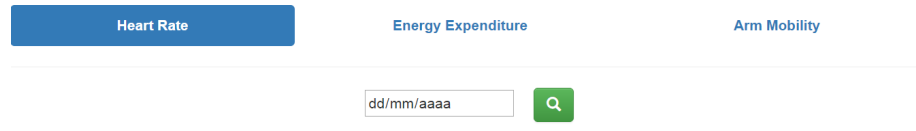


Figure 32: Heart Rate, Energy Expenditure and Arm Mobility tabs with date selector

The heart rate tab will display the following charts:

- **Heart rate (average) - line chart.**
X-axis: time values (seconds).
Y-axis: average heart rate values (bpm).
- **Standard deviation - line chart.**
X-axis: time values (seconds)
Y-axis: standard deviation of heart rate values.
- **Activity Recognition HR - bar chart.**
X-axis: time values (seconds) and activities performed
Y-axis: confidence of activities.

The energy expenditure tab will display the following charts:

- **Activity counts (average) - line chart.**
X-axis: time values (seconds)
Y-axis: average activity counts values (bpm).
- **METs - line chart.**
X-axis: time values (seconds)
Y-axis: average MET values (kcal / kg * h).
- **Activity Recognition EE - bar chart.**
X-axis: time values (seconds) and activities performed
Y-axis: confidence of activities.

The arm mobility tab will display the following charts:

- **Arm activity with angles measured in the vertical axis.**
X-axis: time values (seconds)
Y-axis: angles measured between -90 and 90 degrees.

■ Activity Recognition AM.

X-axis: time values (seconds) and activities performed

Y-axis: confidence of activities.

To build the charts using nvd3, first, the data of one test related to a type test, patient and date, must be retrieved from the tests database. This is done through a script and a proper URL:

Listing 18: Retrieving test data from the database

```
1  // to establish parameters
2  function viewtesthr() {
3
4      // retrieves date from date picker
5      date = new Date($('#mydatehr').val())
6      date = date.getDate() + "/" + (parseInt(date.getMonth()+1) + "/" +
          date.getFullYear()
7
8      // hr chart
9      var options = {
10         url: "/get_dataHR",
11         x_title: "Time (s)",
12         y_title: "Heart Rate (average) (bpm)",
13         id: "#charthr",
14         key: "Heart Rate",
15         color: "#ff0000",
16         ydomain: [50,90]
17     }
18
19     var charthr = new charthrsd(options)
20
21     charthr.show(date)
22 }
23
24 // retrieves data from database
25 function charthrsd(options) {
26     var data = undefined
27     var thisHolder = this
28     var options = options
29
30     this.show = function(){
```



```

31
32     // specific url related to one test type
33     var url = myhost + options.url
34     // related to one patient and date
35     var request_data = {user_id : user_id, date : date}
36
37     $.ajax({
38         url: url,
39         contentType: "application/json; charset=utf-8",
40         type: "post",
41         data: JSON.stringify(request_data),
42         success: function(result){
43             data = result // holds retrieved data from database
44             var chart = $(options.id)
45             $(chart).show()
46             var message = chart.parent().find('.message')
47             $(message).html("")
48             thisHolder.draw(result) // function to build the chart
49         },
50         // if no data is retrieved from database
51         error: function(error){
52             var chart = $(options.id)
53             $(chart).hide()
54             var message = chart.parent().find('.message')
55             $(message).html("No data found. Choose another day.")
56         }
57     });
58 }
59 }

```

The `nvd3` code that builds a *lineChart* with the values retrieved from the database is the following:

Listing 19: Building a *lineChart* with `nvd3`

```

1  this.draw = function(data) {
2
3      var series = [];
4
5      // to build series array

```

```

6     for(var i = 0; i < data.y_values.length; i++) {
7         series.push({
8             x: data.x_values[i],
9             y: data.y_values[i]
10        });
11    }
12
13    // nvd3 functions
14    nv.addGraph(function() {
15        var chart = nv.models.lineChart();
16
17        chart.useInteractiveGuideline(true); // tooltips and guideline
18
19        chart.x(function(d) {return d.x})
20        chart.y(function(d) {return d.y})
21
22        chart.showLegend(true);
23
24        // axis configuration
25        chart.showXAxis(true);
26        chart.xAxis.axisLabel(options.x_title);
27
28        chart.yAxis.axisLabel(options.y_title);
29        chart.showYAxis(true);
30        chart.yDomain(options.ydomain);
31
32        d3.select(options.id)
33            .datum([
34                {
35                    key: options.key,
36                    values: series,
37                    color: options.color,
38                    area: true
39                }
40            ])
41            .transition().duration(500).call(chart);
42
43        nv.utils.windowResize(
44            function() {
45                chart.update();
46            }
47        );
48    });

```

```

47         return chart;
48     });
49 }

```

Same code can be applied to build other charts with other data from the database. For example, energy expenditure charts are built the same way as the heart rate charts.

Examples of tests, and thus charts, can be found in the System test subsection.

4.5. System usage

The complete utilization of the system is schematized as it follows, divided by the patient applications and the medical expert application:

Usage by the patient

1. Download the Android Wear application from the Google Play Store and configure it to link the smartwatch with the smartphone (and viceversa).
2. Enable Bluetooth connectivity on the mobile phone.
3. Check on the Android Wear application on the smartphone if the devices are connected (this is attempted automatically by Android if the configuration of the devices is done properly).
4. Start the smartphone application.
5. Fill the formulary with personal information (only the first time) and press *START*.
6. Start the smartwatch application while wearing the device on the wrist.
7. On the mobile phone application, set and send the configuration parameters (monitoring and frequency times) of heart rate, energy expenditure or arm mobility (or an independent combination of them), by pressing on *SEND TO SMARTWATCH*.
8. On the smartwatch application, start the desired reading by pressing on the corresponding image button.

9. Wait until the monitoring time is finished. A countdown appears on the smartwatch and the mobile phone application will indicate that data is being received.
10. Once the monitoring time has concluded, press on the *SEND TO SERVER* button on the smartphone application to send the processed data of the performed reading to the web server application.
11. Repeat process whenever it is desired to attempt more readings. One estimation of heart rate of 5 minutes, energy expenditure and arm mobility of some hours per day are recommended.

Usage by the medical expert

1. Access to the web server application and login (by default, *username* = admin, *password* = pass).
2. Select options of patients and tests databases if management of them is desired.
3. Access to the *Patients* section and click on one patient's picture to visualize performed readings of that precise patient.
4. Select between Heart Rate, Energy Expenditure or Arm Mobility tabs to view charts of these types of readings.
5. Choose a date from the date picker input field and press on the green button.
6. Charts corresponding to that patient, of that test type and on the selected date will be displayed if available. If not, a message will inform the medical expert that there is no data corresponding to that date.
7. Repeat process with other patients, test types and dates. Manage the databases adding or deleting patients and check numerical data from the tests received from the mobile phone applications of the patients.

5. System Test

In this section some tests performed with are presented. The devices used to attempt these tests are a LG G Watch R smartwatch, a Samsung Galaxy A5 smartphone and any device with Internet connection. The graphically representation of these tests show the functionality of the system. The displayed data began being monitored on the smartwatch, passing through the smart-phone application and ending in the web server application where are stored and represented as it follows.

5.1. Heart rate tests

Monitoring time: 3 minutes.

Frequency time: 30 seconds.

Observations: heart rate increased according to activity.

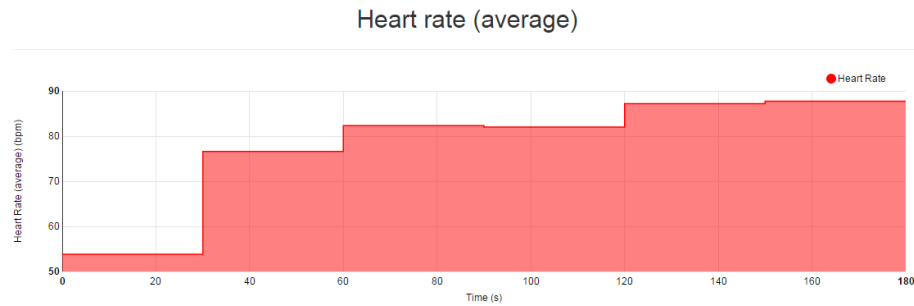


Figure 33: Heart rate (average) chart

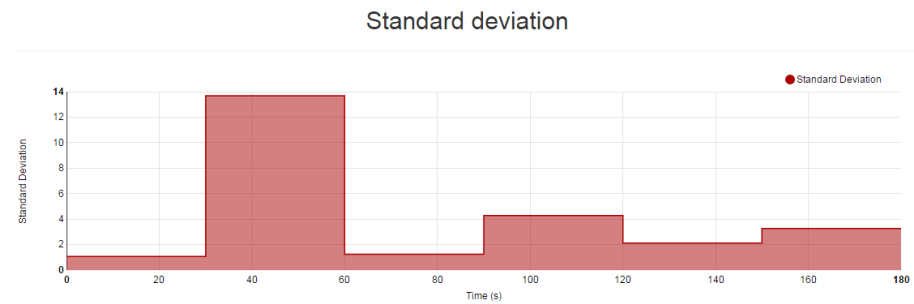


Figure 34: Standard deviation chart

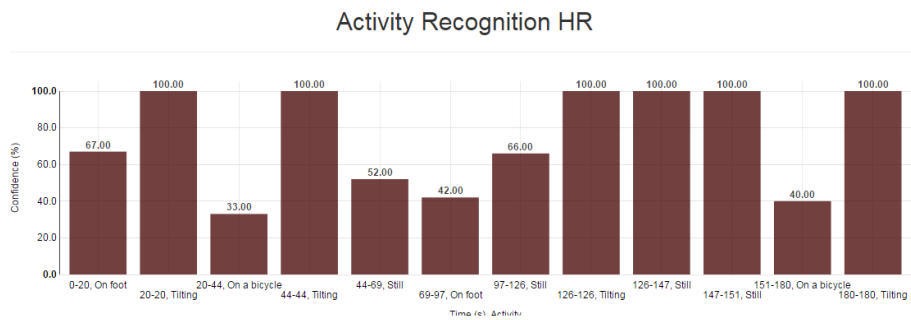


Figure 35: Activity Recognition Heart Rate chart

5.2. Energy expenditure tests

Monitoring time: 3 minutes.

Frequency time: 30 seconds.

Observations: first more intense activities were performed, then more relaxed.

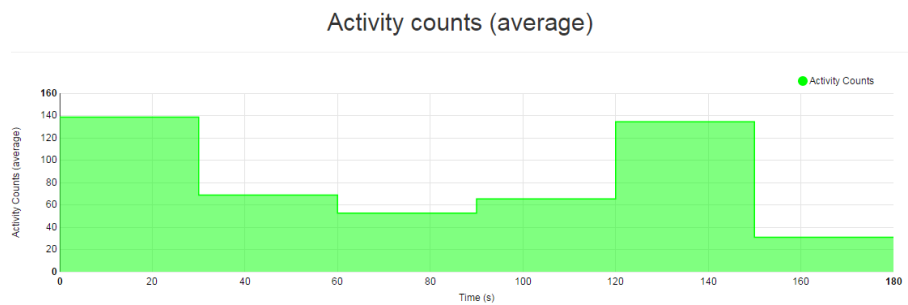


Figure 36: Activity counts (average) chart

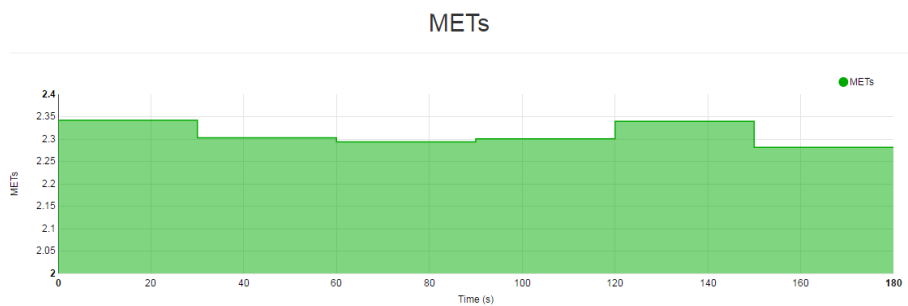


Figure 37: METs (average) chart

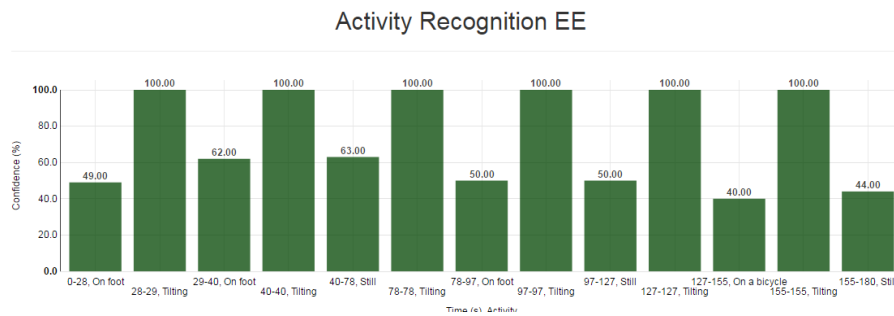


Figure 38: Activity Recognition Energy Expenditure chart

5.3. Arm mobility tests

Monitoring time: 1 minutes.

Frequency time: every data.

Observations: the arm was still at the beginning and then moving up and down continuously.

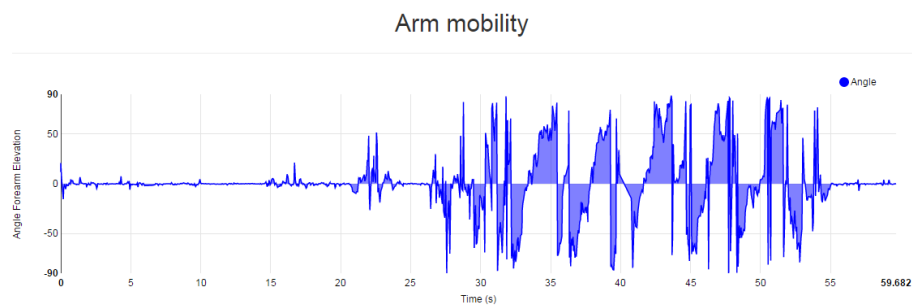


Figure 39: Arm mobility chart

6. Conclusions

This work has presented a self contained system to monitor breast cancer survivors. Comprising 3 different devices and applications, patients are able to monitor crucial parameters such as heart rate, energy expenditure and arm mobility from a smartwatch device running on the Android Wear operating system, send the sensor generated data to a mobile phone application running on the Android operating system so the data is processed and finally send this information to a web server application. A medical expert in charge of these patients is able to access to a web server application, from any environment or device with Internet connection, to manage patients, performed tests and visualize readings that the patients have performed comfortably in their home environment, without the need of going to a medical center.

6.1. Aims achieved

All objectives established in this work have been achieved, at least to provide the basic functionality of the system and all its applications. However, as will be commented in the section Conclusions → Future work some features can be extended or improved. The aims achieved are:

- Standalone system that allows monitoring of crucial parameters from sensors, such as heart rate, energy expenditure and arm mobility, to supervise patients recovery from breast cancer.
- Efficient communication between the developed applications on different devices.
- Storage mechanisms to save physiological information of patients and tests performed on a remote web server application.
- User-friendly interfaces to make the use of the applications as easy and comfortable as possible through accessible and interactive elements such as tabs and charts.
- Implementation of a compact and flexible system that can be expanded in the future with new improvements.

6.2. Future work

Given the huge importance of the wearable technology applied to the health environment, more specifically to the breast cancer and other diseases monitor-

ing, this functional, compact and flexible system could be improved with new functionalities. Some of them are highlighted as it follows:

- Calculate heart rate variability due to its importance for functional status assessment as described in System Design → Methods → Heart Rate. There could be 2 different possibilities that will enable the calculation of HRV. The first one is that Android Wear allows access to low implementation of the heart rate sensor and not only returns the value of heart rate in beats per second. The second one would be to use a specific device equipped with an electrocardiogram (ECG) sensor so it is possible to obtain the peak-to-peak (RR) intervals in heart beats.
- Implement activity recognition on the smartwatch instead of the mobile phone application as is it done and described in System Implementation → Smartwatch application → Implementation. Google has recently announced that future versions of Android Wear will implement this feature. It would be a better idea to implement this function on the smartwatch because it is the device that generates the sensor data, so the totality of information would come from the same source.
- Send data from the mobile phone application to the web server application not only once, at the end of every monitoring time, but every frequency time or implement a local database on the mobile phone application synchronized with the database of the server to correct the possibility of loss of data during the monitoring of data. If any error occurs while performing a reading, this data will not be saved anywhere and would be lost, so the reading must be started from the beginning.
- It could be a possibility to unify the heart rate, energy expenditure and arm mobility interfaces on the same user interface (in both the smartwatch and smartphone applications) so there are not 3 different tabs with almost the same information displayed. This could be achieved by implementing new options to select one of them or even a combination of them to perform readings.
- Improve the web server application user interface to transform it into a panel administration style so the navigation through the application becomes more dynamic.
- Medical records are the most confidential data individuals can have. In

this sense, proper encryption of medical data would be needed to be implemented to make the sent and stored data immune to possible attacks.

- The real automatic monitoring of heart rate, energy expenditure or arm mobility of patients would be achieved if these individuals would not have to perform any input on the system. The smartwatch and smartphone applications would still exist to give information related to the monitoring of parameters, but it would be the medical expert, according to its necessities, from the web server application the one that would send the monitoring and frequency times to the handheld applications and would activate the sensors so the readings are performed and the patient would only have to wear and carry the devices.
- Conduct experiments with patients in real life to validate the real functionality of the system. This study should be performed with a number of patients who have overcome breast cancer. Smartwatches running on the Android Wear operating system, for example the used LG G Watch R, should be given to these patients along with specific medical instructions to perform readings. These individuals could use their own mobile phone (if they already have one and it is an Android mobile phone) to monitor and send data to the web server application so the medical expert is able to check the performed readings. This kind of study would throw interesting and decisive conclusions about the functioning of the system in order to improve it and make it more accessible to the users that use it.

References

References

- [1] Claudia Arab, Daniel Penteado Martins Dias, Renata Thais de Almeida Barbosa, Tatiana Dias de Carvalho, Vitor Engracia Valenti, Tania Brusque Crocetta, Marcelo Ferreira, Luiz Carlos de Abreu, and Celso Ferreira. Heart rate variability measure in breast cancer patients and survivors: A systematic review. *Psychoneuroendocrinology*, 68:57–68, 2016.
- [2] Elena Caro-Moran, Carolina Fernandez-Lao, Noelia Galiano-Castillo, Irene Cantarero-Villanueva, Manuel Arroyo-Morales, and Lourdes Diaz Rodriguez. Heart rate variability in breast cancer survivors after the first year of treatments a case-controlled study. *Biological research for nursing*, 18(1):43–49, 2016.
- [3] Emil Jovanov. Preliminary analysis of the use of smartwatches for longitudinal health monitoring. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 865–868. IEEE, 2015.
- [4] Subhas Chandra Mukhopadhyay. Wearable sensors for human activity monitoring: A review. *IEEE Sensors Journal*, 15(3):1321–1330, 2015.
- [5] Ronald C Merrell and Charles R Doarn. m-health. *Telemedicine and e-Health*, 20(2):99–101, 2014.
- [6] Chang Liu, Qing Zhu, Kenneth A Holroyd, and Elizabeth K Seng. Status and trends of mobile-health applications for ios devices: A developer’s perspective. *Journal of Systems and Software*, 84(11):2022–2033, 2011.
- [7] Caitlin M Stackpool. The accuracy of various activity trackers in estimating steps taken and energy expenditure. 2013.
- [8] Mingui Sun, Lora E Burke, Zhi-Hong Mao, Yiran Chen, Hsin-Chen Chen, Yicheng Bai, Yuecheng Li, Chengliu Li, and Wenyan Jia. ebutton: A wearable computer for health monitoring and personal assistance. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, pages 1–6. ACM, 2014.

- [9] Sherif Mekky. Wearable computing and the hype of tracking personal activity.
- [10] Ankur Agarwal, Borko Furht, and Mamata Yenagi. Mobile medical and healthcare applications. *Handbook of Medical and Healthcare Technologies*, pages 3–15, 2013.
- [11] Sara Eriksen, Mattias Georgsson, Malin Hofflander, Lina Nilsson, and Jenny Lundberg. Health in hand: Putting mhealth design in context. In *Usability and Accessibility Focused Requirements Engineering (UsARE), 2014 IEEE 2nd International Workshop on*, pages 36–39. IEEE, 2014.
- [12] Maged N Kamel Boulos, Ann C Brewer, Chante Karimkhani, David B Buller, and Robert P Dellavalle. Mobile medical and health apps: state of the art, concerns, regulatory control and certification. *Online journal of public health informatics*, 5(3):229, 2014.
- [13] Susannah Fox and Maeve Duggan. Mobile health 2012. *Pew Research Center’s Internet & American Life Project [Internet]*, 2012.
- [14] Mladen Milosevic, Aleksandar Milenkovic, and Emil Jovanov. mhealth@uah: computing infrastructure for mobile health and wellness monitoring. *XRDS: Crossroads, The ACM Magazine for Students*, 20(2):43–49, 2013.
- [15] Val Jones, Valerie Gay, and Peter Leijdekkers. Body sensor networks for mobile health monitoring: Experience in europe and australia. In *Digital Society, 2010. ICDS’10. Fourth International Conference on*, pages 204–209. IEEE, 2010.
- [16] Jingzhao Li, Xueqin Wu, and Hui Chen. Research on mobile digital health system based on internet of things. *Electrical Power Systems and Computers*, pages 495–502, 2011.
- [17] Lucy E Dunne, Halley Profita, Clint Zeagler, James Clawson, Scott Gilliland, Ellen Yi-Luen Do, and Jim Budd. The social comfort of wearable technology and gestural interaction. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 4159–4162. IEEE, 2014.
- [18] Daniel Roggen, Stephane Magnenat, Markus Waibel, and Gerhard Troster. Wearable computing. *Robotics & Automation Magazine, IEEE*, 18(2):83–95, 2011.

- [19] Francis Collins. How to fulfill the true promise of mhealth. *Scientific American*, 307(1):16–16, 2012.
- [20] Melanie Swan. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *Journal of Sensor and Actuator Networks*, 1(3):217–253, 2012.
- [21] Andrew Miller. Fitness trackers. *XRDS: Crossroads, The ACM Magazine for Students*, 20(2):24–26, 2013.
- [22] Insoo Kim, Po-Hsiang Lai, Ryan Lobo, and Bruce J Gluckman. Challenges in wearable personal health monitoring systems. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 5264–5267. IEEE, 2014.
- [23] Efthimios Alepis, Maria Virvou, and Savas Drakoulis. Human smartphone interaction: Exploring smartphone senses. In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pages 44–48. IEEE, 2014.
- [24] Kyoungwhan Oh. The effects of brand, design, and price on intent to purchase an activity tracker. 2014.
- [25] Kai Kunze and Paul Lukowicz. Sensor placement variations in wearable activity recognition. *Pervasive Computing, IEEE*, 13(4):32–41, 2014.
- [26] Kristina Grifantini. How’s my sleep?: Personal sleep trackers are gaining in popularity, but their accuracy is still open to debate. *Pulse, IEEE*, 5(5):14–18, 2014.
- [27] Lei Jing, Zixue Cheng, Yinghui Zhou, Junbo Wang, and Tongjun Huang. Magic ring: A self-contained gesture input device on finger. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, page 39. ACM, 2013.
- [28] Tomoya Tanaka, Koji Sonoda, Sayaka Okochi, Alex Chan, Manabu Nii, Kensuke Kanda, Takayuki Fujita, Kohei Higuchi, and Kazusuke Maenaka. Wearable health monitoring system and its applications. In *Emerging Trends in Engineering and Technology (ICETET), 2011 4th International Conference on*, pages 143–146. IEEE, 2011.
- [29] Nike fuelband se. http://www.nike.com/us/en_us/c/nikeplus-fuelband, 2014. Accessed: 2015-01-22.

- [30] Fitbit flex. <http://www.fitbit.com/flex>, 2014. Accessed: 2015-01-22.
- [31] Jawbone up. <http://jawbone.com/up>, 2013. Accessed: 2015-01-22.
- [32] Samsung gear fit. <http://www.samsung.com/es/consumer/mobile-phone/wearables/wearables/SM-R3500ZKAPHE>, 2014. Accessed: 2015-01-22.
- [33] Misfit wearables shine. <http://misfit.com/products/shine>, 2014. Accessed: 2015-01-22.
- [34] Athos gear. <http://www.liveathos.com/apparel/gear>, 2014. Accessed: 2015-01-22.
- [35] Fin wearable ring. <http://www.finrobotics.com/>, 2015. Accessed: 2015-01-22.
- [36] Jolt sensor. <http://http://www.joltsensor.com>, 2015. Accessed: 2015-01-22.
- [37] Diane J Skiba. The connected age and wearable technology. *Nursing Education Perspectives*, 35(5):346–347, 2014.
- [38] Pebble. <https://getpebble.com/>, 2013. Accessed: 2015-01-22.
- [39] Kickstarter. <https://www.kickstarter.com>, 2009. Accessed: 2015-01-22.
- [40] Wallace Ugulino, Débora Cardador, Katia Vega, Eduardo Velloso, Ruy Milidiú, and Hugo Fuks. Wearable computing: accelerometers data classification of body postures and movements. *Advances in Artificial Intelligence-SBIA 2012*, pages 52–61, 2012.
- [41] Sebastian Witt. Wearable computing: Smart watches. *Fun, Secure, Embedded*, 2014.
- [42] Megan C Kelley. The impact of fitness technology on health outcomes. 2014.
- [43] Application framework. http://en.wikipedia.org/wiki/Application_framework, 2015. Accessed: 2015-01-22.
- [44] Samsung s.a.m.i. http://www.samsung.com/us/globalinnovation/innovation_areas, 2014. Accessed: 2015-01-22.
- [45] Apple healthkit. <https://developer.apple.com/healthkit/>, 2014. Accessed: 2015-01-22.

- [46] Open mhealth. <http://www.openmhealth.org/>, 2014. Accessed: 2015-01-22.
- [47] mhealthdroid. <https://github.com/mHealthTechnologies/mHealthDroid>, 2014. Accessed: 2015-01-22.
- [48] Angel Ruiz-Zafra, Eva Orantes-González, Manuel Noguera, Kawtar Benghazi, and Jose Heredia-Jimenez. A comparative study on the suitability of smartphones and imu for mobile, unsupervised energy expenditure calculi. *Sensors*, 15(8):18270–18286, 2015.
- [49] Kaspar Leuenberger, Roman Gonzenbach, Susanne Wachter, Andreas Luft, and Roger Gassert. A method to qualitatively assess arm use in stroke survivors in the home environment. *Medical & biological engineering & computing*, pages 1–10, 2016.
- [50] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7. IEEE, 2011.
- [51] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. Holgado, S. Lee, H. Pomares, and I. Rojas. Design, implementation and validation of a novel open framework for agile development of mobile health applications. *Biomedical Engineering Online*, 14(S2:S6):1–20, 2015.
- [52] Oresti Banos, Claudia Villalonga, Miguel Damas, Peter Gloesekoetter, Hector Pomares, and Ignacio Rojas. Physiodroid: Combining wearable health sensors and mobile devices for a ubiquitous, continuous, and personal monitoring. *The Scientific World Journal*, 2014, 2014.
- [53] Paolo Bonato. Advances in wearable technology and its medical applications. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 2021–2024. IEEE, 2010.
- [54] Paolo Bonato. Wearable sensors and systems. *Engineering in Medicine and Biology Magazine, IEEE*, 29(3):25–36, 2010.
- [55] Jérôme Couturier, Davide Sola, Giovanni Scarso Borioli, and Cristina Raiciu. How can the internet of things help to overcome current healthcare challenges. *Communications and Strategies*, (87):67, 2012.

- [56] Bruce H Dobkin and Andrew Dorsch. The promise of mhealth daily activity monitoring and outcome assessments by wearable sensors. *Neurorehabilitation and Neural Repair*, 25(9):788–798, 2011.
- [57] Lucy Dunne. Wearable technology. 2013.
- [58] Vivian Genaro Motti, Spencer Kohn, and Kelly Caine. Wearable computing: a human-centered view of key concepts, application domains, and quality factors. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 563–564. ACM, 2014.
- [59] Chanmi Hwang. Consumers’ acceptance of wearable technology: Examining solar-powered clothing. 2014.
- [60] Smita Jhajharia, SK Pal, and Seema Verma. Wearable computing and its application. *International Journal of Computer Science & Information Technologies*, 5(4), 2014.
- [61] James G Kahn, Joshua S Yang, and James S Kahn. Mobile health needs and opportunities in developing countries. *Health Affairs*, 29(2):252–258, 2010.
- [62] Ali Mehmood Khan. Wearable health monitoring system. pages 173–177, 2013.
- [63] Jeongeun Kim. Analysis of health consumers’ behavior using self-tracker for activity, sleep, and diet. *Telemedicine and e-Health*, 2014.
- [64] Santosh Kumar, Wendy J Nilsen, Amy Abernethy, Audie Atienza, Kevin Patrick, Misha Pavel, William T Riley, Albert Shar, Bonnie Spring, Donna Spruijt-Metz, et al. Mobile health technology evaluation: the mhealth evidence workshop. *American journal of preventive medicine*, 45(2):228–236, 2013.
- [65] Alain B Labrique, Lavanya Vasudevan, Erica Kochi, Robert Fabricant, and Garrett Mehl. mhealth innovations as health system strengthening tools: 12 common applications and a visual framework. *Global Health: Science and Practice*, 1(2):160–171, 2013.
- [66] Henrique MG Martins. Critical concepts in m-health technology development: Time, space, and mobility. *Telemedicine and E-Health Services, Policies, and Applications*, pages 140–150, 2012.

- [67] Yao Meng, Heung-Kook Choi, and Hee-Cheol Kim. Exploring the user requirements for wearable healthcare systems. In *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*, pages 74–77. IEEE, 2011.
- [68] Eleni Nasiopoulos, Evan F Risko, Tom Foulsham, and Alan Kingstone. Wearable computing: Will it make people prosocial? *British Journal of Psychology*, 2014.
- [69] Christine Zhenwei Qiang, Masatake Yamamichi, Vicky Hausman, Daniel Altman, and IS Unit. Mobile applications for the health sector. *World Bank*, 2012.
- [70] Jesse Jayne Rutherford. Wearable technology. *Engineering in Medicine and Biology Magazine, IEEE*, 29(3):19–24, 2010.
- [71] Thad Starner. Wearable computing: through the looking glass. In *Proceedings of the 17th annual international symposium on International symposium on wearable computers*, pages 125–126. ACM, 2013.
- [72] Steven R Steinhubl, Evan D Muse, and Eric J Topol. Can mobile health technologies transform health care? *JAMA*, 310(22):2395–2396, 2013.
- [73] Yongqiang Sun, Nan Wang, Xitong Guo, and Zeyu Peng. Understanding the acceptance of mobile health services: A comparison and integration of alternative models. *Journal of Electronic Commerce Research*, 14(2):183–200, 2013.
- [74] Joseph Wei. How wearables intersect with the cloud and the internet of things: Considerations for the developers of wearables. *Consumer Electronics Magazine, IEEE*, 3(3):53–56, 2014.