UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FINAL DEL

REALIZADO POR

TUTORIZADO POR

CURSO ACADÉMICO: 2020/2021

Acknowledgements

I would like to thank everyone who has made these four years at the UPV a period of continuous learning, self-improvement and the desire to continue learning.

First of all, I would like to thank the teachers I have had during this period because without them it would not have been possible to get this far. I would like to make a special mention to my tutor, Salvador, for guiding me and helping me during the course of the project. Thanks also to my colleagues in the ARA group, for all the little moments lived together inside and outside the classroom that I will always remember.

Secondly, I would like to thank Pranav Manoj for helping me to learn and comprehend the Flutter programming language when I felt stuck with the mobile app coding. His help was life-saving.

Thirdly, I would like to stress that I would not be here without the support and effort of my family. Thanks to my brother Javier who was the tester of the app and performed the shots to acquire the data. I also feel very grateful for all the encouragement from my parents during the process of making the project.

Finally, I would also like to mention my friends who supported me at all times, tried to keep me positive and trusted in me when I didn't even trust myself in finishing the project on time.

Carlos Blasco Andreu

Abstract

The goal of the project is to develop a wearable system devoted to measure the free throw shooting performance while practicing basketball. The device will obtain the required information using acceleration and orientation sensors. The data will be stored in a smartphone using Bluetooth Low Energy (BLE) and could be downloaded to a personal computer. The project is based on the MikroE Hexiwear development kit, which will be paired to the phone by means of an ad hoc application.

Keywords: Basketball, Inertial Measurement Unit, IMU, BLE, Wearable, Hexiwear, MikroE, Flutter.

Resumen

El objetivo del trabajo consiste en el desarrollo de un sistema wearable para la medida del rendimiento en el tiro libre durante la práctica de baloncesto. El dispositivo obtendrá la información necesaria gracias a sensores de aceleración y orientación. Los datos se almacenarán en un dispositivo móvil mediante comunicación Bluetooth Low Energy (BLE) y podrán descargarse a un ordenador personal. El proyecto se basa en el kit de desarrollo MikroE Hexiwear, el cual se emparejará al teléfono empleando una aplicación diseñada ex profeso.

Palabras Clave: Baloncesto, Unidad de medida inercial, IMU, BLE, Wearable, Electrónica Vestible, Hexiwear, MikroE, Flutter.

Resum

L'objectiu del treball consisteix en el desenvolupament d'un sistema wearable per la mesura del rendiment en el llançament de tirs lliurats en la pràctica de bàsket. El dispositiu obtindrà la informació necessaria gràcies a sensors d'acceleració i orientació. Les dades s'emmagatzemaran en un dispositiu mòbil mitjançant comunicació Bluetooth Low Energy (BLE) i podran descarregar-se a un ordinador personal. El projecte es basa en el kit de desenvolupament MikroE Hexiwear, el qual s'emparellarà amb el telèfon emprant una aplicació disenyada específicament.

Paraules clau: Bàsket, Unitat de mesura inercial, IMU, BLE, Wearable, Hexiwear, MikroE, Flutter.

Contents

List of Figures

Part I

Report

Chapter 1

Introduction

1.1 Scope of the project

The scope of the project is to develop and to validate a wearable system to analyze the shooting efficiency of free throws during basketball training practice.

1.2 Objectives

The overall objectives to be pursued during the project are the development of a prototype that allows the collection of movement data of interest in basketball free throws and the validation of an analysis methodology to be able to measure these shots with objective parameters.

The specific objectives are presented below. Firstly, the choice of a wearable device on the market that meets the specifications of the project. Specifically, the aim is to choose a technology that allows obtaining the physical magnitudes of interest that will be used to measure the performance of the shots made by the player. Also, familiarisation with the development environment and with the functionalities of the inertial measurement unit to be contained in the device.

The second objective is to understand the communication protocol that will allow the wearable's software to communicate with the mobile device, to obtain the necessary information from it. In addition, it is also intended to configure the sensors of the device without altering its internal code, through the communication protocols.

The third objective is the development of the mobile application including the different necessary functionalities: registering, representing and storing in a database the activities carried out, as well as being able to download this data for export to other devices.

Finally, the fourth objective is to develop and validate a methodology for analysing performance and assessing the quality of the recorded shots as a potential tool to support the players during training sessions.

1.3 Structure of the document

The structure followed for the development of the work has the following layout. In chapter 2, a brief introduction about basketball and an explanation of the importance of free shots in the games is presented. Then, in chapter 3, the main wearable developing kit devices on the market are analysed, as well as the available mobile operating systems, to subsequently choose and develop the technologies selected in chapter 4. Chapter 5 briefly introduces some of the concepts necessary to carry out the project. It will mainly deal with the representation and measurement of orientation and how to obtain it from the information provided by the sensors and its fusion employing Kalman filters. In chapter 6, the development and implementation of the prototype will be explained in more detail, followed by an explanation of the different results obtained in chapter 7. Finally, chapter 8 presents the conclusions of the work and, as an appendix, chapter 9 includes the code written for the mobile app.

Chapter 2

Basketball

2.1 Basketball as a sport

Basketball is one of the most popular team sports in the world. It was created by Canadian-American teacher James Naismith in 1891. In the Berlin games of 1936, it officially became Olympic.
In Spain, basketball is the second most practised sport, especially by children at schools. In 2019 there were 200.000 federated players in the country [18], which suggests that the total number is even higher. The United States is the country of basketball par excellence. There were about 25 million players in 2019 according to Sports & Fitness Industry Association (SFIA) [5].

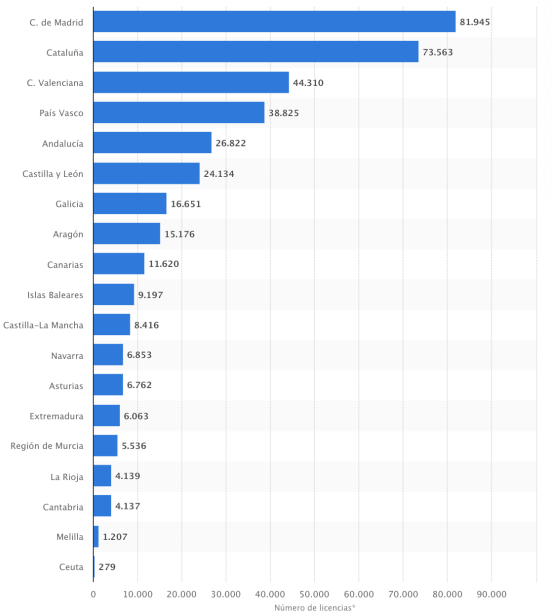

Figure 2.1: Federated basketbal players in Spain by Autonomous Community, year 2019.

Basketball is a precision sport that heavily depends on the ability and marksmanship of the players to introduce the ball inside the hoop to score points. This skill is so important that shooting becomes a preponderant priority when tailoring training sessions. Besides developing the proper technique, the mastering of shooting comes from repetition. In this scenario, statistics and historic values come in handy, allowing players and coaches to monitor their improvements. On the other side, if the results are not proper, they can be used to warn the team about what kind of shots basketballers shall focus on in the next training.

2.2 Classification of basketball shots

Shots can be divided into two categories: jump shot and free throw. The former is a shot performed by the player after he or she jumps and releases the ball at the peak of the elevation (ideally) to take advantage of the extra power given by the legs' flexion and extension. Free throws, on the other hand, consist of steady shots where the feet are not allowed to leave the ground [21]. They are vital in games since they can determine the outcome when the point difference between both teams is tight.

Figure 2.2: Player shooting a free thow [32]

2.2.1 Free throws

The complexity of free throws comes from the very situation in which they take place. They are performed by the player that has suffered a foul from the other team or when some opponent does a technical foul. As its name suggests, the basketballer can shoot freely, with no blockade from other players. Moreover, this freedom on the shot sometimes distracts even more due to the pressure he or she is under. In some cases, there will be supporters of the other team trying to divert the attention of the shooter. For the reasons mentioned previously it can be said that a good free thrower needs two components to succeed: a calm mind and exhaustive training in the mechanics of such shot through repetition.

Figure 2.3: Player showing the mechanical sequence in a free throw [34].

2.3 Opportunities for technology implementation inside basketball practice

Basketball is a dynamic sport with various motions of the ball influencing the gameplay dynamics, therefore incorporating a complete inertial sensing system would be ideal to capture basketball dynamics [31]. With the development of technology, MEMS-based IMUs are present more and more inside sports. In past years, several academic articles [31] [3] [19] [8] [28] [27] proved that shooting can be easily monitored and characterized. When talking about shooting performance systems, *ShotTracker* [2] was a pioneering device, partnering with NBA star Klay Thompson, one of the best shooters in the league. Nowadays, *ShotTracker* system is an evolved statistics giant that works with several teams in the American college league, the NCAA.

Chapter 3

Available technologies

In the following chapter, several alternatives to develop the project will be discussed. Characteristics and specifications are presented to select the most appropriate solutions.

3.1 Wearable technology

The so-called wearable technology includes all the electronic devices that are attached to some part of the user's body to interact with and to measure the desired magnitudes. Wearable devices are often paired with a smartphone app to display and process information.

From the early years of the 2010's decade, the users of such devices have grown significantly, partly due to technological development, lower consumer prices and integration as common use gadgets. Wearables are especially useful in sports and healthcare since the data that they collect can be interesting to keep track of the performance in a given activity or time. In sports, the athletes can easily monitor their training, as well as physical parameters as heart rate, burnt calories, blood oxygen level, among others.

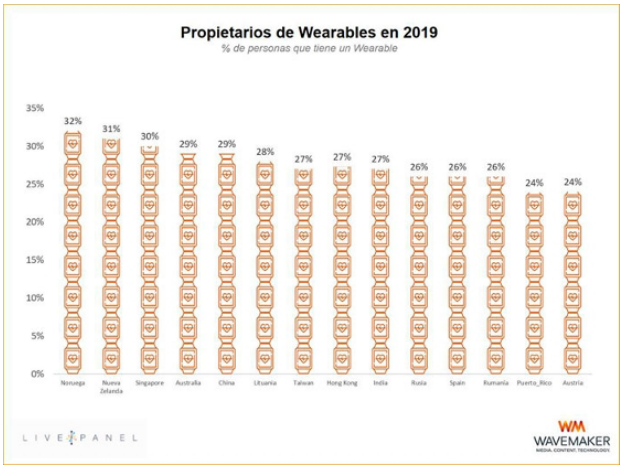

Figure 3.1: Wearable users by country, year 2019. Source: *Wavemaker* [10]

As it can be seen in figure 3.1 the penetration rate of wearables in world markets is around 30% by the year 2019 and it is expected to keep growing. This data points out how these devices have become part of the everyday life of a lot of people, but also suggests that there is a huge percentage of potential users that are not aware of this technology.

Due to the health situation derived from the SARS-CoV-2 virus (COVID-19), there has been a significant upswing in the value society places on health. On the one hand, organisations and companies have started to use wearables to fight the pandemic. On the other hand, the enacted lockdowns, as well as closures of gyms and other sports venues, have led people to use fitness apps or other online resources to keep fit. The popularisation of these services is driving the sale of smartwatches and smartbands focused on health and sport, as they allow users themselves to monitor workouts, anytime, anywhere. Moreover, in the vast majority of cases, it is the wearable company that offers a complementary application, so that the part corresponding to the services of the corporation is also boosted. Hence, it can be said then that wearables are a promising technology that is making its mark in the present but will undoubtedly be very important in the short and medium term.

To develop this project a development kit is used. It consists of a functional prototype of a wearable with includes several sensors to test and create applications or other devices. In the following sections, some alternatives are discussed.

3.1.1 MikroE Hexiwear

MikroElectronika's MIKROE-2026 or Hexiwear [16] is a smartwatch in the form of a development kit. It offers the possibility to create one's wearable device compatible with mobile applications. It consists of a 1.1 inch OLED display with 6 capacitive buttons. Micro USB-B connection for charging the battery and for developing the device's software. It weighs 40 grams. It contains sensors for heart rate, temperature, humidity, pressure, ambient light and orientation measurement.

Figure 3.2: MikroE Hexiwear with its watch strap. Source: *MikroElectronika*[16]

The embedded processor is an NXP-Kinetis K64 MCU, based on an ARM Cortex-M4. It features:

- Clock speed up to 120 MHz.

- 256 KB SRAM memory.
- 1 MB flash memory.

For communication, it incorporates an NXP-Kinetis KW4x processor, based on an ARM Cortex-M0+ and providing BLE (Bluetooth Low Energy) connectivity.

As for the sensors, it incorporates, on the one hand, the FXOS8700CQ, which integrates an accelerometer and magnetometer, and on the other, the FXAS21002, which provides gyroscope measurements. Both are provided by NXP.

- Accelerometer (FXOS8700CQ)
  - Sample frequency: up to 800 Hz
  - Range: $\pm 2g, \pm 4g, \pm 8g$
- Gyroscope (FXAS21002C)
  - Sample frequency: up to 800 Hz
  - Range: $\pm 250°/s, \pm 500°/s, \pm 1000°/s, \pm 2000°/s$
- Magnetometer (FXOS8700CQ)
  - Sample frequency: up to 800 Hz
  - Range: $\pm 12 gauss$

3.1.2 STEVAL-WESU1

Manufactured by STMicroelectronics[30] this device mounts three low power sensors that can measure pressure, acceleration, magnetic field and rotation. It is small and lightweight since its size is 30x35mm and weighs around 15g.

STEVAL-WESU1 allows connection with other devices through Bluetooth Low Energy (BLE). The development kit integrates a BLE processor, the BlueNRG-MS, which is also manufactured by STMicroelectronics [30].

Figure 3.3: STEVAL-WESU1 inside its package. Source:*ST Electronics*[30]

As the main processor, the development kit includes the STM32L151VEY6, a 32-bit ARM Cortex-M3-based CPU provided by the company. Its main characteristics are:

- Clock speed up to 32MHz
- 512 KB Flash memory
- 48 KB RAM

The three sensors are distributed between two modules (LSM6D3 and LIS3MDL) which are also developed of ST. The specifications are presented as

- Accelerometer (LSM6DS3)
  - Sample frequency: up to 1.6kHz
  - Range: $\pm 2g, \pm 4g, \pm 8g, \pm 16g$
- Gyroscope (LSM6DS3)
  - Sample frequency: up to 1.6kHz
  - Range: $\pm 125°/s, \pm 245°/s, \pm 500°/s, \pm 1000°/s$
- Magnetometer (LIS3MDL)
  - Sample frequency: up to 1kHz
  - Range: $\pm 4, \pm 8, \pm 12, \pm 16 gauss$

3.1.3 SensorTag CC2650

The SimpleLinkTM SensorTag CC2650 from Texas Instruments [33] contains ten low-power sensors that can measure various quantities such as acceleration, magnetism, rotation, pressure, temperature, humidity, etc. Its dimensions are 5x6.7x1.4cm.

Communication with other external devices can be carried out via Bluetooth Low Energy technology or via its derivative iBeacon. However, the device also allows communication via new technologies such as ZigBee or 6LoWPAN. This device allows its firmware to be modified employing a DevPack offered by the same manufacturer, as well as offering the possibility of incorporating other development kits to be able to introduce new sensors and actuators.

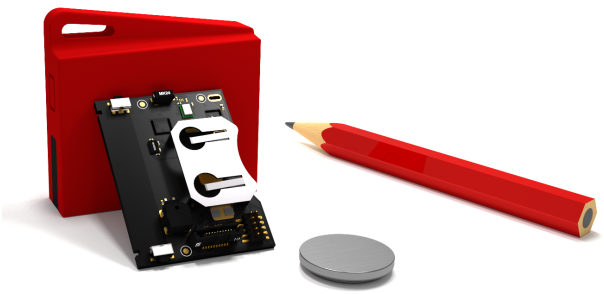

Figure 3.4: SensorTag CC2650 next to some objects to see its size. Source: *Texas Instruments*[33]

Similarly to the STEVAL-WESU1 it incorporates a 32-bit ARM Cortex-M3 processor. CC2650's specifications are:

- Clock speed up to 48MHz
- 128 KB Flash memory

The device contains a motion processing unit, namely the MPU-9250 module manufactured by InvenSense [24]. This unit can measure in 9 axes and contains an accelerometer, a gyroscope and a magnetometer with the following characteristics.

- Accelerometer (MPU-9250)
  - Sample frequency: up to 4 kHz
  - Range: $\pm 2g, \pm 4g, \pm 8g, \pm 16g$
- Gyroscope (MPU-9250)
  - Sample frequency: up to 1.6kHz
  - Range: $\pm 250°/s, \pm 500°/, \pm 1000°/s$
- Magnetometer (MPU-9250)
  - Sample frequency: up to 8 Hz
  - Range: $\pm 4.8 gauss$

3.2 Operating systems

As pointed out in 3.1, wearables often need a companion smartphone app to interact with the device. When comes to mobile development, choosing the operating system conditions all the process, since different environments need unique tools, languages and programming skills.

3.2.1 Android

Android is a mobile operating system with a kernel based on Linux designed for touchscreen devices such as smartphones, tablets, smartwatches, cars and TVs. It is currently the most popular on the market and is used by the vast majority of users (around to 73 % [23]). It was developed by Android Inc. which was later acquired by Google in 2005.

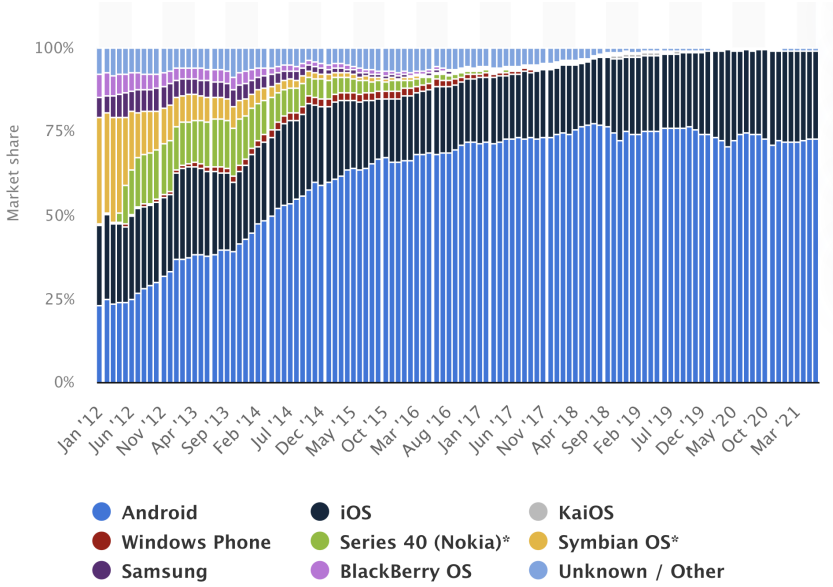

Figure 3.5: Mobile OS market share evolution from January 2012 to June 2021. Source: *Statista* [23].

Android is an open-source operating system, which is why there are many mobile devices of different sizes and resolutions providing this software. Fragmentation is one of its strengths and weaknesses at the same time since it helps to the popularity of the SO as well as makes it difficult for programmers to optimize the apps for every device. In addition, it offers different forms of messaging and supports many multimedia formats. Android also allows different connectivity technologies such as GSM/EDGE, NFC, Bluetooth, Wi-Fi, etc.

Another important feature is the catalogue of applications. Android uses Google Play which allows users to browse and download them. To upload applications, you have to register as a developer and pay a registration fee. Although it has many more features, finally it should be noted that the web browser included in Android is based on the open-source WebKit rendering engine that makes it possible to interact with a web server to retrieve and render web pages, download files, and manage plugins.

The main features of this operating system are the following:

- Free and open source development
- Kernel based on the Linux Kernel
- Large number of APIs available
- Use of SQLite for databases
- Allows Java, C or C++ programming
- Allows multitasking of applications
- It has a very extensive catalogue of applications (free and paid) through Google Play
- Support for a multitude of multimedia formats
- Support for HTML, HTML5 and Adobe Flash Player
- Google Assistant virtual assistant

- Includes a web browser (Google Chrome) based on Blink
- Offers different forms of messaging
- Has an official development environment: Android Studio

3.2.2 iOS

iOS is Apple Inc.'s proprietary mobile operating system for its mobile devices: iPhone, iPad and iPod touch. After Android, iOS is the second most popular mobile operating system in the world, with a market share of around 26.34% [23].

The characteristics of iOS are similar to those of Android, with the main difference being the adaptability of the system and the freedom of the code. In this case, the system does not adapt to different devices because it is not free; it is only used by Apple for its handsets. In the rest, it works similarly, with applications implemented by developers using Swift, the language created by the same company for this purpose, and put on the market through the App Store. It also has multimedia and streaming support, web browsing via HTML and a virtual assistant, Siri.

The main specifications of iOS are:

- Exclusive code for Apple devices.
- XNU-based core.
- Allows multitasking of applications.
- Extensive catalogue of applications (free and paid) through the App Store.
- Support for a multitude of media formats.
- HTML and HTML5 support.
- Siri virtual assistant.
- Includes a web browser (Safari) based on WebKit Offers different forms of messaging.

3.2.3 KaiOS

It is a Linux-based operating system that aims to bring the best of smartphones to affordable devices and the so-called *feature phones*. That is, it is a system designed for technologically accessible terminals, such as those used by the elderly. It is an open-source platform born out of the Firefox community.

Its main features are similar to those of the dominant systems on the market but aimed at non-touch phones, with a fully optimised interface and low power and memory requirements. It is capable of running on devices with only 256 MB of memory and a battery life of weeks. It brings applications and the use of cellular networks for Internet connectivity closer to the devices used in the past decade. Its market share is estimated at 0.35% [23].

Chapter 4

Description of the chosen solution

This chapter will present the chosen solution for the development of the project. Initially, a general block diagram will be shown, and then details of the hardware and software chosen are going to be discussed.

4.1 General block diagram

Figure 4.1: General block diagram of the project.

4.2 Hardware

4.2.1 Reasons for the choice

For the hardware part, MikroElectronika's Hexiwear was selected among the other development kits. Firstly, its design is perfect for the application, since the manufacturer has in its catalogue a strap that helps to wear it as a watch. Although it is kind of bulky, it does not interfere with the player's motion.

Secondly, the processing power of the CPU is way higher than the other competitors, as well as bigger storage and memory. This will help when acquiring data from the sensors because the quantity of information stored every second is going to be noticeable.

Thirdly, Hexiwear includes health sensors such as heart rate and calorie counter. In a sports context, these magnitudes are always interesting to measure. Biometrics can be included in more advanced stages of the project. But for the moment, only inertial magnitudes will be employed

Fourthly, this hardware option incorporates Bluetooth Low Energy (BLE) communication, which is considered the most optimal for the required application, as it does not require long-range communication but the lowest possible power consumption in data transmission.

Finally, the availability of the device in the UPV Electronics department has also been another fundamental aspect of the choice. Because of this, it has been possible to work with the device from the first moment, becoming familiar with its operation in a faster way.

4.2.2 Block diagram

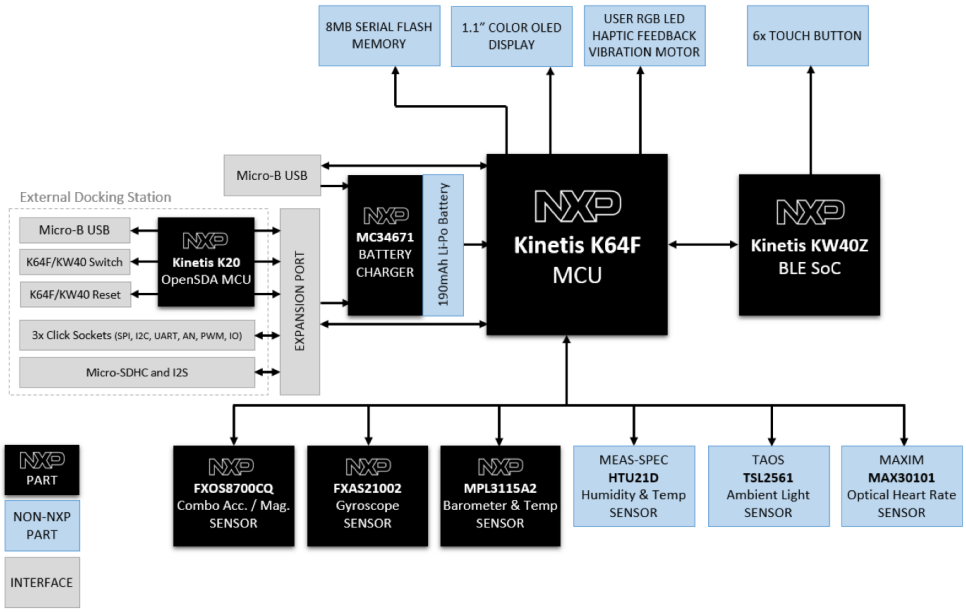

Figure 4.2: Hardware's block diagram. Source: *MikroElectronika/NXP*[16]

4.2.3 Technical specifications

- **MCU:** NXP Kinetis K64 MCU (ARM Cortex-M4, 120 MHz, 1M Flash, 256K SRAM)
- **BLE:** NXP Kinetis KW4x (ARM Cortex-M0+, Bluetooth Low Energy & 802.15.4 Wireless MCU)
- **3D Accelerometer and 3D Magnetometer:** NXP FXOS8700CQ
- **3-Axis Digital Gyroscope:** NXP FXAS21002
- **Absolute Digital Pressure sensor:** NXP MPL3115A2R1
- **600 mA Single-cell Li-Ion/Li-Polymer Battery Charger:** NXP MC34671

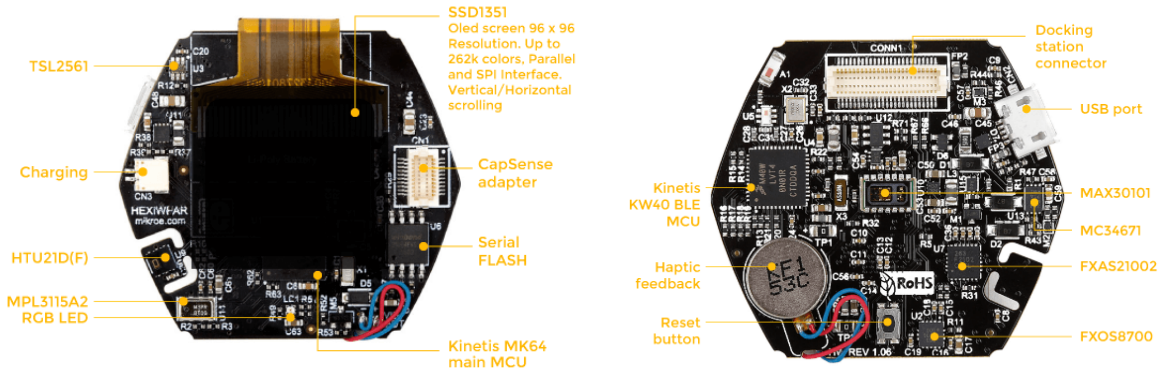

Figure 4.3: Left: Front view of Hexiwear's PCB. Right: Back view. Source: *MikroElectronika*[16]

- **Light-to-digital converter:** TAOS TSL2561
- **Digital humidity and temperature sensor:** MEAS HTU21D
- **Heart-rate sensor:** Maxim's MAX3010x
- **1.1" full color OLED display**
- **Haptic feedback engine**
- **190 mAh 2C Li-Po battery**
- **Capacitive touch interface**
- **RGB LED**
- **8 MB of additional Flash memory**

4.3 Software

4.3.1 Reasons for the choice

As pointed out in chapter 3, the mobile market is coped by Android and iOS, which own almost 99% of the share. Instead of choosing one of them, a multiplatform SDK such as Flutter was chosen. Flutter is Google's UI toolkit for making beautiful, natively compiled applications for mobile, web and desktop from a single code base [13]. That means that it is possible to code an app for iOS that will also run flawlessly into Android. Since mobile app coding was an unknown topic and it had to be learnt from the very beginning, Flutter was chosen to get the most versatility from this learning experience. It is also a popular developing environment, so there is plenty of packages that make coding easier. Furthermore, the available device for the testing was an iPhone 11, so creating the app exclusively for Android was discarded.

Figure 4.4: Flutter logo. Source: *Google Inc.*[13]

Concerning the development environment chosen, this has been Android Studio, the official one from Google. It is an IDE based on the IntelliJ IDEA software from JetBrains and is available free of charge. It should also be noted that this development environment is available for the main platforms (Microsoft Windows, macOS and GNU/Linux).

4.3.2 Block diagram

Figures 4.5 and 4.6 show the block diagrams of the software. The first one corresponds to the block diagram of the developed application that will run on the smartphone with the chosen operating system, while the second refers to the code implemented for the data analysis in MATLAB.

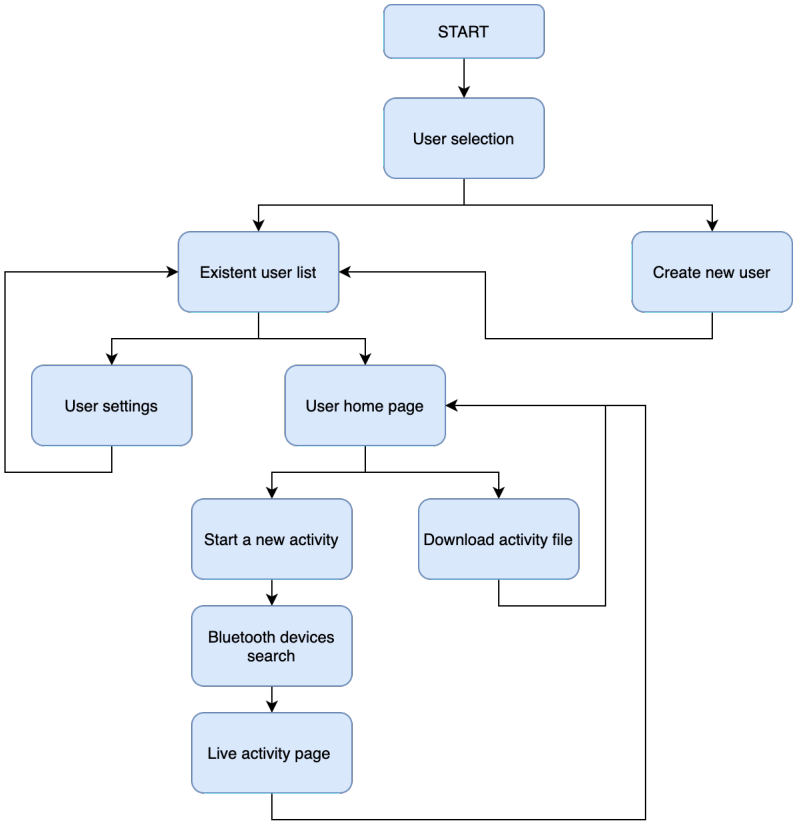

Figure 4.5: Softwate block diagram. General app diagram.

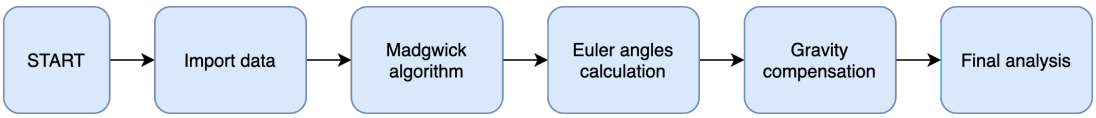

Figure 4.6: Software block diagram. MATLAB script.

4.3.3 Other software tools used

Other tools used during the process of developing this project, as well as the reason for their use, are described below.

- **Dart**: Dart is an open-source language developed at Google to allow developers to use an object-oriented language with static type analysis. It was created to compile Javascript more efficiently. Some reasons to choose Dart were:
  - Syntax is similar to C, Javascript and Java.
  - Flutter is based on it.
  - It allows Just-in-Time compilation, which allows applying the changes in the code immediately.
- **MATLAB:** This numeric computing environment program has been used for the processing, adjustment and analysis of the data collected by the sensors. It works with its multi-paradigm programming language and offers an integrated development environment. The reason for the choice is due to:
  - Fast execution and calculation accuracy.
  - It offers wide support of predefined functions.
  - It is compatible with a large number of file formats, such as .csv.
  - Previous knowledge of the software and its programming language.
- **Visual Studio Code:** It is a code editor developed by Microsoft that also allows code compilation and app debugging. Although Android Studio was the main IDE to develop the project, VS Code was also used due to:
  - Faster and smoother coding experience.
  - Lightweight and less resource-demanding environment.
  - Flutter and Dart support.
- **Xcode:** It is Apple's official IDE to develop applications and programs for its devices. It is required to make the build for iOS apps to test and debug them. It was used because:
  - It is mandatory for developing iOS apps
  - Allows deploying iOS apps in the iPhone without publishing them in the App Store.
- **Draw.io:** It has been used as a graphic design tool to make block diagrams and flowcharts. It was chosen for the following features:
  - It is an online, efficient and lightweight tool.
  - It includes a handful of graphic design resources.
- **LaTeX text editor:** This report has been written in LaTeX by means of TeXstudio editor for macOS. It was chosen based on:
  - It creates professional-looking documents which are the standard among the scientific community.

  - It allows to focus on the content rather than the formatting.
  - It provides consistency throughout the document.
  - Top-notch citation and bibliography management.

- **macOS Big Sur** The computer used for coding the app, analyze the sensors' information, make research and write the report used macOS Big Sur version 11.4. The reasons for the choice were:
  - It was the available device at the workplace.
  - iOS development requires Xcode, which is exclusive of Apples' Mac computers.

Chapter 5

Theoretical background

This chapter highlights the theoretical concepts necessary for the development and understanding of the project. Firstly, the communication protocol used is explained, and then key concepts about the orientation of objects in space are developed.

5.1 Bluetooth Low Energy (BLE)

The communication protocol used to connect the wearables to the smartphone is Bluetooth Low Energy. Its main advantage is the very low power consumption since it was developed to send small amounts of data to short distances. Depending on the use case it can consume a hundred times less than regular Bluetooth [9]. That unique feature makes the protocol perfect for wearable technology because these devices do not provide big batteries and thus any way of increasing its duration is highly appreciated.

BLE communication is based on a specification called GATT (General ATTribute profile), which defines how information known as attributes is transferred and received between two devices, where one acts as a server and the other as a client.

The GATT is built on top of the ATT attribute protocol (ATTribute protocol). ATT is optimised for execution on BLE devices. For this purpose, it uses as few bytes as possible. Each attribute is uniquely identified by a universally unique identifier (UUID), which is a standardised 128-bit format for uniquely identifying information. Attributes transferred by ATT are in the format of characteristics, services and descriptors.

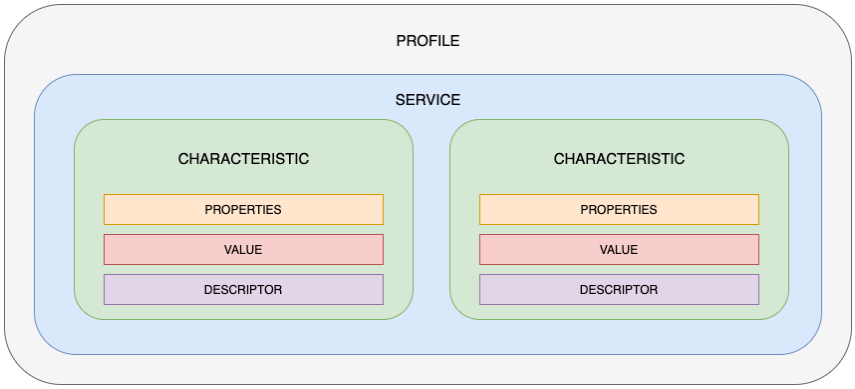

Figure 5.1: GATT profile structure. Source: *Own preparation based on* [9]

Figure 5.1 shows the structure of the GATT profile. Each profile can contain several services, which in turn group several characteristics. In addition, these features can contain a descriptor, values and properties. Each one of these elements is identified with its UUID.

5.2 Representation of the orientation

There are different systems for the representation of orientation such as rotation matrixes, Euler angles, quaternions... For the application in question, a three-dimensional representation is of interest to be able to know the exact position of a body in space. Of the different alternatives of representing orientation, quaternions and Euler angles have been chosen for the project and are explained below.

5.2.1 Euler angles

Euler angles are a set of three angular coordinates that specify the orientation of a moving orthogonal axis reference system relative to another fixed orthogonal axis reference system.

Given two coordinate systems xyz and XYZ with a common origin, the position of one system relative to the other can be determined using three angles (α, β and γ). The mathematical definition is based on choosing two planes, one in the reference frame and one in the rotated trihedron. In figure 5.2 they would correspond to the xy and XY planes, but choosing other planes would give different alternative conventions.

For the project, we will use the Euler angle representation with the ZYX convention, called navigation angles or Tait-Bryan angles, as they can be seen in figure 5.3. These are called ϕ (roll or bank), θ (pitch or attitude) and ψ (yaw or heading).

The intersection of the chosen xy and YZ coordinate planes is called the line of nodes (N), and is used to denote the three angles:

- ϕ corresponds to the angle between the line of nodes and the y-axis.
- θ is the angle between the xy-plane and the x-axis.
- ψ corresponds to the angle between the y-axis and the line of nodes.

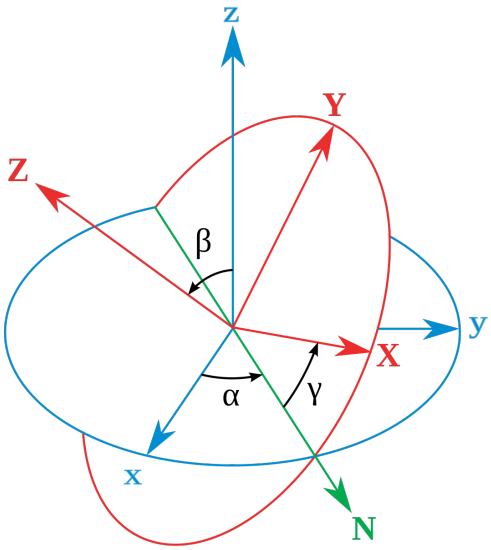

Figure 5.2: Euler angles.
Source: *Wikimedia Commons* [7]

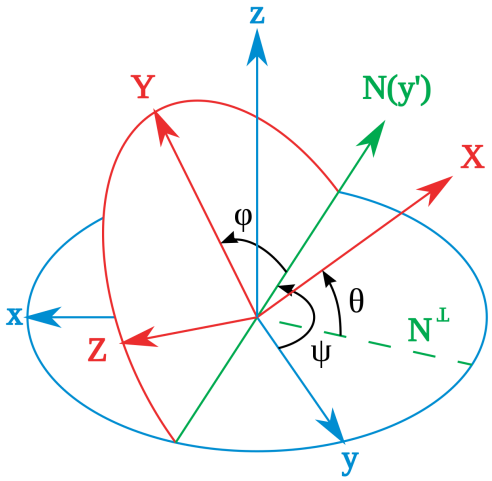

Figure 5.3: Tait-Brian ZYX convention.
Source: *Wikimedia Commons* [12]

However, it should be noted that the use of this representation system brings with it the problem of gimbal lock.

5.2.2 Gimbal lock

The gimbal lock is a phenomenon derived from the use of Euler angles. It consists of the loss of one degree of freedom when two of the three rotation axes are aligned. Hence, it exists a direction in which is not possible to instantly rotate.

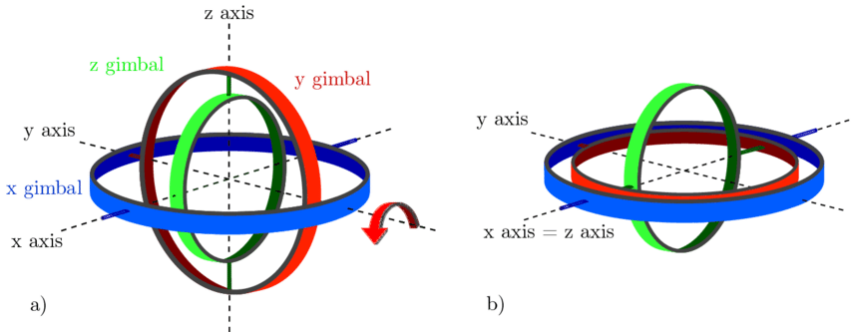

Figure 5.4: Gimbal lock representation. Source: *ResearchGate*[11]

There are different solutions to this problem, one possible solution is to rotate one or more axes to an arbitrary position at the moment the gimbal is locked, thus resetting the device. The most commonly used solution is to avoid the three-dimensional representation with Euler angles and use the quaternion representation.

5.2.3 Quaternions

Quaternions are an extension of real numbers with four dimensions. Their usefulness in this field is to provide a mathematical notation for representing the orientations and rotations of objects in three-dimensional space. They are simpler to compose than Euler Angles and avoid the Gimbal Lock problem, as they have four axes. Quaternions are represented as follows:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{5.1}$$

Where q_0 is the scalar part and $q_1 i$, $q_2 j$ and $q_3 k$ the vectorial complex part. i,j and k follow these properties:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{5.2}$$

Quaternions have the characteristic of not allowing the commutative property in multiplication. They are very useful in computer graphics and robotics applications, as well as in navigation. Nevertheless, its main drawback is the difficulty for representation, since quaternions have four dimensions. Madgwick [20] presents the necessary operations to obtain Euler angles from quaternions:

$$\phi = \arctan 2(2(q_2 q_3 - q_0 q_1), 2q_0^2 - 1 + 2q_3^2) \tag{5.3}$$

$$\theta = -\arctan(\frac{2(q_1 q_3 + q_0 q_2)}{\sqrt{1 - (2q_1 q_3 + 2q_0 q_2)^2}}) \tag{5.4}$$

$$\psi = \arctan 2(2(q_1 q_2 - q_0 q_3), 2q_0^2 - 1 + 2q_1^2) \tag{5.5}$$

5.3 Orientation measurement

To determine the orientation of a body, it is necessary to use certain sensors that, in combination, provide its orientation in space. Two mechanisms for obtaining orientation are highlighted below: Inertial Measurement Units (IMU) and Attitude and Heading Reference Systems (AHRS).

5.3.1 Inertial Measurement Units (IMU)

An inertial measurement unit or IMU is an electronic device that allows measurements of velocity, rotation and gravitational forces to be obtained using a combination of accelerometers, gyroscopes and magnetometers.

They are used as a fundamental component in the navigation systems of any mobile phone wherever these measurements are required, without the possibility of using external references or direct measurements.

One of the major disadvantages of using IMUs is that they are usually affected by a cumulative error. This is due to the operation of the system, which adds the detected changes to the

previously calculated positions. Therefore, any error in the measurement, however small, accumulates, leading to an increasing difference between the actual position and the position the system thinks it is in.

5.3.2 Attitude and heading reference system (AHRS)

An attitude and heading reference system (AHRS) is an electronic device that allows orientation to be obtained by providing information such as roll, pitch and yaw. They are also called MARG sensors and consist of three types of sensors: gyroscopes, accelerometers and magnetometers. The key difference between an IMU and an AHRS is that the latter also incorporates an onboard processing system, which provides attitude and heading information. An IMU only delivers sensor data to an additional device that calculates attitude and heading. Therefore, and thanks to the integration of a magnetometer, these systems can avoid the cumulative error of IMUs mentioned above. AHRS typically uses Kalman filters as a non-linear estimation method, intending to calculate a single solution from the different sources available to them.

5.3.3 Kalman filter

The Kalman filter is an algorithm developed by Rudolph E. Kalman in 1960 to track and predict linear systems. The algorithm uses least-squares estimation to process all available measurements, regardless of their precision, to estimate the current value of the variables of interest. This is possible due to:

- The knowledge of the system and the measurement devices.
- The statistical data of the measurement devices.
- The available information on the variables of interest.

The Kalman filter works on discrete measurements instead of working in continuous time. It is also a recursive algorithm, i.e. it does not require the storage of all previous data to be reprocessed when new samples are added. Figure 5.5 shows the general structure of a Kalman filter in a block diagram.

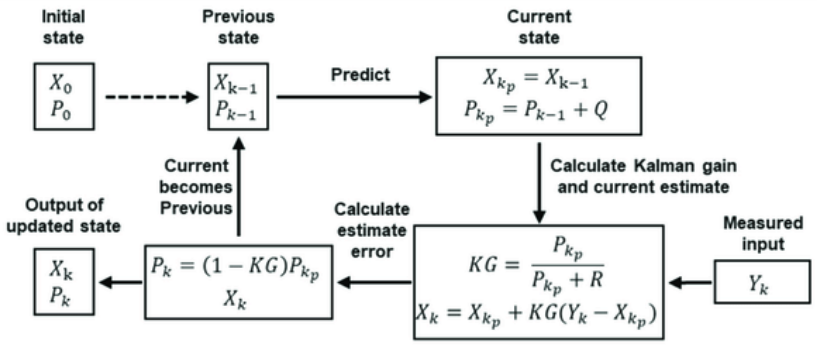

Figure 5.5: Kalman Filter block diagram structure. Source: *ResearchGate*[1]

It should be noted that the Kalman filter has been a tool that has revolutionised current technology, since it has solved numerous problems related to control and estimation, providing great

precision and efficiency. Its main applications include its use in modern control systems, as well as in the tracking and navigation of any type of vehicle.

5.3.4 Madgwick algorithm

The Madgwick algorithm is a filter implemented by Sebastian O.H. Madgwick based on the Kalman filter, applicable to both IMUs and MARGs, as detailed in his report [20]. This filter uses the quaternion as a method for orientation representation since, as we have seen, it avoids the singularities associated with Euler angle representation.

This algorithm in its MARG version (since it has a gyroscope, an accelerometer and a magnetometer, with information on each of its three axes) is the one we have decided to use for the development of this project since it has numerous advantages over other Kalman-derived filters. Some of the most important improvements are mentioned below:

- The low computational cost it requires concerning other filters of this calibre. It requires only 109 arithmetic operations per filter update in the case of IMUs and 277 in the case of MARGs.
- The adjustment of one (in IMUs) or two (in MARGs) parameters such as the filter gain β that allow obtaining more optimal results depending on the characteristics of the system.
- The implementation of algorithms to compensate for magnetic distortion and gyroscope drift.
- Good effectiveness at relatively low sampling times.

With Madgwick's algorithm, the aim is to obtain information about how the position of the Hexiwear device evolves during the free throw. With this, it will be possible to extract information and, after analysing it, detect the shot performed and other characteristics related to performance.

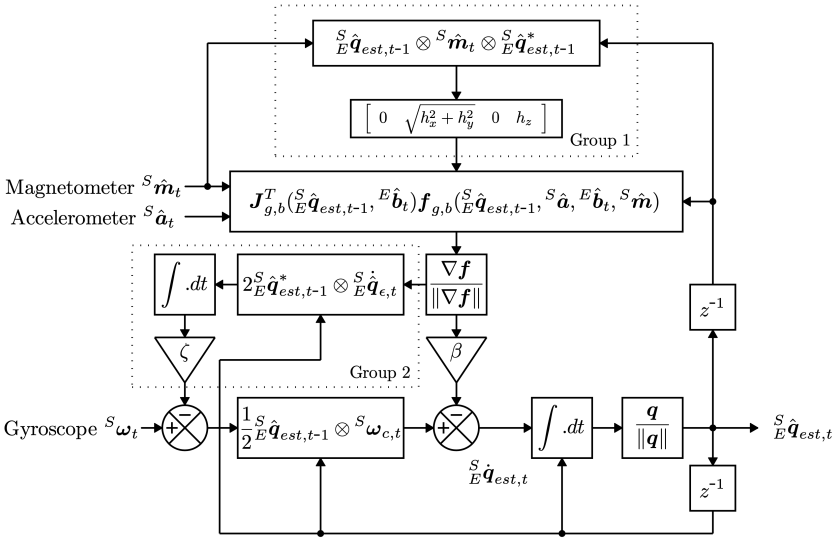

Figure 5.6: Madgwick's algorithm block diagram structure. Source: q*Madgwick* [20]

Chapter 6

Implementation details

In this chapter, the development of the prototype implementation will be explained in detail. First of all, the hardware highlights will be outlined, followed by an explanation of the various software parts.

6.1 Hardware

Regarding hardware, one of the main goals was not to alter Hexiwear's firmware. To do so, the sensors have been configured via the developed app.

The project requires two sensors, one on the shooting wrist and the other placed on the net. To make the process easier, it was decided to use another Hexiwear. It is known that this solution is overdimensioned since it includes way more functionalities and sensors that will not be used in the project. Only the accelerometer data is going to be recorded from this secondary device.

6.2 Software

This section shows the different parts of the developed software. The block diagrams seen in section 4.3.2 will be used to better understand the location of each part within the mobile application and the MATLAB script, respectively.

6.2.1 Searching, selecting and connecting BLE devices

To be able to communicate with the wearables, an app snippet written by Lee Lup Yuen [36] and based on the `flutter blue` package [14] from Paul de Marco was used as a blueprint to create the module of the application that allows the user to access the data and pair the devices. This section corresponds to the part of the block diagram shown in figure 6.1. In section 7.1.3 the screenshots of the app are shown. In the annex, the full code is displayed (9.2.6).

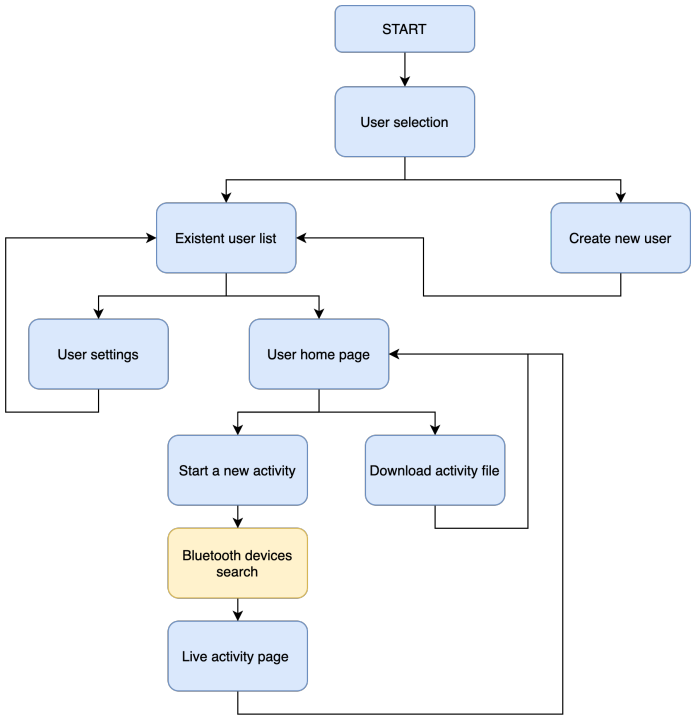

Figure 6.1: Software block diagram. Bluetooth devices search.

Before implementing the code, Bluetooth and location permissions are required. To obtain them, instructions must be written in the `AndroidManifest.xml` file (Android) or `Info.plist` file (iOS). Since the app works for both platforms, both codes are inserted.

```
<!--AndroidManifest.xml-->
<uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
 <application
```

```
<!--Info.plist-->
  <dict>
      <key>NSBluetoothAlwaysUsageDescription</key>
      <string>Need BLE permission</string>
      <key>NSBluetoothPeripheralUsageDescription</key>
      <string>Need BLE permission</string>
      <key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
      <string>Need Location permission</string>
      <key>NSLocationAlwaysUsageDescription</key>
      <string>Need Location permission</string>
      <key>NSLocationWhenInUseUsageDescription</key>
      <string>Need Location permission</string>
```

6.2.2 Data collection and adjustments

Data collection

Looking at Hexiwear's user manual, the manufacturer indicates that the motion service where the accelerometer, gyroscope and magnetometer data can be consulted corresponds to the 0x2000. Figure 6.2 gives also information about the data format and the units.

UUID	Characteristic	Format	Security Mode	R/W permissions	Details
0x2001	Accelerometer	int16_t[3]	Encryption with authentication	Read	Accel measurement for x, y, z coordinate. Range: +/- 4g
0x2002	Gyro	int16_t[3]	Encryption with authentication	Read	Gyro measurement for x, y, z coordinate. Range: +/- 256 deg/sed.
0x2003	Magnetometer	int16_t[3]	Encryption with authentication	Read	Magnet measurement for x, y, z coordinate.

Figure 6.2: Hexiwear's motion service 0x2000. Source: *MikroElectronika* [16]

Data collection methods belong to `live-activity.dart` and the code is fully displayed on the annex, in 9.2.7. To obtain the data from the different channels of the sensors, the app asks the wearable for the values of the characteristics in the service 0x2000.

```
            //display sensor info
            StreamBuilder<List<BluetoothService>>(
              stream: deviceList.first.services,
              initialData: [],
              builder: (c, snapshot) {
                if (snapshot.data != null &&
                    snapshot.data!.isNotEmpty &&
                    snapshot.data![3].characteristics.isNotEmpty) {
                  // set characteristics
                  accelerometerCharacteristic1 =
                      snapshot.data![3].characteristics[0];

                  gyroscopeCharacteristic1 =
                      snapshot.data![3].characteristics[1];

                  magnetometerCharacteristic1 =
                      snapshot.data![3].characteristics[2];
```

Once the app access the characteristic, it is time to store the values inside the database. Snapshots of each channel are taken each 1/60 s, 17 ms approximately.

```

    //loop that will fetch the sensor info every 60Hz
    _timer = Timer.periodic(
      const Duration(milliseconds: 17),
      (timer) {
        if (

            // check device 1
            accelerometerCharacteristic1 != null &&
```

```
            gyroscopeCharacteristic1 != null &&
            magnetometerCharacteristic1 != null

            // check device 2

            &&
            accelerometerCharacteristic2 != null &&
            gyroscopeCharacteristic2 != null &&
            magnetometerCharacteristic2 != null) {
        // read characteristics from device 1

        accelerometerCharacteristic1!.read();
        gyroscopeCharacteristic1!.read();
        magnetometerCharacteristic1!.read();

        // read characteristics from device 2

        accelerometerCharacteristic2!.read();
        gyroscopeCharacteristic2!.read();
        magnetometerCharacteristic2!.read();

        _hiveDB.updateActivityWithMovement(
          name: widget.nameOfUser,
          movement: Movement()
            ..timestamp = DateTime.now()
            ..accX = accelerometerValues1[0]
            ..accY = accelerometerValues1[1]
            ..accZ = accelerometerValues1[2]
            ..gyrX = gyroscopeValues1[0]
            ..gyrY = gyroscopeValues1[1]
            ..gyrZ = gyroscopeValues1[2]
            ..magX = magnetometerValues1[0]
            ..magY = magnetometerValues1[1]
            ..magZ = magnetometerValues1[2]
            ..accX2 = accelerometerValues2[0]
            ..accY2 = accelerometerValues2[1]
            ..accZ2 = accelerometerValues2[2]
            ..gyrX2 = gyroscopeValues2[0]
            ..gyrY2 = gyroscopeValues2[1]
            ..gyrZ2 = gyroscopeValues2[2]
            ..magX2 = magnetometerValues2[0]
            ..magY2 = magnetometerValues2[1]
            ..magZ2 = magnetometerValues2[2],
        );
```

Sample frecuency justification

Sample frequency is an important parameter that will affect the process since it will determine the number of samples taken per second as well as the bandwidth. 100 Hz was the first option to be chosen because the papers taken as reference [28] [8] [19] used that frequency, but they do not provide a proper justification. Hence, the CSV file created with 100 Hz showed that there was a significant amount of data that was repeated and did not offer higher insight. Thus, there was room to try lower sampling frequencies without losing accuracy.

In his report [4] Barranco found that the energy of the signal provided by the wrist motion is focused in the frequency spectrum between 0 and 7 Hz. Applying the Nyquist theorem it can be determined that the minimum sampling frequency must be 14 Hz. Despite that, 14 Hz seemed

not enough since basketball is a fast sport in which small motions can make the difference and the point of the project is to be able to capture these nuances. Hence, it was decided to double the minimum value to obtain a 30 Hz maximum bandwidth. That value implies a sampling frequency of 60 Hz, which is a good compromise between the minimum value (14 Hz) and the one used by other researchers (100 Hz) obtaining enough data to characterize fast motions without wasting computing power.

Adjustments

The output data of the sensor is displayed as a 6 number array like this: `[61, 0, 215, 255, 190, 255]`. Each pair of data corresponds to one axis. The information is coded in binary using 2's complement, so to access the decimal values an adjustment through code is required.

```
1
2  //function to convert the sensor readings to decimal values
3  List convertFromBinary(List<int> hexList) {
4    print('here is the original list' + hexList.toString());
5
6    //inverts the order of each pair of data to display values properly
7    int firstValue = (hexList[0] + (hexList[1] * 256));
8    int secondValue = (hexList[2] + (hexList[3] * 256));
9    int thirdValue = (hexList[4] + (hexList[5] * 256));
10
11   firstValue = readSignedInt(firstValue);
12   secondValue = readSignedInt(secondValue);
13   thirdValue = readSignedInt(thirdValue);
14
15   return [
16     (firstValue),
17     (secondValue),
18     (thirdValue),
19   ];
20 }
21
22 //function to read the signed ints coded in 2's complement
23 int readSignedInt(m) {
24   int value = m;
25
26   //checks if this is a negative number
27   if ((value & 0x8000) > 0) {
28
29     value = value | 0xFFFFFFFFFFFF0000;
30   }
31
32   return value;
33 }
```

Database: Hive

Although SQLite is the most common database when developing mobile apps, the Hive package was used in this project. Hive is a simple, fast and lightweight database written natively in Dart [17], in which "boxes" are created to store the information. The syntax is way easier than using SQL and it has a better performance. On the page of the package, the author performed a benchmark against several database alternatives and Hive was by far the fastest as it can be seen in figures 6.3 and 6.4. As pointed out before, data from the sensors need to be captured each 10ms, so quickness and efficiency are required.

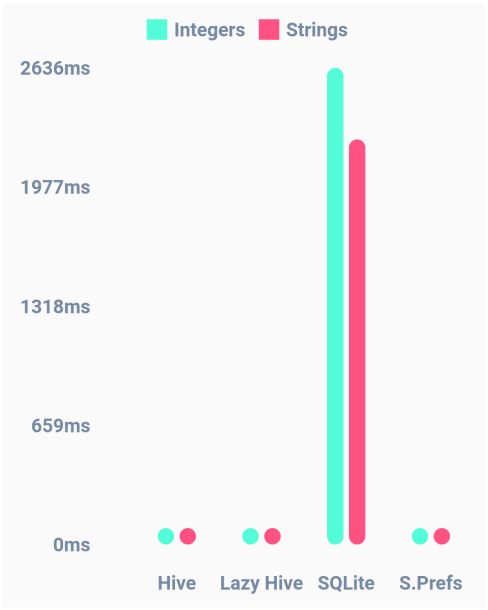

Figure 6.3: Benchmark results for reading 1000 iterations. Source: *Simon Eiler* [17]

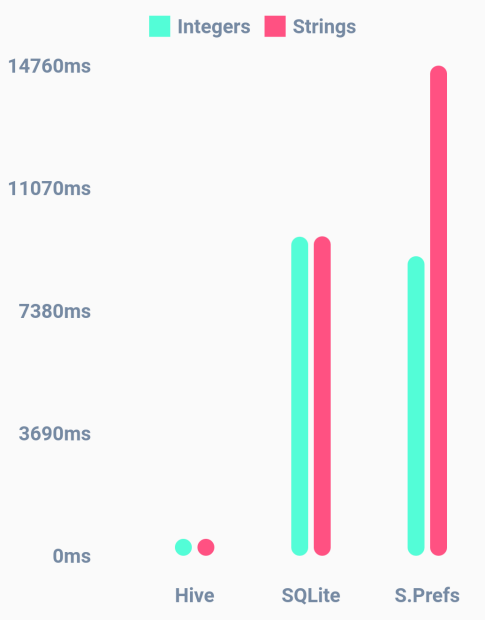

Figure 6.4: Benchmark results for writing 1000 iterations. Source: *Simon Eiler* [17]

For the development of the application, three boxes have been created within the database, as shown in figure 6.5. The first, under the name "user", contains the different users and their corresponding information. The second, "activities", contains the characteristics of the activities, and the third, "movement", contains the adjusted data from the sensors.

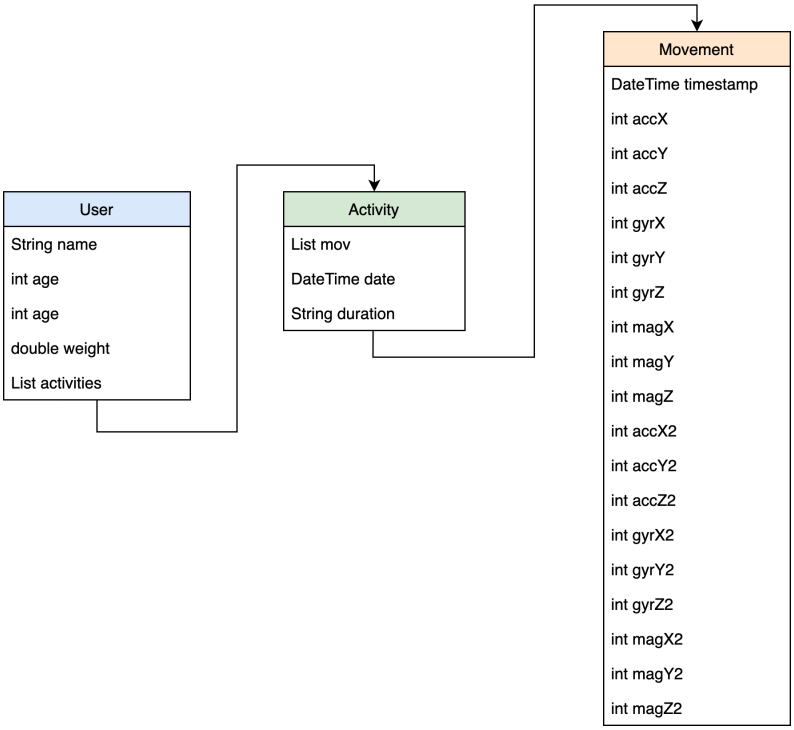

Figure 6.5: Relation between the created boxes for the database.

The "movement" box correspond to each data collected by the 3-D wearable sensors in an instant of time while the "activity" contains the general information of the shot session. Since the sampling frequency has been adjusted to 60Hz, it will get sixty snapshots from the sensors per second. In other words, a shot performed in 6 seconds would correspond to 1 activity and approximately 360 movements. To be able to relate the different movements to the activities, an identifier has been created, which corresponds to the parameter "mov". Continuing with the example, the 360 movements would have the same identifier as the activity to which they correspond.

It is important to mention that a different class has been created for each table, in addition to the fact that the names of the columns of each table have been defined through variables in the files `user.dart`, `activity.dart` and `movement.dart` to facilitate communication in the development of the application. Automatically, auxiliar files called `user.g.dart`, `activity.g.dart` and `movement.g.dart` are generated by the database.

Data export

To export the data, the app opens the `user` box and converts the date from that box to a CSV file. As it is stored in the local memory, the code asks for the permissions to do so. If they are granted, the CSV will appear in the app's folder inside the smartphone local directory. Full code is available in 9.2.5.

```
1              ValueListenableBuilder<Box<User>>(
2                valueListenable: Hive.box<User>(widget.name).listenable(),
3                builder: (_, box, __) => box.getAt(0)!.activities.isNotEmpty
4                    ? ListView.builder(
5                        physics: const BouncingScrollPhysics(),
```

```
                        shrinkWrap: true,
                        itemCount: box.getAt(0)!.activities.length,
                        itemBuilder: (ctx, index) {
                          return TextButton(
                            onPressed: () async {
                              //when we press the button it creates the csv
                              List<List<dynamic>> data = [
                                [
                                  "Time_Stamp",
                                  "Acc_X",
                                  "Acc_Y",
                                  "Acc_Z",
                                  "Gyr_X",
                                  "Gyr_Y",
                                  "Gyr_Z",
                                  "Mag_X",
                                  "Mag_Y",
                                  "Mag_Z",
                                  "Acc_2_X",
                                  "Acc_2_Y",
                                  "Acc_2_Z",
                                  "Gyr_2_X",
                                  "Gyr_2_Y",
                                  "Gyr_2_Z",
                                  "Mag_2_X",
                                  "Mag_2_Y",
                                  "Mag_2_Z",
                                ],
                              ];

                              for (var movement
                                  in box.getAt(0)!.activities[index].mov) {
                                // print('got a movement object ${movement}');
                                data.add(
                                  [
                                    movement.timestamp.millisecond,
                                    movement.accX,
                                    movement.accY,
                                    movement.accZ,
                                    movement.gyrX,
                                    movement.gyrY,
                                    movement.gyrZ,
                                    movement.magX,
                                    movement.magY,
                                    movement.magZ,
                                    movement.accX2,
                                    movement.accY2,
                                    movement.accZ2,
                                    movement.gyrX2,
                                    movement.gyrY2,
                                    movement.gyrZ2,
                                    movement.magX2,
                                    movement.magY2,
                                    movement.magZ2,
                                  ],
                                );
                              }

                              //this part works
                              String csvData =
                                  const ListToCsvConverter().convert(data);
```

```
67
68                              //this line prints all the data from the csv to the
    terminal, it works
69                              // debugPrint(csvData.toString(), wrapWidth: 1024);
70
71                              var _isGranted =
72                                  await Permission.storage.request().isGranted;
73                              print(_isGranted);
74
75                              if (await Permission.storage
76                                  .request()
77                                  .isGranted) {
78                                final String directory = Platform.isIOS
79                                    ? (await getApplicationDocumentsDirectory())
80                                        .path
81                                    : (await getExternalStorageDirectory())!
82                                        .path;
83
84                                final path =
85                                    "$directory/csv-${DateTime.now()}.csv";
86                                final File file = File(path);
87                                await file.writeAsString(csvData, flush: true);
88
89                                print(
90                                    'completed save successfully and saved file to '
91                                    "$directory/csv-${DateTime.now()}.csv");
92                              } else {
93                                await Permission.storage.request();
94                              }
95                            },
```

6.2.3 Data treatment

This section is comprised of the part of the block diagram shown in figure 6.6. It corresponds to the code implemented in MATLAB for data processing. It first includes an import of the data from the CSV file. Then the Madgwick algorithm is applied, obtaining the orientation in the form of quaternions. In addition, the Tait-Bryan angle orientation has also been obtained in the data processing. Finally, gravity compensation has been implemented in the acceleration measurement to compensate for its effect in the measures. All these implementations are detailed below.

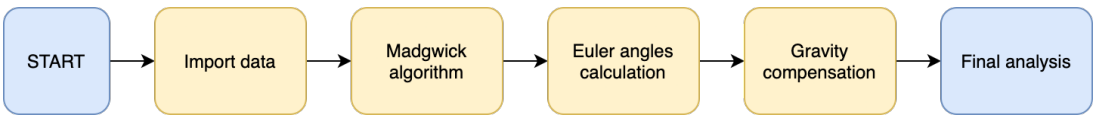

Figure 6.6: Software block diagram. Data treatment.

Data import

For the import of data, the layout of the data in the CSV file has to be taken into account. Figure 6.7 shows an extract of one of the exported data files inside MATLAB. The columns corresponding to the wrist sensor are highlighted in yellow to differentiate them easily.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	prueba11																		
	Time_...	Acc_X	Acc_Y	Acc_Z	Gyr_X	Gyr_Y	Gyr_Z	Mag_X	Mag_Y	Mag_Z	Acc_2_X	Acc_2_Y	Acc_2_Z	Gyr_2_X	Gyr_2_Y	Gyr_2_Z	Mag_2_X	Mag_2_Y	Mag_2_Z
	Num...	Nu...	Nu...	Num...	Nu...	Nu...	Nu...	Num...	Num...	Nu...	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	Time...	Acc_X	Acc_Y	Acc_Z	Gyr_X	Gyr_Y	Gyr_Z	Mag_X	Mag_Y	Mag_Z	Acc_2_X	Acc_2_Y	Acc_2_Z	Gyr_2_X	Gyr_2_Y	Gyr_2_Z	Mag_2_X	Mag_2_Y	Mag_2_Z
2	266	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	274	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	295	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	306	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	314	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	324	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	334	5	80	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	345	5	80	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	354	5	80	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	366	5	80	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	375	5	80	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	384	5	80	58	0	0	0	0	0	0	-75	-18	-56	0	0	0	0	0	0
15	395	5	80	58	0	0	0	0	0	0	-75	-18	-56	0	0	0	0	0	0
16	404	5	80	58	0	0	0	0	0	0	-75	-18	-56	0	0	0	0	0	0
17	414	5	80	58	0	0	0	0	0	0	-75	-18	-56	0	0	0	0	0	0
18	423	5	80	58	0	0	0	0	0	0	-75	-18	-56	0	0	0	0	0	0
19	434	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
20	446	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
21	454	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
22	464	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
23	474	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
24	484	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	0	0	0
25	504	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	-40	60176	7760
26	504	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	-40	60176	7760
27	513	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	-40	60176	7760
28	524	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	-40	60176	7760
29	534	5	80	58	0	0	0	0	0	0	-75	-18	-56	4	-1	0	-40	60176	7760

Figure 6.7: Example of CSV file once it has been exported to MATLAB.

Once in MATLAB, the script must be able to scan the columns storing in the different variables the corresponding data to perform the later analysis. The code is the following:

```
%% DATA TREATMENT
%imports and represents the sensor data
addpath('quaternion_library');     %includes library
filename1 = 'tirodentro2.csv';

M = csvread(filename1,2,0);%reads data from the .csv
time = M(:,1);

 for i=1:size(M,1)
     time(i)= t;
     t=t+(1/60);
 end

%assigns the values from the csv to variables to work with
Accelerometer = M(:, [2:4]);
Gyroscope     = M(:, [5:7]);
Magnetometer  = M(:, [8:10]);
Accelerometer2= M(:, [11:13]);
```

Madgwick algorithm

Once the data are available, the two adjustable parameters of the filter (the gain β and the sampling period or frequency) must be defined for the implementation of the Madgwick algorithm. The sampling period sets the algorithm's update period, which must coincide with that indicated on the sensors, i.e. 1/60 s (60Hz).

Concerning the gain, this refers to the measurement errors of the gyro in units of the derivative of the quaternion. This parameter includes all possible errors such as noise, calibration errors and quantization errors, among many others. From this value, the convergence ratio for the elimination of these errors is established, so the higher the β, the shorter the convergence time but also the greater the error. The author of this algorithm recommends in his report [20] a value of $\beta = 0.033$ for the IMU version, and $\beta = 0.041$ for the MARG implementation, as optimal values to achieve good performance. Therefore, as the MARG version has been used in the project, the corresponding recommended value has been taken.

For the implementation, the last parameter has also been defined which corresponds to the orientation quaternion. This variable has been initialised to (1,0,0,0,0) as indicated by the author in his report. Firstly, the code implemented in the script created to call the class that implements the algorithm can be seen below. Secondly, you can see the code extract containing the definition of this class created `MadgwickAHRS.m`. It should be noted that from these parameters the complete algorithm is implemented, which has been extracted from the repository [25].

```
%processes sensor data through the algorithm
AHRS = MadgwickAHRS('SamplePeriod', 1/60, 'Beta', 0.041);

%initializes quaternion matrix to value 0
quaternion = zeros(length(time), 4);

%obtains the quaternions using Madgwick's algorithm
%/100 because the sensor gives values times 100
% gyroscope units must be radians
for t = 1:length(time)
    AHRS.Update((Gyroscope(t,:)*(pi/180)/100), Accelerometer(t,:)/100, Magnetometer(t,:)
    /100);
    quaternion(t, :) = AHRS.Quaternion;
end
```

```
classdef MadgwickAHRS < handle
%MADGWICKAHRS Implementation of Madgwick's IMU and AHRS algorithms

    %% Public properties
    properties (Access = public)
        SamplePeriod = 1/60; %11/100
        Quaternion = [1 0 0 0]; %output quat describing Earth relative to the sensor
        Beta = 0.041;           %algorithm gain
    end

    %% Public methods
    methods (Access = public)
        function obj = MadgwickAHRS(varargin)
            for i = 1:2:nargin
                if strcmp(varargin{i}, 'SamplePeriod'),obj.SamplePeriod = varargin{i+1};
                elseif strcmp(varargin{i},'Quaternion'),obj.Quaternion = varargin{i+1};
                elseif strcmp(varargin{i},'Beta'),obj.Beta = varargin{i+1};
                else error('Invalid argument');
                end
```

```
            end;
        end
```

Euler angles

For the subsequent analysis, due to the great complexity of the quaternions in their 4-D representation, it has been decided to also obtain the Euler angles. For this purpose, the Euler angles (in degrees) have been obtained with the ZYX convention (Tait-Bryan angles) from the quaternions. This method has been extracted from the report [25] and the implementation proposed on it. The script has also been modified to include the gimbal lock solution mentioned in section 5.2.2.

```
function euler = quatern2euler(q)
%QUATERN2EULER Converts a quaternion orientation to ZYX Euler angles
%
%   q = quatern2euler(q)
%
%   Converts a quaternion orientation to ZYX Euler angles where
%   phi is a rotation around X,
%   theta around Y and
%   psi around Z.

    R(1,1,:) = 2.*q(:,1).^2-1+2.*q(:,2).^2;
    R(2,1,:) = 2.*(q(:,2).*q(:,3)-q(:,1).*q(:,4));
    R(3,1,:) = 2.*(q(:,2).*q(:,4)+q(:,1).*q(:,3));
    R(3,2,:) = 2.*(q(:,3).*q(:,4)-q(:,1).*q(:,2));
    R(3,3,:) = 2.*q(:,1).^2-1+2.*q(:,4).^2;

    phi = atan2(R(3,2,:), R(3,3,:) )* (180/pi);
    theta = -atan(R(3,1,:) ./ sqrt(1-R(3,1,:).^2) )* (180/pi);
    psi = atan2(R(2,1,:), R(1,1,:) )* (180/pi);

    euler = [phi(1,:)' theta(1,:)' psi(1,:)'];

    if((theta > 86) & (theta < 94))

        theta = pi/2* (180/pi);
        phi = 0* (180/pi);
        psi =  2 * atan2(q(:,2),q(:,1))* (180/pi);
    end

     if((theta > -94) & (theta < -86))
        theta = -pi/2* (180/pi);
        phi = 0* (180/pi);
        psi =  -2 * atan2(q(:,2),q(:,1))* (180/pi);
     end

     euler = [phi(1,:)' theta(1,:)' psi(1,:)'];

end
```

Below is the code included in the script for the use of this function, as well as its graphical representation.

```
% use conjugate for sensor frame relative to Earth and convert to degrees.
euler = quatern2euler(quaternConj(quaternion));

figure('Name', 'Euler Angles 1');
hold on;
plot(time, euler(:,1), 'r');
```

```
plot(time, euler(:,2), 'g');
plot(time, euler(:,3), 'b');
title('Euler Angles');
xlabel('Time (s)');
ylabel('Angle (degrees)');
legend('Roll (\phi)', 'Pitch (\theta)', 'Yaw (\psi)');
hold off;
```

Gravity compensation

The last adjustment of the data is the gravity compensation. This adjustment is necessary to obtain the coordinates of the real acceleration, since otherwise, it is under the effect of gravity, as explained by Verasano [35] in his article. For this purpose, the method proposed by the author has been implemented in MATLAB language.

```
function [ACC] = compensateAcc(q,acc)
%function to adjust the accelerometer values taking into account the gravity

g = [0,0,0];
        %Get expected direction of gravity
       g(1) = 2 * (q(2) * q(4) - q(1) * q(3));
       g(2) = 2 * (q(1) * q(2) + q(3) * q(4));
       g(3) = q(1) * q(1) - q(2) * q(2) - q(3) * q(3) + q(4) * q(4);

%compensates accelerometer readings with the expected direction of gravity
         ACC = [acc(1) - g(1), acc(2) - g(2), acc(3) - g(3)]
end
```

The following is the code implemented in the script for the use of this function.

```
acc = [];

for j=1:1:(size(Accelerometer,1))
    acc = compensateAcc(quaternion, Accelerometer(j,:));
    ACC(j,1) = acc(1);
    ACC(j,2) = acc(2);
    ACC(j,3) = acc(3);
end
```

6.2.4 Data analysis

This section comprises the part of the block diagram shown in figure 6.8. It corresponds to the code implemented in MATLAB for the validation of the data analysis methodology. After having obtained data from different shots, the validation of a methodology to analyse them is going to be carried out, to identify if they are free throws, their outcome and the score they deserve taking into account their technique.

Firstly, to identify the parameters that can be used to classify the movements, the different magnitudes of each of the recorded activities have been represented. For this purpose, the following lines of code have been implemented in the script, together with the representation of the orientation in Euler angles explained in section 6.2.3.

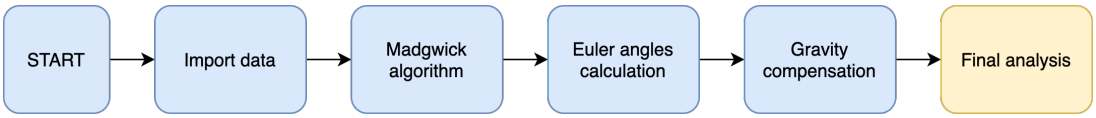

Figure 6.8: Software block diagram. Final analysis.

```
%plots the output of the different sensors individually
figure('Name', 'Sensor data (compensated)');

%subplot1 Acceleration (wrist)
axis(1) = subplot(5,1,1);
hold on;
plot(time, Accelerometer(:,1), 'r');
plot(time, Accelerometer(:,2), 'g');
plot(time, Accelerometer(:,3), 'b');
legend('X', 'Y', 'Z');
xlabel('Time (s)');
ylabel('Acceleration (g)');
title('Accelerometer');
hold off;

%subplot2 Acceleration (wrist) compensated
axis(2) = subplot(5,1,2);
hold on;
plot(time, ACC(:,1), 'r');
plot(time, ACC(:,2), 'g');
plot(time, ACC(:,3), 'b');
legend('X', 'Y', 'Z');
xlabel('Time (s)');
ylabel('Acceleration (g)');
title('Accelerometer (Compensated)');
hold off;

%subplot 3 Gyroscope (wrist)
axis(3) = subplot(5,1,3);
hold on;
plot(time, Gyroscope(:,1), 'r');
plot(time, Gyroscope(:,2), 'g');
plot(time, Gyroscope(:,3), 'b');
legend('X', 'Y', 'Z');
xlabel('Time (s)');
ylabel('Angular velocity (rad/s)');
title('Gyroscope');
hold off;

%subplot4 Magnetometer (wrist)
axis(4) = subplot(5,1,4);
hold on;
plot(time, Magnetometer(:,1), 'r');
plot(time, Magnetometer(:,2), 'g');
plot(time, Magnetometer(:,3), 'b');
legend('X', 'Y', 'Z');
xlabel('Time (s)');
ylabel('Magnetic Field (G)');
title('Magnetometer');
hold off;

%subplot5 Accelerometer (Net Sensor)
axis(5) = subplot(5,1,5);
hold on;
plot(time, Accelerometer2(:,1), 'r');
plot(time, Accelerometer2(:,2), 'g');
plot(time, Accelerometer2(:,3), 'b');
legend('X', 'Y', 'Z');
xlabel('Tiempo (s)');
```

```
ylabel('Acceleration (g)');
title('Net Sensor Accelerometer');
hold off;

linkaxes(axis, 'x');
```

After the representation of all variables, one variable has been chosen to analyse its variation with all activities. This elected variable is the roll or ϕ, which represents the X-axis of the object orientation. This election will be further discussed in section 7.2.1

Next, a model of one of the performed activities and another activity has been chosen to compare their similarity. Their corresponding ϕ values have been plotted in degrees versus time and cross-correlation between the two vectors has been applied using the MATLAB function `xcorr`. This function returns two parameters. The first one corresponds to the result vector of the cross-correlation between two sequences, while the second one, after applying a small operation, indicates the number of samples that the first vector has to be shifted to achieve maximum correlation. Knowing this, both vectors can be aligned in time to apply a second `corr2` function that returns the correlation coefficient between the two. However, to be able to apply this function, both vectors must have the same length, so we proceed first to perform this step as shown below.

```
%obtains the cross correlation between both vectors
[acor,lag] = xcorr(s1,s2);
[~,I] = max(abs(acor));

%obtains the number of samples to move for aligning the graphs
lagDiff = lag(I)

%obtains the time in seconds to move for aligning the graphs
timeDiff = lagDiff/60

%obtains the absolute value of samples to move for aligning
 if(lagDiff>0)
    lagDiff=-lagDiff
 end


if(timeDiff>0) %if the model has to be moved
    s1a = s1(-lagDiff+1:end);
    t1a = (0:length(s1a)-1)/60;

    %represents the aligned activity and model
    figure
    subplot(2,1,1)
    plot(t1a,s1a)
    title('Activity (aligned)')
    ylabel('Euler Angles (degrees)')

    subplot(2,1,2)
    plot(t2',s2)
    title('Free throw model')
    xlabel('Time (s)')
    ylabel('Euler Angles (degrees)')
    linkaxes

    s1=s1a

else
```

```
38
39      %if the activity has been moved
40      s2a= s2(-lagDiff+1:end);
41      t2a = (0:length(s2a)-1)/60;
42
43      %represents the aligned activity and model
44      figure
45      subplot(2,1,1)
46      plot(t1',s1)
47      title('Free throw model')
48      ylabel('Euler Angles (degrees)')
49
50      subplot(2,1,2)
51      plot(t2a,s2a)
52      title('Activity (aligned)')
53      ylabel('Euler Angles (degrees)')
54      xlabel('Time (s)')
55
56      linkaxes
57
58      s2=s2a
59
60  end
61      %makes both vectors the same length
62      s=[]
63       if(length(s1)>length(s2))
64            for i=1:length(s2)
65                s(i)=s1(i)
66            end
67            s1=s'
68       else
69            for i=1:length(s1)
70                s(i)=s2(i)
71            end
72            s2=s'
73       end
74
75
76       %obtains the correlation between both vectors
77       c = corr2(s1,s2)
```

Thus, the correlation coefficient between both signals takes values from -1 to 1, with being 1 the maximum value of similarity between both. Therefore, the following code has been implemented to display if the movement of the free throw is detected, its outcome (shot in or out) and its corresponding score on the MATLAB screen. Hence, the closer the correlation coefficient is to 1, the more the activity performed resembles the model, thus obtaining a higher score. However, for values lower than 0.85, the shot will be considered unidentified, as it does not sufficiently resemble the adopted model. In section 7.2, the results and the graphs obtained in this section can be observed.

```
1
2       %0.85 is the threshold stablished to detect a free throw
3       if(c>0.85)
4            fprintf('Free throw detected\n')
5
6    %'okey' shot
7            if(c<0.89)
8                 fprintf('Technique score: 7\n')
9            end
```

```
%'good' shot
      if(c>0.9 & c<0.95)
            fprintf('Technique score: 8\n')

       end
    %'great' shot
     if(c>0.95 & c<0.97)
            fprintf('Technique score: 9\n')

        end

     %'perfect' shot
     if(c>0.97 & c<=1)
            fprintf('Technique score: 10\n')

        end

   else
       fprintf('Free throw not identified')
   end


   %checks if the net sensor has been hit
   k=0;  %initialises the flag to zero
      for j=1:size(M,1)

    %if accel value surpasses the threshold
          if Accelerometer2(j,1)>20
          k=1; %rises flag
          end
      end

      if k==1
          fprintf("Outcome: Player made the basket!!\n")
      else
          fprintf("Outcome: Player missed the basket...\n")
      end
```

6.3 Experimental procedure

One player was invited to participate in the experiment. He was a 16-year-old left-handed male that plays basketball in federated competitions and has nine years of experience in the sport. His job was to shot some free throws while wearing the wrist sensor to collect the information required for the analysis. To ensure the best performance, the player warmed up for 20 minutes before start recording the shots.

Figure 6.9: Full court view of the player shooting.

Although the court has not official dimensions, the distance of the free-throw line (4.5 m) and the height of the basket (3.05 m) are the proper ones. As only free throws are going to be assessed, this fact is not an issue.

In figures 6.10 and 6.11 there are closer looks to the player wearing the Hexiwear and the net sensor placed strategically behind the net to avoid as much as possible the accidental movements that could happen if the ball does not enter inside the hoop but hits the net. Figure 6.12 shows the collision between ball and sensor when a basket is made.

Figure 6.10: Closer look of the player in loading position. Hexiwear's wrist sensor can be seen in blue.

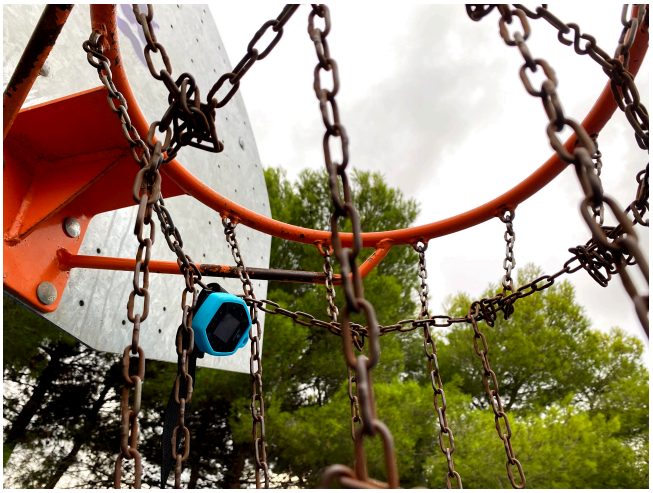

Figure 6.11: Net sensor placed behind the net.

Figure 6.12: Net sensor being hit by the ball.

Chapter 7

Obtained results

This chapter will show the results obtained in the development of the project. Firstly, screenshots of the different screens of the designed mobile application will be shown. Finally, the different graphs obtained during the processing and analysis of the data will be displayed, as well as the final result.

7.1 *Free Throw Trainer* app

The following images from figure 7.2 to figure 7.16 show the different screenshots of the screens implemented in the *Free Throw Trainer* application. Moreover, a quick walkthrough and tutorial will be presented.

7.1.1 User management

User management pages are all the screens in which the user creates, edits and selects the different profiles to start recording his or her training session. They correspond to the highlighted blocks in figure 7.1.

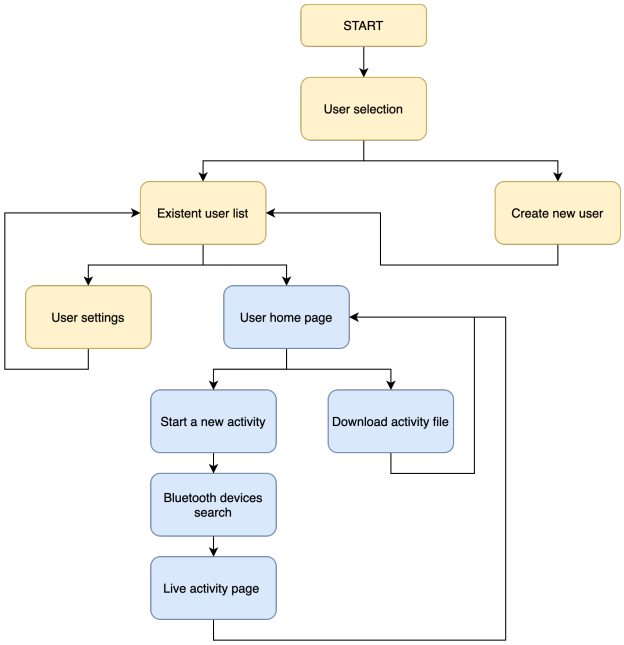

Figure 7.1: Software block diagram. User management screens.

User selection

To begin, the first page, also known as *splash screen*, allows one to either select one of the existent users or to create a new one from scratch.

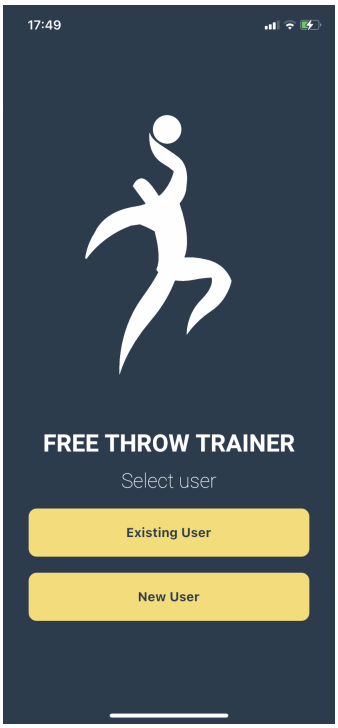

Figure 7.2: Screenshot of the *User selection* screen.

Create new user

This screen is designed to create a new profile. The first field is to input the name (text) and the other three remaining fields are to introduce the data with the numeric keyboard. Once all the form is filled, the `Save` button leads to the *Existing user list*.

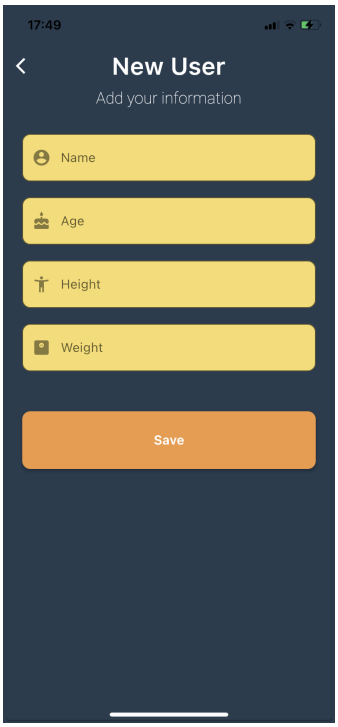

Figure 7.3: Screenshot of the *New user* screen before filling the form.

Existing user list

Here the user can select one of the created profiles. To enter inside one particular user, the name shall be pressed. If he or she wants to edit one particular user, the `pencil` icon next to the name will lead to *User settings*. To delete one profile, just swipe from left to right and it will be erased.

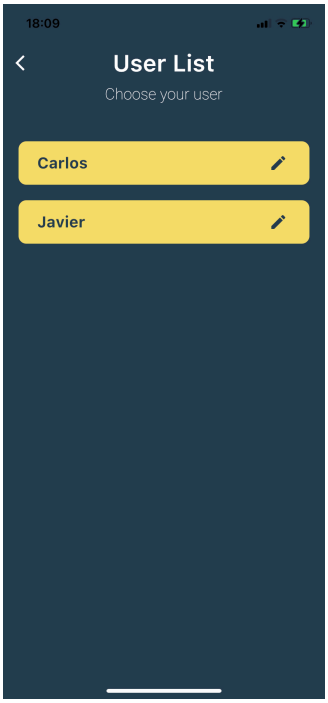

Figure 7.4: Screenshot of the *Existing user list* screen with two created users.

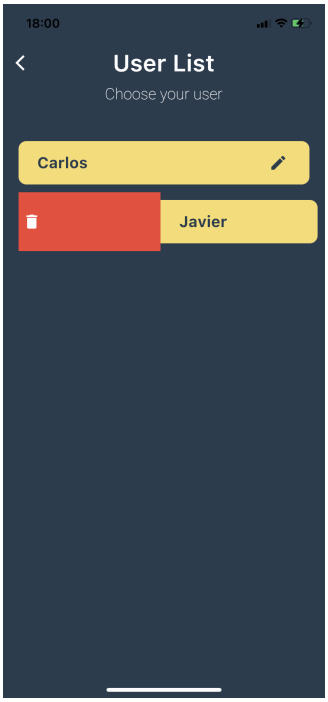

Figure 7.5: Screenshot of the *Existing user list* screen trying to delete a user.

User settings

As explained in 7.1.1 the function of this screen is to modify the personal information of the user. Pressing the `Save` button leads to the *Existing user list* again.

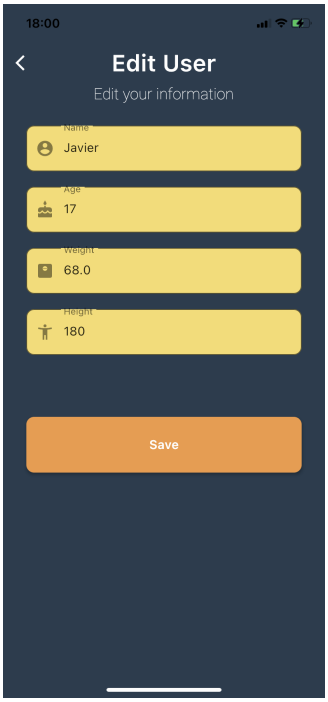

Figure 7.6: Screenshot of the *User settingst* screen trying to delete a user.

7.1.2 User home page

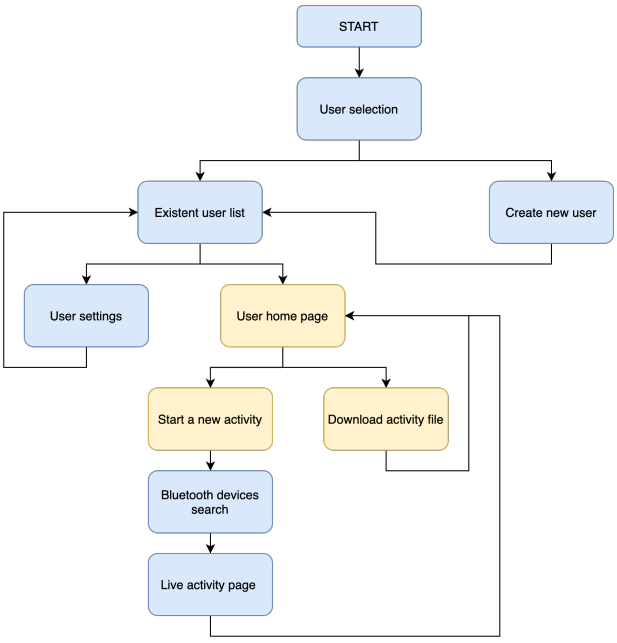

Figure 7.7: Software block diagram. User home page.

Once the user is selected, *User home page* is displayed. In this screen, the button `Start activity` accesses to the *Search and connection of BLE devices* screen. Below this button, the record of previous activities will be shown, indicating the date and time. To download the CSV file of any activity, just press the `download` icon or the text.

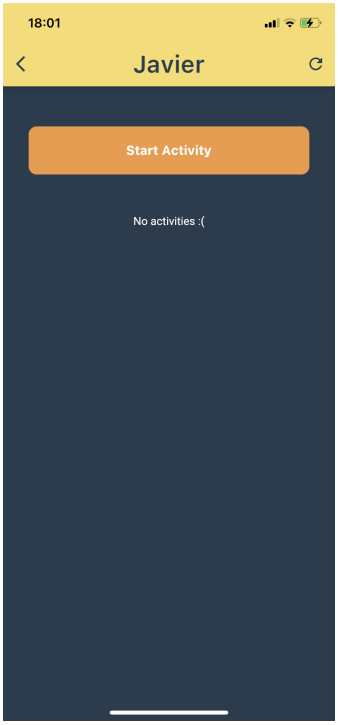

Figure 7.8: Screenshot of the *Home page* with no activities.

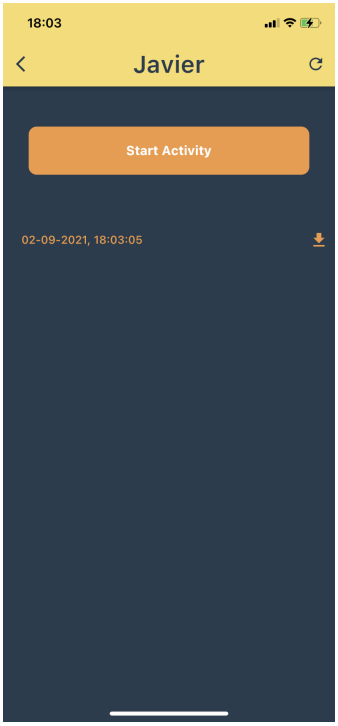

Figure 7.9: Screenshot of the *Home page* with a downloadable activity.

7.1.3 Search and connection of BLE devices

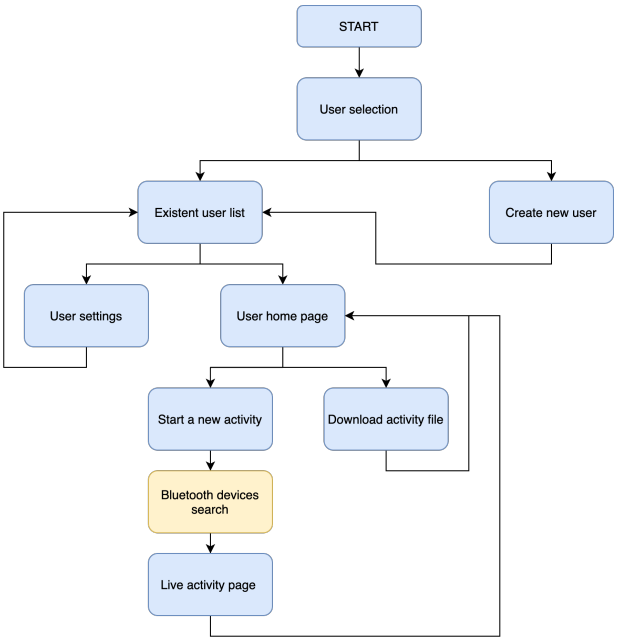

Figure 7.10: Software block diagram. Find devices page.

Before starting the new activity the user has to ensure that both sensors are connected to the smartphone. To scan the devices, press the orange floating button (`search`). Therefore, all the visible Bluetooth devices will appear as well as their UUID. In the app bar, there is a counter initialised to zero that shows how many devices are currently paired. Next to the number, there is a red bin icon (`delete`) that can be used to unpair every BLE gadget and restart the counter.

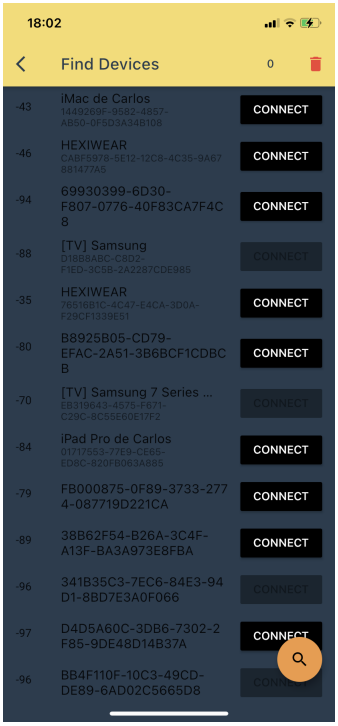

Figure 7.11: Screenshot of the *Find devices* screen after scanning.

If the user clicks on the name, the tile expands and some characteristics can be shown if the manufacturer allows so. In figure 7.12 the locked characteristics of the Hexiwear are shown as "N/A". To pair the device to the app the button `CONNECT` shall be pressed. Once this happens, the counter updates. Due to the inability to rename both Hexiwears, for this experience UUIDs are being used as identifiers. The first device to connect is the one whose UUID starts with CABF and corresponds to the wrist sensor.

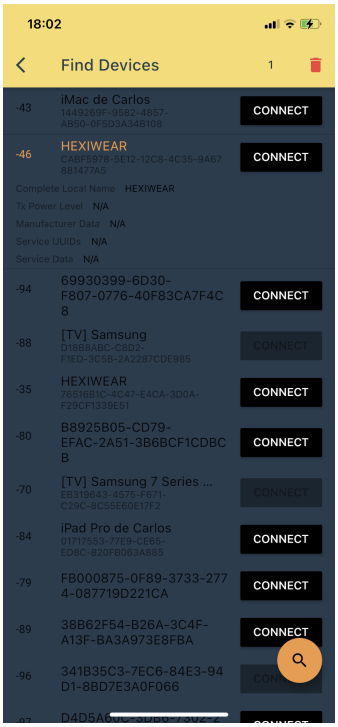

Figure 7.12: Screenshot of the *Find devices* screen after connecting the wrist sensor.

Once it is connected, now it is time to pair the net sensor. In this case, it is the other Hexiwear remaining, but it can also be identified by its UUID, the one starting with 7651. When the second device is connected the counter changes to the word `START`. Pressing this word will lead to the *Live activity* page.

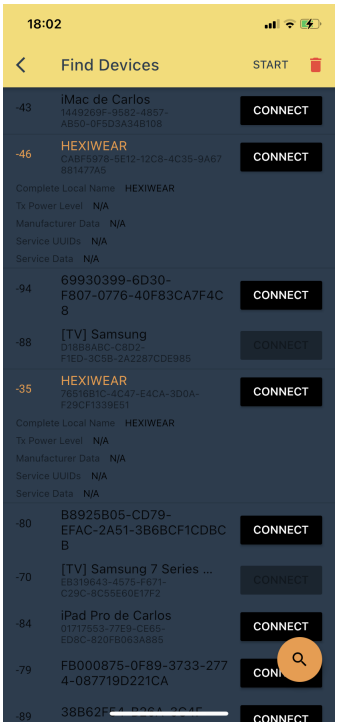

Figure 7.13: Screenshot of the *Find devices* screen after connecting the net sensor. The word START is shown in the app bar.

7.1.4 Live activity page

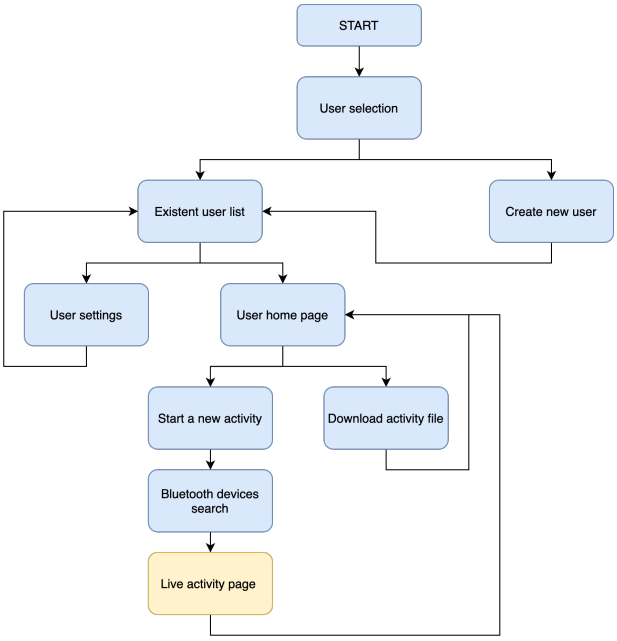

Figure 7.14: Software block diagram. Live activity page

This page is the one that can be seen while the user is performing the shot. It includes a timer and two sections where show the values that both sensors are obtaining in real-time.

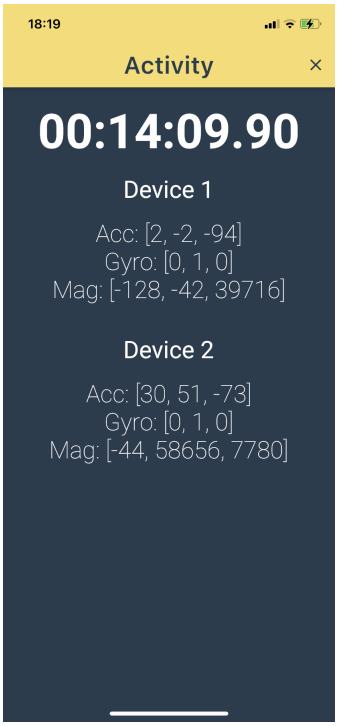

Figure 7.15: Screenshot of the *Live activity* page.

When the activity is over, the `X` button opens a dialog (figure 7.16). Pressing `OK` ends the activity and goes back to the *Find devices* page, disconnecting the wearables to avoid any errors.

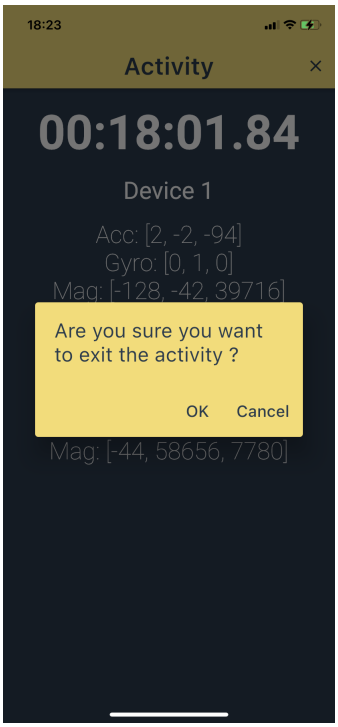

Figure 7.16: Dialog shown when X icon is pressed.

7.2 Analysis

The analysis in this project has two different parts. On the one hand, the script must detect if the player is shooting a free throw reading the data from the wrist sensor. On the other hand, the outcome of the shot is determined using the values given by the accelerometer on the net. Both experiences are analysed separately and combined afterwards.

The results obtained in the processing and validation of the data analysis, which corresponds to the software block diagram shown in figure 4.6, are shown in section 7.2.2.

7.2.1 Model determination

For the motion analysis, one free throw shot with proper technique that went inside the hoop has been chosen as a model, to compare the rest of the attempts with it. In figure 7.17 the values obtained by all the sensors are displayed.

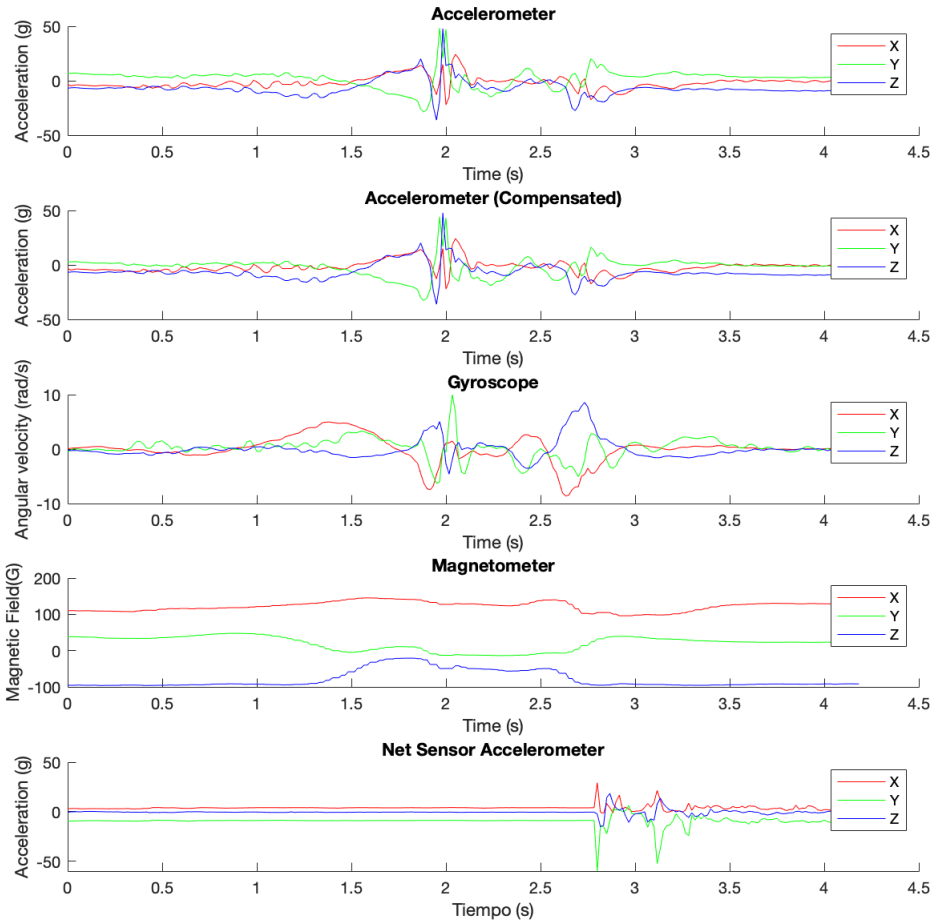

Figure 7.17: Plots of the different data provided by the sensors. The first three correspond to the wrist sensor and the last one to the net one.

Once the data of the sensors is collected properly, the Euler angles of the shot movement can be obtained as explained in section 6.2.3. They can be seen in figure 7.18.

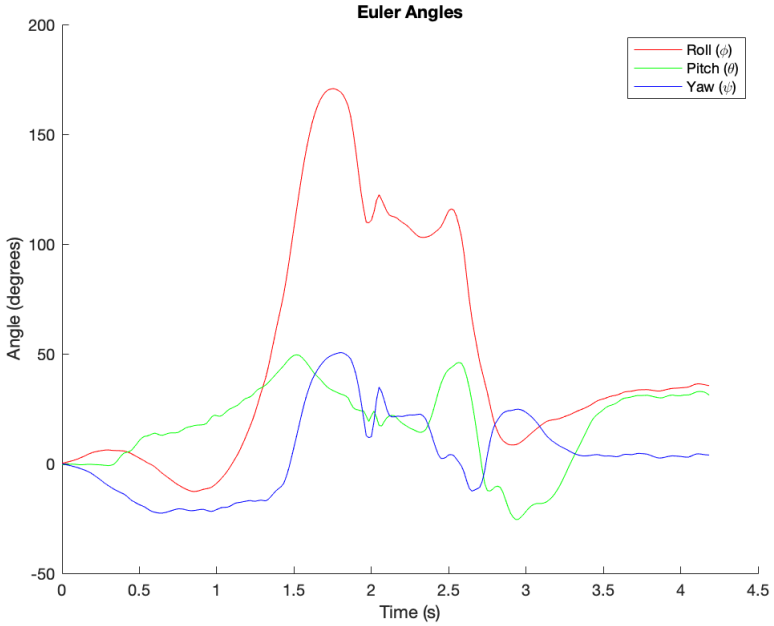

Figure 7.18: Graphical representation of the Euler angles of the free throw used as a model.

In their report, Shankar and Shuresh [28] determined that the roll angle (ϕ) is the best to characterize the free throw shooting action because it reflects better flick velocity and loading angle. Moreover, to verify their assumption it was decided to test the performance of the three angles using the correlation technique explained in section 6.2.4.

Choosing the right angle to characterize the shot

To select the most representative angle, it has decided to compare the free-throw model with other shot performed after the first one with very similar characteristics in technique and outcome. The results gave the highest correlation value to the roll angle (c = 0.9854). Pitch obtained 0.8657 and yaw 0.8979. Therefore, the Shankar and Shuresh hypothesis is validated and roll angle is taken as reference. Figures 7.19, 7.20 and 7.21 shows the corresponding graphs. To make the correlation process easier, aligned plots of the current shots were used, as pointed out in section 6.2.4.

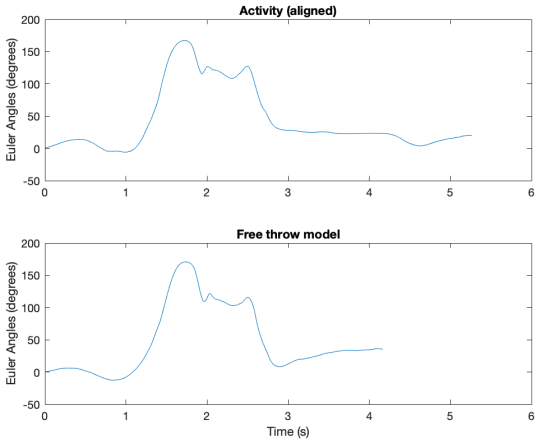

Figure 7.19: Graphical representation of the roll (ϕ) angle of the current shot (top) and the free throw used as a model (bottom). Correlation: c = 0.9854.

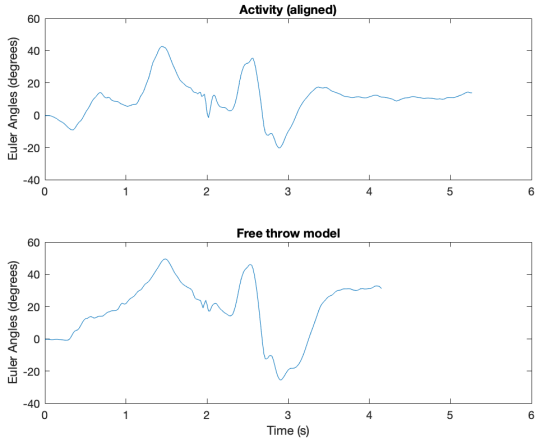

Figure 7.20: Graphical representation of the pitch (θ) angle of the current shot (top) and the free throw used as a model (bottom). Correlation: c = 0.8657.

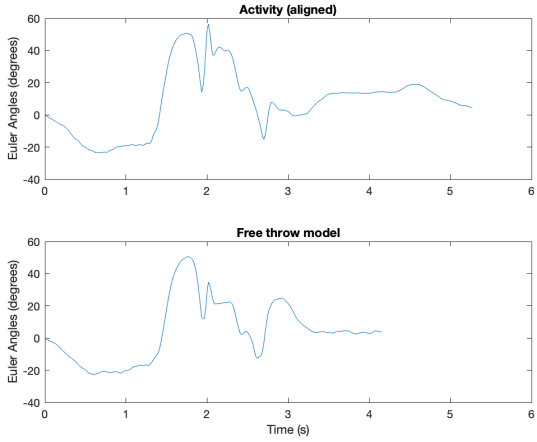

Figure 7.21: Graphical representation of the yaw (ψ) angle of the current shot (top) and the free throw used as a model (bottom). Correlation: c = 0.8979.

Determining when the ball enters the hoop

The approach used to determine the outcome was explained in section 6.2.4. For the sake of simplicity and after the proper test, it was determined that only the X-axis channel of the accelerometer is required to characterize if the ball hits the net sensor. When that happens, a spike appears, as it can be seen in the last graph of figure 7.17. In the MATLAB script, the flag rises when the spike crosses the value of 20 g.

7.2.2 Results

In this section, the results given by the performance of different shots with diverse outcomes will be discussed. Sensor readings, Euler angles representation and correlation values are presented for each case.

Perfect free throw

The shot shown in 7.2.1 is going to be considered as the *perfect* free throw because it is quite close to the one chosen as a model and the player performed a great technique when shooting. Moreover, he made the basket. The data readings from the sensors are the following:

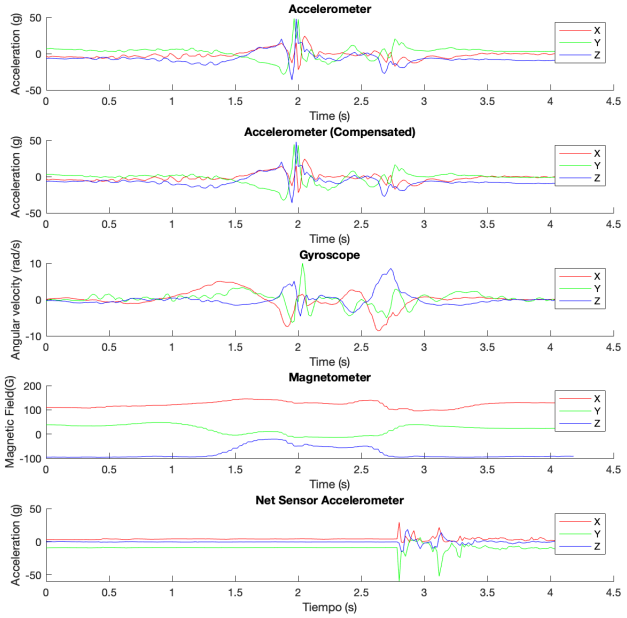

Figure 7.22: Plots of the different data provided by the sensors for the *perfect free throw*. Notice the spike in the last plot which shows when the ball hits the net sensor.

The representation of the roll angle of the activity (aligned) and the model shows the similarity between both graphs (figure 7.23).

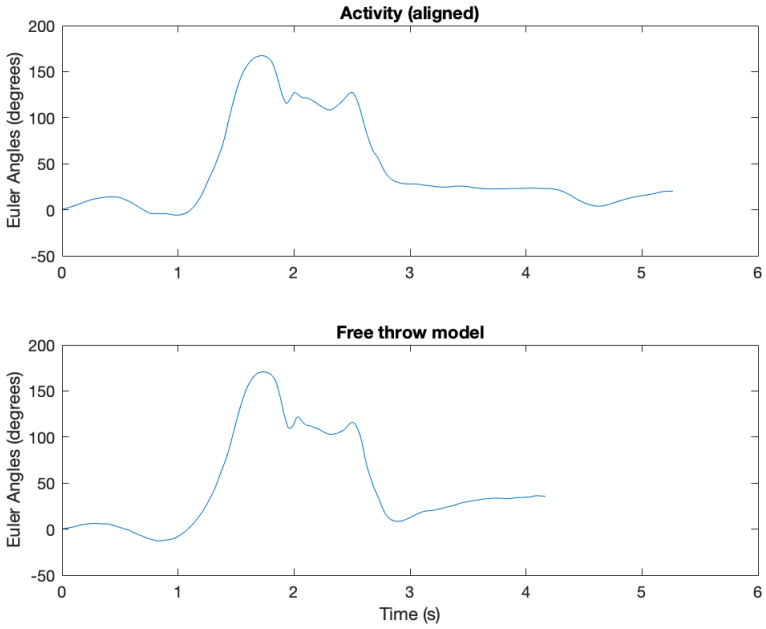

Figure 7.23: Roll (ϕ) angle from the *perfect free throw* (top) and the model (bottom).

Finally, figure 7.24 shows the MATLAB command window in which the verdict of the shot performed is displayed numerically. In this case, the correlation coefficient obtained between both vectors is 0.9854, very close to 1. This means that the activity performed is almost equal to the model and shows its corresponding score, a 10. Since the net sensor also detected the ball going inside the hoop, it prints the message `Outcome:Player made the basket!`.

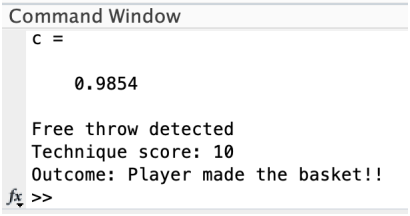

Figure 7.24: MATLAB command window showing the output messages.

Free throw with good technique that scores

This shot was performed with a good technique, but it was not *perfect*. Nevertheless, the player made the basket. To know how well was the attempt sensor data can be used. In figure 7.25 sensor readings are displayed.

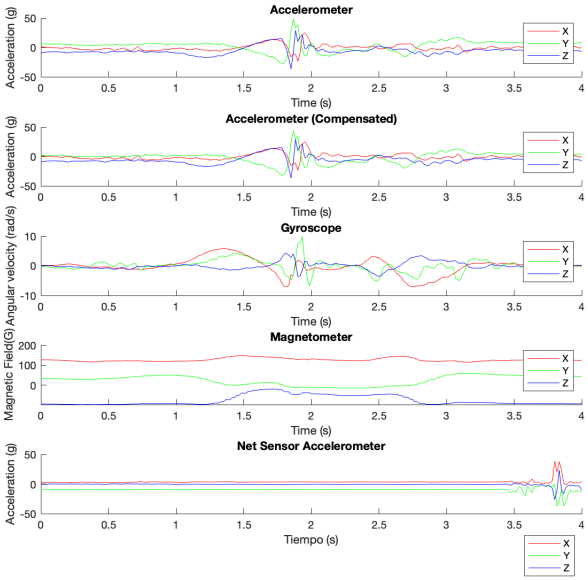

Figure 7.25: Plots of the different data provided by the sensors for a *good free throw*. Notice the spike in the last plot which shows when the ball hits the net sensor.

In this case, the graph of the roll angle is not as close as before but the overall shape remains the same. That suggests that even though it was not perfect, the shot was not bad at all.

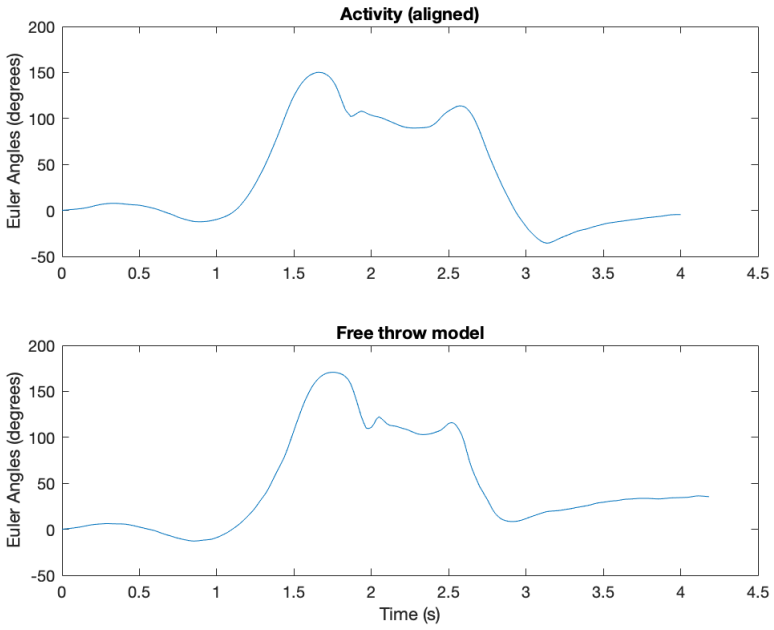

Figure 7.26: Roll (ϕ) angle from a *good free throw* (top) and the model (bottom).

Lastly, the command window shows the final correlation (c = 0.9071) and the corresponding score, an 8. Since the player made the basket, a message in the terminal is written.

```
Command Window

c =

    0.9071

Free throw detected
Technique score: 8
Outcome: Player made the basket!!
fx >>
```

Figure 7.27: MATLAB command window showing the output messages.

Free throw with good technique that does not score

The third case is when the player shoots with a good technique but does not make the basket because the ball hits the rim. Let's see if the analysis can clarify what happened. As always, the first figure shows the sensor readings:

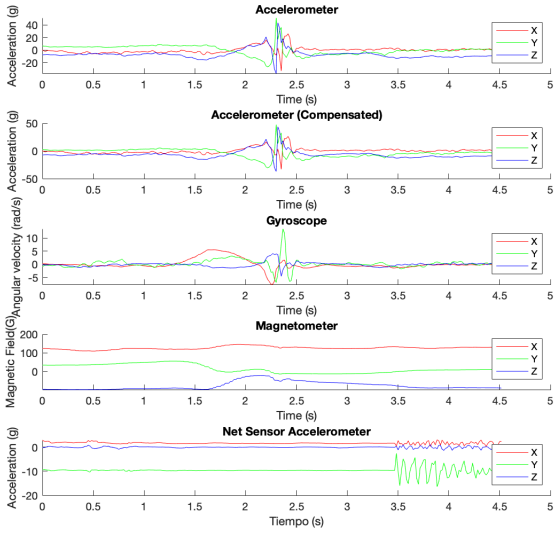

Figure 7.28: Plots of the different data provided by the sensors for the *not so good free throw*. Notice the absence of the spike in the last plot which shows when the ball hits the net sensor.

In figure 7.28 it can be seen that the net sensor does not receive a direct impact but it reflects the oscillations caused by the ball hitting the hoop. When analysing the next graph, figure 7.29, there is one spike missing in the aligned activity at the end of the movement if compared with the model. That fact could be the factor that made the player miss the shot.

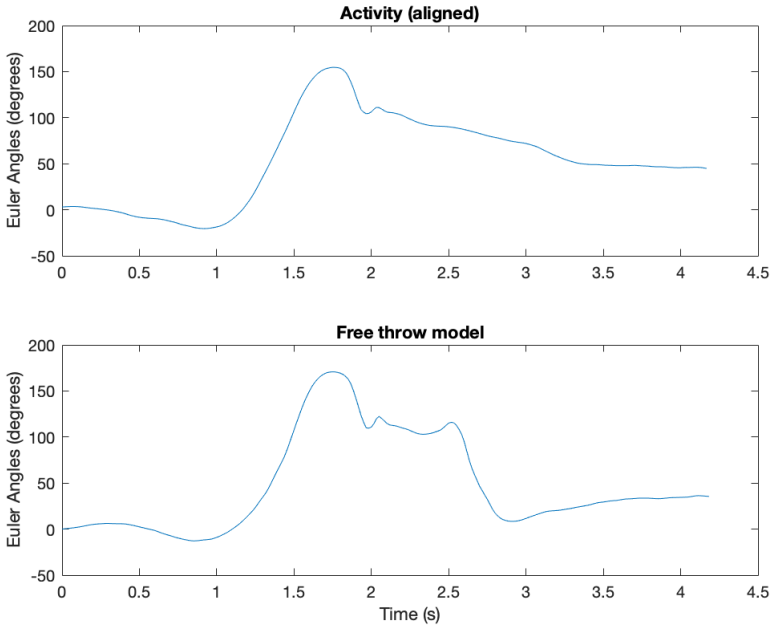

Figure 7.29: Roll (ϕ) angle from a *not so good free throw* (top) and the model (bottom).

To conclude, the MATLAB command window displays the verdict. In this case, the correlation coefficient obtained between both vectors is 0.9059, marked with an 8. Since the net sensor did not detect the ball going inside the hoop, it prints the message `Outcome:Player missed the basket...`.

```
Command Window

c =

    0.9059

Free throw detected
Technique score: 8
Outcome: Player missed the basket..
fx >>
```

Figure 7.30: MATLAB command window showing the output messages.

Free throw with bad technique that does not score

The last case scenario to consider is when the players shoot with a bad technique and, as expected, misses the basket. Looking at the sensor readings:

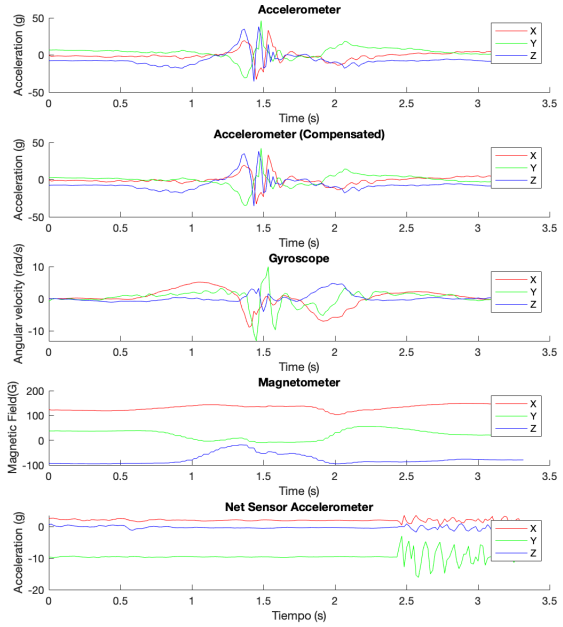

Figure 7.31: Plots of the different data provided by the sensors for the *bad free throw*. Notice the absence of the spike in the last plot which shows when the ball hits the net sensor.

The obtained information is similar to the previous case, the ball does not enter. To interpret the wrist sensor data, Euler angle representation is used as before.

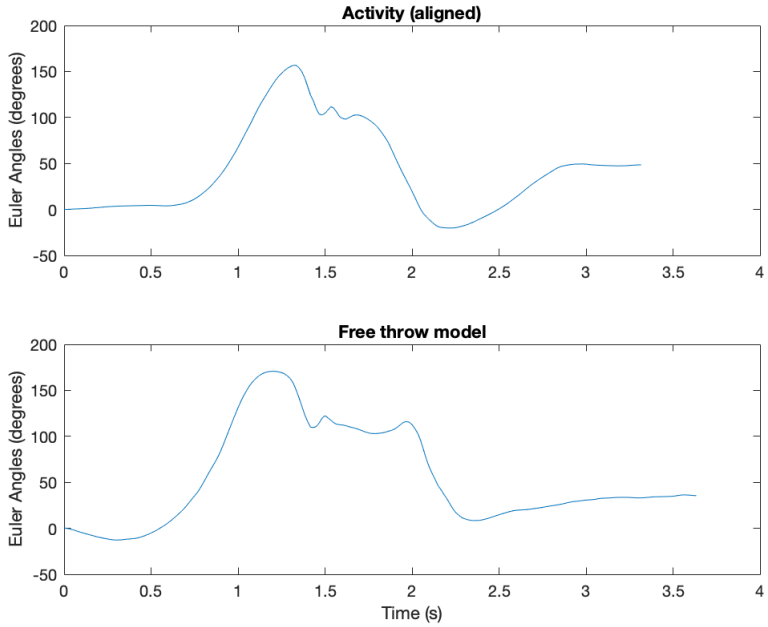

Figure 7.32: Roll (ϕ) angle from a *bad free throw* (top) and the model (bottom).

The *bad free throw* has a roll angle graph that remembers vaguely the model. The overall shape looks akin, but it can be appreciated that crucial parts are not how they should be. Notice the

first and second spikes. In the first one it is pointier and the second one lacks the valley with the other small mountain at the end. It should not give a good value in the correlation, revealing a faulty performance.

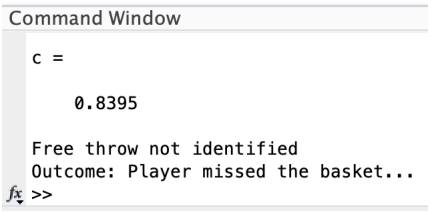

Figure 7.33: MATLAB command window showing the output messages.

As predicted before, the command window shows a c = 0.8395. This value has not enough resemblance to characterize that it belongs to a free throw. Furthermore, the shot was missed, as it remembers the message in the terminal.

Chapter 8

Final conclusions

8.1 Summary of the work done

After having analysed the basketball training routines it has been concluded that it could be an improvement to the sessions to implement some kind of technology in the shooting practice. This has led to the choice of a wearable device for basketball.

Firstly, a study was made of the main wearables on the market with which it is possible to obtain the necessary information from the outside for the classification of the different movements, i.e. a device containing an inertial measurement unit. Following this search, two Hexiwear devices from Mikroelektronika were chosen as the hardware for obtaining the data.

Secondly, a mobile application has been developed for devices with the Android and iOS operating systems, using the Android Studio development environment and Flutter SDK. This application has been configured to connect via Bluetooth Low Energy to the chosen hardware devices. It also allows the activities carried out to be recorded in a local database, as well as downloading in CSV files.

Finally, in an environment with greater computing power such as a computer and with a numerical computation programme such as MATLAB, an analysis methodology for the data collected by the hardware device has been processed and validated, extracting them from the files generated by the application. For data processing, a fusion of the information collected by the inertial measurement unit has been carried out to obtain the orientation of the device and to identify the shots and their outcome. In addition, the acceleration effect of gravity has been compensated and problems derived from the orientation, such as the gimbal lock, have been solved. In the analysis part, the first identification of different types of free throws has been validated, proving the usefulness and success of the prototype.

Therefore, the objectives set out in the project have been successfully achieved, obtaining a prototype capable of recording, identifying and qualify free throws. The project developed shows the wide field of application of wearable devices in basketball as well as in any type of sport,

being able to help in the training of mechanical skills. Moreover, it can be said that it is a multidisciplinary project as it has been necessary to apply knowledge from different fields such as signal processing, programming, graphic design or statistics.

8.2 Proposal for future work

As future work, some improvements could be included in the project. On the analysis side, much more could be done based on the validation carried out. With a larger number of different shots recorded and more samples of each one, much more reliable models could be obtained for each type of shot, providing more accurate statistics. The presence of different players of diverse sex, dominant hand, skill and size will also enrich the data and make the models more useful for more users. Also, with a larger volume of data, other more sophisticated types of analysis such as machine learning and neural networks can be applied and then expand the usefulness to another kind of shots like 3-pointers or lay-ups.

Another possible improvement would be to link the mobile application with the analysis carried out on the computer with a cloud-based database like Firebase, skipping the step of downloading the CSV file in the smartphone and then send it to the computer.

A possible extension of the above improvement would be to merge the app and the analysis into a web application. Here the user could log activities on their mobile app with their device, and at the same time, these would be stored in the web application while being analysed and their statistics could be visualised. Likewise, the ultimate option can be to include the analysis inside the mobile app and thus get rid of the use of an external computer to access the web app or MATLAB, allowing the user to monitor everything with the smartphone.

Chapter 9

Annex: Mobile app code

Although some code has been displayed during the report, it has been preferred to include an annex at the end of the report that contains all the source code of the **Free Throw Trainer** mobile application. The app directory is organized in three foldes: *screens*, *models* and *widgets*. The first one includes the source code for each page seen by the user; the second, the database files to create the different boxes; and the last one are auxiliar files to perform several functions. **main.dart** file is alone and it is used to initialize the process as the root file.

9.1 main.dart

```
1
2  import 'dart:io';
3  import 'package:flutter/cupertino.dart';
4  import 'package:flutter/material.dart';
5  import 'package:get/get.dart';
6  import 'package:hive/hive.dart';
7  import 'package:hive_intro/constants.dart';
8  import 'package:hive_intro/models/activity.dart';
9  import 'package:hive_intro/models/movement.dart';
10 import 'package:hive_intro/screens/splash.dart';
11 import 'package:path_provider/path_provider.dart' as pathProvider;
12
13 import 'models/user.dart';
14
15 void main() async {
16   WidgetsFlutterBinding.ensureInitialized();
17
18   Directory directory = await pathProvider.getApplicationDocumentsDirectory();
19   Hive.init(directory.path);
20
21   Hive.registerAdapter(UserAdapter());
22   Hive.registerAdapter(ActivityAdapter());
23   Hive.registerAdapter(MovementAdapter());
24
25   Hive.openBox<String>('users');
26
```

```
  // var usersBox = await Hive.openBox('users');
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of the application.
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Hive Intro',
      theme: ThemeData(
        primaryColor: kYellow,
        accentColor: kOrange,
        scaffoldBackgroundColor: kBlue,
        textTheme: const TextTheme(
          bodyText1: TextStyle(
            color: Colors.white,
            fontFamily: 'Roboto',
            fontWeight: FontWeight.w400,
          ),
          bodyText2: TextStyle(
            color: Colors.white,
            fontFamily: 'Roboto',
            fontWeight: FontWeight.w400,
          ),
          // button: TextStyle(color: Colors.white),
        ),
      ),
      home: const SplashScreen(),
    );
  }
}
```

9.2 screens

9.2.1 splash.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:hive_intro/constants.dart';
import 'package:hive_intro/screens/ble_search.dart';
import 'package:hive_intro/screens/new_user.dart';
import 'package:hive_intro/screens/user_list.dart';

class SplashScreen extends StatelessWidget {
  const SplashScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
```

```
                //Splash image
                const Image(image: AssetImage('images/splash_image.png')),

                //space
                const SizedBox(
                  height: 50,
                ),

                //Welcome text
                const Text(
                  'FREE THROW TRAINER',
                  style: TextStyle(
                    fontWeight: FontWeight.w600,
                    fontSize: 30,
                  ),
                ),
                //space
                const SizedBox(
                  height: 15,
                ),

                //Select user
                const Text(
                  'Select user',
                  style: TextStyle(
                    // fontWeight: FontWeight.w100,
                    fontSize: 25,
                    fontWeight: FontWeight.w100,
                  ),
                ),
                const SizedBox(
                  height: 20,
                ),
                TextButton(
                  style: flatButtonStyle,
                  child: Text(
                    'Existing User',
                    style: titleStyle,
                  ),
                  onPressed: () => Get.to(() => UserList()),
                ),
                const SizedBox(
                  height: 20,
                ),
                TextButton(
                  style: flatButtonStyle,
                  onPressed: () => Get.to(() => const NewUser()),
                  child: Text(
                    'New User',
                    style: titleStyle,
                  ),
                ),
              ],
            ),
          ),
        ),
      );
  }
}
```

9.2.2 new-user.dart

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';
import 'package:hive_intro/constants.dart';
import 'package:hive_intro/hive_db.dart';
import 'package:hive_intro/models/user.dart';
import 'package:hive_intro/screens/user_list.dart';
import 'package:hive_intro/widgets/custom_form_widget.dart';

class NewUser extends StatelessWidget {
  const NewUser({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: kBlue,
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: Colors.white, //change your color here
        ),
        backgroundColor: kBlue,
        title: const Text(
          'New User',
          style: TextStyle(
            fontSize: 30,
            fontWeight: FontWeight.w600,
            color: Colors.white,
          ),
        ),
        centerTitle: true,
        elevation: 0,
      ),
      body: Center(
        child: Column(
          children: [
            const Text(
              'Add your information',
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.w100,
              ),
            ),
            const SizedBox(height: 20),
            Expanded(
              child: Card(
                color: kBlue,
                child: const MyCustomForm(),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

// Create a corresponding State class.
// This class holds data related to the form.
```

```
class MyCustomForm extends StatefulWidget {
  const MyCustomForm({Key? key}) : super(key: key);

  @override
  _MyCustomFormState createState() => _MyCustomFormState();
}

class _MyCustomFormState extends State<MyCustomForm> {
  final _formKey = GlobalKey<FormState>();

  //controllers to retrieve the data from the form
  var nameController = TextEditingController();
  var ageController = TextEditingController();
  var heightController = TextEditingController();
  var weightController = TextEditingController();

  //variables for the datepicker
  DateTime date = DateTime.now(); //initialize to today

  //function to display and get the selected date
  Future<void> selectTimePicker(BuildContext context) async {
    DateTime? picked = await showDatePicker(
      context: context,
      initialDate: date,
      firstDate: DateTime(1930),
      lastDate: DateTime(2021),
    );

    if (picked != null && picked != date) {
      setState(() {
        date = picked;
        print(date.toString());
        //    selectTimePicker = new DateFormat.yMMMMd("en_US").format(picked);
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    nameController = TextEditingController();
    ageController = TextEditingController();
    heightController = TextEditingController();
    weightController = TextEditingController();

    // Build a Form widget using the _formKey created above.
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          //NAME -> write
          CustomFormWidget(
            controller: nameController,
            content: 'Name',
            icon: Icons.account_circle_sharp,
            willAcceptNumbersOnly: false,
          ),

          // age
          CustomFormWidget(
            controller: ageController,
```

```
            icon: Icons.cake,
            content: 'Age',
          ),

          //height
          CustomFormWidget(
            controller: heightController,
            icon: Icons.accessibility_new,
            content: 'Height',
          ),

          //weight
          CustomFormWidget(
            controller: weightController,
            icon: Icons.monitor_weight,
            content: 'Weight',
          ),

          // const Spacer(),
          const SizedBox(
            height: 20,
          ),
          //SAVE BUTTON
          Padding(
            padding: const EdgeInsets.symmetric(
              vertical: 20,
              horizontal: 20,
            ),
            child: SizedBox(
              height: 72,
              width: double.infinity,
              child: ElevatedButton(
                child: const Text('Save',
                    style:
                        TextStyle(fontSize: 16, fontWeight: FontWeight.w600)),
                style: ElevatedButton.styleFrom(
                    primary: kOrange, //background color of button
                    elevation: 3, //elevation of button
                    shape: RoundedRectangleBorder(
                        //to set border radius to button
                        borderRadius: BorderRadius.circular(10)),
                    padding:
                        const EdgeInsets.all(20) //content padding inside button
                    ),
                onPressed: () async {
                  // Validate returns true if the form is valid, or false otherwise.
                  if (_formKey.currentState!.validate()) {
                    //check if we are editing
                    HiveDB _hive_db = HiveDB();
                    _hive_db.addNewUser(
                      User(activities: [])
                        ..name = nameController.text
                        ..age = int.parse(ageController.text)
                        ..height = int.parse(heightController.text)
                        ..weight = double.parse(
                          weightController.text,
                        ),
                    );

                    nameController.clear();
                    ageController.clear();
```

```
                    heightController.clear();
                    weightController.clear();

                    Get.to(() => const UserList());
                  }
                },
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

9.2.3 user-list.dart

```
import 'dart:core';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:hive/hive.dart';
import 'package:hive_intro/hive_db.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:hive_intro/models/user.dart';
import 'package:hive_intro/screens/home.dart';
import 'package:hive_intro/screens/user_edit.dart';

import '../constants.dart';

class UserList extends StatelessWidget {
  const UserList({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: kBlue,
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: Colors.white, //change your color here
        ),
        backgroundColor: kBlue,
        title: const Text(
          'User List',
          style: TextStyle(
            color: Colors.white,
            fontSize: 30,
            fontWeight: FontWeight.w600,
          ),
        ),
        centerTitle: true,
        elevation: 0,
      ),
      body: Center(
        child: Column(
          // crossAxisAlignment: CrossAxisAlignment.start,
          children: const [
            Text(
              'Choose your user',
              style: TextStyle(
```

```
                fontSize: 20,
                fontWeight: FontWeight.w100,
              ),
            ),
            SizedBox(height: 20),
            Padding(
              padding: EdgeInsets.symmetric(
                vertical: 20,
                horizontal: 20,
              ),
              child: UserListItem(),
            ),
          ],
        ),
      ),
    );
  }
}

class UserListItem extends StatefulWidget {
  const UserListItem({Key? key}) : super(key: key);

  @override
  _UserListItem createState() => _UserListItem();
}

class _UserListItem extends State<UserListItem> {
  final HiveDB _hiveDB = HiveDB();

  @override
  void initState() {
    super.initState();
    // getUsers();
  }

  @override
  Widget build(BuildContext context) {
    return ValueListenableBuilder<Box<String>>(
      valueListenable: Hive.box<String>('users').listenable(),
      builder: (ctx, box, _) => ListView.builder(
        shrinkWrap: true,
        itemCount: box.length,
        itemBuilder: (context, index) => box.isEmpty
            ? const Center(
                child: Text('No users found'),
              )
            : Dismissible(
                key: UniqueKey(),
                direction: DismissDirection.startToEnd,
                background: Container(
                  color: Colors.red,
                  padding: const EdgeInsets.only(left: 5),
                  child: const Align(
                    alignment: Alignment.centerLeft,
                    child: Icon(
                      Icons.delete,
                      color: Colors.white,
                    ),
                  ),
                ),
                onDismissed: (_) async {
```

```
                    //open the box
                    var users = Hive.box<String>('users');

                    // remove the i-th person
                    _hiveDB.removeUserBox(users.getAt(index)!);

                    // remove entry from general user list
                    users.deleteAt(index);
                  },
                  child: Padding(
                    padding: const EdgeInsets.symmetric(vertical: 10.0),
                    child: ListTile(
                      tileColor: kYellow,
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                      title: Align(
                        alignment: Alignment.centerLeft,
                        child: TextButton(
                            child: Text(
                              box.getAt(index)!,
                              style: TextStyle(
                                color: kBlue,
                                fontSize: 20,
                                fontWeight: FontWeight.bold,
                              ),
                              textAlign: TextAlign.start,
                            ),
                            onPressed: () async {
                              Get.to(
                                () => Home(
                                  name: box.getAt(index)!,
                                ),
                              );
                            }),
                      ),
                      trailing: IconButton(
                        onPressed: () async {
                          print(box.getAt(index)!);

                          User _user = await _hiveDB
                              .returnUserFromAName(box.getAt(index)!);

                          Get.to(
                            () => EditUser(
                              user: _user,
                              indexOfName: index,
                            ),
                          );
                        },
                        icon: Icon(
                          Icons.edit,
                          color: kBlue,
                        ),
                      ),
                    ),
                  ),
                ),
              ),
          ),
        );
      }
```

```
}
```

9.2.4 user-edit.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:hive/hive.dart';
import 'package:hive_intro/hive_db.dart';
import 'package:hive_intro/models/user.dart';
import 'package:hive_intro/widgets/custom_form_widget.dart';

import '../constants.dart';

class EditUser extends StatefulWidget {
  const EditUser({
    Key? key,
    required this.user,
    required this.indexOfName,
  }) : super(key: key);

  final User user;
  final int indexOfName;

  @override
  State<EditUser> createState() => _EditUserState();
}

class _EditUserState extends State<EditUser> {
  final _formKey = GlobalKey<FormState>();

  var nameController = TextEditingController();

  var ageController = TextEditingController();

  var heightController = TextEditingController();

  var weightController = TextEditingController();

  @override
  void initState() {
    super.initState();
    nameController.text = widget.user.name!;
    ageController.text = widget.user.age!.toString();
    heightController.text = widget.user.height!.toString();
    weightController.text = widget.user.weight!.toString();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        backgroundColor: kBlue,
        iconTheme: IconThemeData(
          color: Colors.white, //change your color here
        ),
        title: const Text(
          'Edit User',
          style: TextStyle(
              fontSize: 30, fontWeight: FontWeight.w600, color: Colors.white),
```

```
        ),
        centerTitle: true,
        elevation: 0,
      ),
      body: Column(
        children: [
          const Text(
            'Edit your information',
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.w200,
            ),
          ),
          const SizedBox(height: 20),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 10),
            child: Form(
              key: _formKey,
              child: Column(
                children: <Widget>[
                  CustomFormWidget(
                    controller: nameController,
                    content: 'Name',
                    icon: Icons.account_circle_sharp,
                    willAcceptNumbersOnly: false,
                  ),
                  CustomFormWidget(
                    controller: ageController,
                    icon: Icons.cake,
                    content: 'Age',
                  ),
                  CustomFormWidget(
                    controller: weightController,
                    icon: Icons.monitor_weight,
                    content: 'Weight',
                  ),
                  CustomFormWidget(
                    controller: heightController,
                    icon: Icons.accessibility_new,
                    content: 'Height',
                  ),
                  const SizedBox(
                    height: 50,
                  ),
                  Padding(
                    padding: const EdgeInsets.symmetric(
                      vertical: 20,
                      horizontal: 20,
                    ),
                    child: SizedBox(
                      height: 72,
                      width: double.infinity,
                      child: ElevatedButton(
                        child: const Text('Save',
                            style: TextStyle(
                                fontSize: 16, fontWeight: FontWeight.w600)),
                        style: ElevatedButton.styleFrom(
                            primary: kOrange, //background color of button
                            elevation: 3, //elevation of button
                            shape: RoundedRectangleBorder(
                                //to set border radius to button
```

```
                                borderRadius: BorderRadius.circular(10)),
                            padding: const EdgeInsets.all(
                                20) //content padding inside button
                            ),
                        onPressed: () async {
                          // Validate returns true if the form is valid, or false
    otherwise.
                          if (_formKey.currentState!.validate()) {
                            HiveDB _hive_db = HiveDB();
                            _hive_db.printAll();

                            if (_formKey.currentState!.validate()) {
                              User user =
                                  User(activities: widget.user.activities)
                                    ..name = nameController.text
                                    ..age = int.parse(ageController.text)
                                    ..height = int.parse(heightController.text)
                                    ..weight =
                                        double.parse(weightController.text);

                              _hive_db.updateUser(user, widget.indexOfName);
                            }

                            Get.back(closeOverlays: true);

                            Get.snackbar(
                              'Yay',
                              'Created new user',
                              colorText: kBlue,
                              titleText: Text(
                                'Done',
                                style: TextStyle(
                                  fontWeight: FontWeight.bold,
                                  color: kBlue,
                                ),
                              ),
                              messageText: Text(
                                'Updated successfully',
                                style: TextStyle(
                                  color: kBlue,
                                ),
                              ),
                              backgroundColor: kYellow,
                              snackPosition: SnackPosition.TOP,
                            );
                          }
                        },
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

9.2.5 home.dart

```dart
import 'dart:io';
import 'package:csv/csv.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:hive/hive.dart';
import 'package:hive_intro/constants.dart';
import 'package:hive_intro/models/user.dart';
import 'package:hive_intro/screens/ble_search.dart';
import 'package:hive_intro/widgets/latest_activity.dart';
import 'package:intl/intl.dart';
import 'package:path_provider/path_provider.dart';
import 'package:permission_handler/permission_handler.dart';

class Home extends StatefulWidget {
  const Home({
    Key? key,
    required this.name,
  }) : super(key: key);

  final String name;

  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  // Future<void> ensureCorrectBoxIsOpen() async {
  //   await Hive.openBox<User>(widget.name);
  // }
  //
  // highLevelCalls() async {
  //   await ensureCorrectBoxIsOpen();
  //   print('box is now open for sure');
  // }

  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: kBlue, //change your color here
        ),
        backgroundColor: kYellow,
        centerTitle: true,
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh),
            onPressed: () => setState(() {}),
          ),
        ],
        title: Text(
          widget.name,
          style: TextStyle(
            fontSize: 30,
```

```
            color: kBlue,
            fontWeight: FontWeight.w600,
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const SizedBox(
                height: 50,
              ),
              TextButton(
                style: orangeFlatButtonStyle,
                child: Text(
                  'Start Activity',
                  style: titleStyle.copyWith(color: Colors.white),
                ),
                onPressed: () => Get.to(
                  () => BluetoothScreen(
                    nameOfUser: widget.name,
                  ),
                ),
              ),
              const SizedBox(
                height: 50,
              ),
              ValueListenableBuilder<Box<User>>(
                valueListenable: Hive.box<User>(widget.name).listenable(),
                builder: (_, box, __) => box.getAt(0)!.activities.isNotEmpty
                    ? ListView.builder(
                        physics: const BouncingScrollPhysics(),
                        shrinkWrap: true,
                        itemCount: box.getAt(0)!.activities.length,
                        itemBuilder: (ctx, index) {
                          return TextButton(
                            onPressed: () async {
                              //when we press the button it creates the csv
                              List<List<dynamic>> data = [
                                [
                                  "Time_Stamp",
                                  "Acc_X",
                                  "Acc_Y",
                                  "Acc_Z",
                                  "Gyr_X",
                                  "Gyr_Y",
                                  "Gyr_Z",
                                  "Mag_X",
                                  "Mag_Y",
                                  "Mag_Z",
                                  "Acc_2_X",
                                  "Acc_2_Y",
                                  "Acc_2_Z",
                                  "Gyr_2_X",
                                  "Gyr_2_Y",
                                  "Gyr_2_Z",
                                  "Mag_2_X",
                                  "Mag_2_Y",
                                  "Mag_2_Z",
                                ],
```

```
                                ];

                                for (var movement
                                    in box.getAt(0)!.activities[index].mov) {
                                  // print('got a movement object ${movement}');
                                  data.add(
                                    [
                                      movement.timestamp.millisecond,
                                      movement.accX,
                                      movement.accY,
                                      movement.accZ,
                                      movement.gyrX,
                                      movement.gyrY,
                                      movement.gyrZ,
                                      movement.magX,
                                      movement.magY,
                                      movement.magZ,
                                      movement.accX2,
                                      movement.accY2,
                                      movement.accZ2,
                                      movement.gyrX2,
                                      movement.gyrY2,
                                      movement.gyrZ2,
                                      movement.magX2,
                                      movement.magY2,
                                      movement.magZ2,
                                    ],
                                  );
                                }

                                //this part works
                                String csvData =
                                    const ListToCsvConverter().convert(data);

                                //this line prints all the data from the csv to the
    terminal, it works
                                // debugPrint(csvData.toString(), wrapWidth: 1024);

                                var _isGranted =
                                    await Permission.storage.request().isGranted;
                                print(_isGranted);

                                if (await Permission.storage
                                    .request()
                                    .isGranted) {
                                  final String directory = Platform.isIOS
                                      ? (await getApplicationDocumentsDirectory())
                                          .path
                                      : (await getExternalStorageDirectory())!
                                          .path;

                                  final path =
                                      "$directory/csv-${DateTime.now()}.csv";
                                  final File file = File(path);
                                  await file.writeAsString(csvData, flush: true);

                                  print(
                                      'completed save successfully and saved file to '
                                      "$directory/csv-${DateTime.now()}.csv");
                                } else {
                                  await Permission.storage.request();
```

```
                                }
                              },
                              child: Row(
                                children: [
                                  Padding(
                                    padding: const EdgeInsets.all(15.0),
                                    child: Text(
                                      DateFormat('dd-MM-yyyy, HH:mm:ss')
                                          .format(box
                                              .getAt(0)!
                                              .activities[index]
                                              .date)
                                          .toString(),
                                      style: TextStyle(color: kOrange),
                                    ),
                                  ),
                                  const Spacer(),
                                  const Icon(
                                    Icons.download,
                                    color: Color(0xffF29A42),
                                  ),
                                ],
                              ),
                            );
                          },
                        )
                      : const Text('No activities :('),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

9.2.6 ble-search.dart

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter_blue/flutter_blue.dart';
import 'package:get/get.dart';
import 'package:hive_intro/widgets/ble_widgets.dart';

import '../constants.dart';
import 'live_activity.dart';

class BluetoothScreen extends StatelessWidget {
  const BluetoothScreen({
    Key? key,
    required this.nameOfUser,
  }) : super(key: key);

  final String nameOfUser;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<BluetoothState>(
          stream: FlutterBlue.instance.state,
          initialData: BluetoothState.unknown,
```

```
          builder: (c, snapshot) {
            final state = snapshot.data;
            if (state == BluetoothState.on) {
              return FindDevicesScreen(
                nameOfUser: nameOfUser,
              );
            }
            return BluetoothOffScreen(state: state!);
          }),
    );
  }
}

class BluetoothOffScreen extends StatelessWidget {
  const BluetoothOffScreen({Key? key, required this.state}) : super(key: key);

  final BluetoothState state;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.lightBlue,
      body: Center(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            const Icon(
              Icons.bluetooth_disabled,
              size: 200.0,
              color: Colors.white54,
            ),
            Text(
              'Bluetooth Adapter is ${state != null ? state.toString().substring(15) : '
    not available'}.',
              style: Theme.of(context)
                  .primaryTextTheme
                  .subtitle1!
                  .copyWith(color: Colors.white),
            ),
          ],
        ),
      ),
    );
  }
}

class FindDevicesScreen extends StatelessWidget {
  FindDevicesScreen({
    Key? key,
    required this.nameOfUser,
  }) : super(key: key);

  final String nameOfUser;
  List<BluetoothDevice> deviceList = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(
          color: kBlue, //change your color here
```

```
        ),
        title: Text(
          'Find Devices',
          style: TextStyle(
            color: kBlue,
          ),
        ),
        actions: [
          StreamBuilder<List<BluetoothDevice>>(
            stream: Stream.periodic(const Duration(seconds: 1)).asyncMap(
              (_) => FlutterBlue.instance.connectedDevices,
            ),
            initialData: [],
            builder: (c, snapshot) => TextButton(
              child: Text(
                snapshot.data != null
                    ? snapshot.data!.length == 2
                        ? 'START'
                        : snapshot.data!.length.toString()
                    : "0",
                style: TextStyle(color: kBlue),
              ),
              onPressed: () async {
                deviceList = await FlutterBlue.instance.connectedDevices;

                for (var device in deviceList) {
                  print('now going to discover services');
                  await device.discoverServices();
                  print(
                      device.services.toList().toString() + 'are the services');
                }

                Get.to(
                  () => LiveActivityScreen(
                    nameOfUser: nameOfUser,
                    // deviceList: deviceList,
                  ),
                );
              },
            ),
          ),
          StreamBuilder<List<BluetoothDevice>>(
            stream: Stream.periodic(const Duration(seconds: 1)).asyncMap(
              (_) => FlutterBlue.instance.connectedDevices,
            ),
            initialData: [],
            builder: (c, snapshot) => IconButton(
              icon: const Icon(Icons.delete),
              color: Colors.red,
              onPressed: () {
                for (BluetoothDevice device in snapshot.data!) {
                  device.disconnect();
                }
              },
            ),
          ),
        ],
      ),
      body: RefreshIndicator(
        onRefresh: () => FlutterBlue.instance.startScan(
          timeout: const Duration(seconds: 4),
```

```
        ),
        child: SingleChildScrollView(
          child: Column(
            children: <Widget>[
              StreamBuilder<List<ScanResult>>(
                stream: FlutterBlue.instance.scanResults,
                initialData: const [],
                builder: (c, snapshot) => Column(
                  children: snapshot.data!
                      .map(
                        (r) => ScanResultTile(
                          result: r,
                          onTap: () async {
                            print('trying to connect');

                            // deviceList.add(r.device);
                            print('finished functions');

                            await r.device.connect();
                            print('connected');

                            await r.device.discoverServices();

                            // );
                          },
                        ),
                      )
                      .toList(),
                ),
              ),
            ],
          ),
        ),
      ),
      floatingActionButton: StreamBuilder<bool>(
        stream: FlutterBlue.instance.isScanning,
        initialData: false,
        builder: (c, snapshot) {
          if (snapshot.data!) {
            return FloatingActionButton(
              child: const Icon(Icons.stop),
              onPressed: () => FlutterBlue.instance.stopScan(),
              backgroundColor: Colors.red,
            );
          } else {
            return FloatingActionButton(
              child: const Icon(Icons.search),
              onPressed: () => FlutterBlue.instance.startScan(
                timeout: const Duration(seconds: 4),
              ),
            );
          }
        },
      ),
    );
  }
}
```

9.2.7 live-activity.dart

```
import 'dart:async';
```

```dart
import 'dart:math';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_blue/flutter_blue.dart';
import 'package:hive/hive.dart';
import 'package:hive_intro/helper.dart';
import 'package:hive_intro/hive_db.dart';
import 'package:hive_intro/models/movement.dart';
import 'package:hive_intro/models/user.dart';
import 'package:hive_intro/widgets/ble_widgets.dart';
import 'package:stop_watch_timer/stop_watch_timer.dart';
import 'package:get/get.dart';

import '../constants.dart';

class LiveActivityScreen extends StatefulWidget {
  const LiveActivityScreen({
    Key? key,
    required this.nameOfUser,
    // required this.deviceList,
  }) : super(key: key);

  // final List<BluetoothDevice> deviceList;
  final String nameOfUser;

  @override
  State<LiveActivityScreen> createState() => _LiveActivityScreenState();
}

class _LiveActivityScreenState extends State<LiveActivityScreen> {
  var displayTime = ''.obs;
  var initProcessComplete = false.obs;

  late StopWatchTimer _stopWatchTimer;
  late Timer _timer;

  BluetoothCharacteristic? accelerometerCharacteristic1;
  BluetoothCharacteristic? gyroscopeCharacteristic1;
  BluetoothCharacteristic? magnetometerCharacteristic1;

  BluetoothCharacteristic? accelerometerCharacteristic2;
  BluetoothCharacteristic? gyroscopeCharacteristic2;
  BluetoothCharacteristic? magnetometerCharacteristic2;

  List accelerometerValues1 = [0, 0, 0];
  List gyroscopeValues1 = [0, 0, 0];
  List magnetometerValues1 = [0, 0, 0];

  List accelerometerValues2 = [0, 0, 0];
  List gyroscopeValues2 = [0, 0, 0];
  List magnetometerValues2 = [0, 0, 0];

  List deviceList = [];

  final HiveDB _hiveDB = HiveDB();

  Future<void> getDevices() async {
    deviceList = await FlutterBlue.instance.connectedDevices;
    print('got ${deviceList.length} devices');

    for (var device in deviceList) {
```

```
63      print('now going to discover services');
64      await device.discoverServices();
65      // print(device.services.toList().toString() + 'are the services');
66    }
67
68    setState(() {});
69  }
70
71  highLevelFunctions() async {
72    await getDevices();
73  }
74
75  @override
76  void initState() {
77    super.initState();
78    print('init state occurred');
79
80    highLevelFunctions();
81
82    _hiveDB.createActivity(name: widget.nameOfUser);
83
84    _stopWatchTimer = StopWatchTimer(
85      mode: StopWatchMode.countUp,
86      onChange: (value) {
87        displayTime.value = StopWatchTimer.getDisplayTime(value);
88      },
89    );
90
91    _stopWatchTimer.onExecute.add(StopWatchExecute.start);
92
93    //loop that will fetch the sensor info every 60Hz
94    _timer = Timer.periodic(
95      const Duration(milliseconds: 17),
96      (timer) {
97        if (
98
99            // check device 1
100           accelerometerCharacteristic1 != null &&
101               gyroscopeCharacteristic1 != null &&
102               magnetometerCharacteristic1 != null
103
104               // check device 2
105
106               &&
107               accelerometerCharacteristic2 != null &&
108               gyroscopeCharacteristic2 != null &&
109               magnetometerCharacteristic2 != null) {
110         // read characteristics from device 1
111
112         accelerometerCharacteristic1!.read();
113         gyroscopeCharacteristic1!.read();
114         magnetometerCharacteristic1!.read();
115
116         // read characteristics from device 2
117
118         accelerometerCharacteristic2!.read();
119         gyroscopeCharacteristic2!.read();
120         magnetometerCharacteristic2!.read();
121
122         _hiveDB.updateActivityWithMovement(
123           name: widget.nameOfUser,
```

```
            movement: Movement()
              ..timestamp = DateTime.now()
              ..accX = accelerometerValues1[0]
              ..accY = accelerometerValues1[1]
              ..accZ = accelerometerValues1[2]
              ..gyrX = gyroscopeValues1[0]
              ..gyrY = gyroscopeValues1[1]
              ..gyrZ = gyroscopeValues1[2]
              ..magX = magnetometerValues1[0]
              ..magY = magnetometerValues1[1]
              ..magZ = magnetometerValues1[2]
              ..accX2 = accelerometerValues2[0]
              ..accY2 = accelerometerValues2[1]
              ..accZ2 = accelerometerValues2[2]
              ..gyrX2 = gyroscopeValues2[0]
              ..gyrY2 = gyroscopeValues2[1]
              ..gyrZ2 = gyroscopeValues2[2]
              ..magX2 = magnetometerValues2[0]
              ..magY2 = magnetometerValues2[1]
              ..magZ2 = magnetometerValues2[2],
          );
          initProcessComplete.value = true;
        } else {
          print(
            'something was null: acc--> $accelerometerCharacteristic1 mag--> $magnetometerCharacteristic1 gyr--> $gyroscopeCharacteristic1',
          );
        }
      },
    );

    print('reached end of init state method');
  }

  @override
  Future<void> dispose() async {
    super.dispose();
    await _stopWatchTimer.dispose();
    _timer.cancel();
  }

  List<int> _getRandomBytes() {
    final math = Random();
    return [
      math.nextInt(255),
      math.nextInt(255),
      math.nextInt(255),
      math.nextInt(255)
    ];
  }

  List<Widget> _buildServiceTiles(List<BluetoothService> services) {
    return services
        .map(
          (s) => ServiceTile(
            service: s,
            characteristicTiles: s.characteristics
                .map(
                  (c) =>
                      // if (c == )
                      CharacteristicTile(
```

```
                    characteristic: c,
                    onReadPressed: () => c.read(),
                    onWritePressed: () async {
                      await c.write(_getRandomBytes(), withoutResponse: true);
                      await c.read();
                    },
                    onNotificationPressed: () async {
                      await c.setNotifyValue(!c.isNotifying);
                      await c.read();
                    },
                    descriptorTiles: c.descriptors
                        .map(
                          (d) => DescriptorTile(
                            descriptor: d,
                            onReadPressed: () => d.read(),
                            onWritePressed: () => d.write(_getRandomBytes()),
                          ),
                        )
                        .toList(),
                  ),
                )
                .toList(),
          ),
        )
        .toList();
  }

  printDataAndUpdate(BluetoothCharacteristic bluetoothCharacteristic) async {
    print('trying to update value');
    await bluetoothCharacteristic
        .setNotifyValue(!bluetoothCharacteristic.isNotifying);
    await bluetoothCharacteristic.read();
    print('updated value');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: Container(),
        centerTitle: true,
        backgroundColor: kYellow,
        title: Text(
          'Activity',
          style: TextStyle(
            fontSize: 30,
            color: kBlue,
            fontWeight: FontWeight.w600,
          ),
        ),
        actions: [
          IconButton(
            color: kBlue,
            onPressed: () => Get.dialog(
              AlertDialog(
                backgroundColor: kYellow,
                content: Text(
                  'Are you sure you want to exit the activity ?',
                  style: TextStyle(
                    color: kBlue,
                    fontSize: 25,
```

```
                    fontWeight: FontWeight.w400,
                  ),
                ),
                actions: [
                  TextButton(
                    onPressed: () async {
                      _hiveDB.updateActivityWithTimestamp(
                        name: widget.nameOfUser,
                        duration: displayTime.value,
                      );

                      _timer.isActive ? _timer.cancel() : null;

                      var flutterBlue = FlutterBlue.instance;
                      await flutterBlue.stopScan();
                      for (var device in await flutterBlue.connectedDevices) {
                        await device.disconnect();
                      }

                      Get.back(
                        closeOverlays: true,
                      );
                    },
                    child: Text(
                      'OK',
                      style: TextStyle(color: kBlue, fontSize: 20),
                    ),
                  ),
                  TextButton(
                    onPressed: () => Get.back(),
                    child: Text(
                      'Cancel',
                      style: TextStyle(color: kBlue, fontSize: 20),
                    ),
                  ),
                ],
              ),
            ),
            icon: const Icon(
              Icons.close,
            ),
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            const SizedBox(height: 20),

            // display time
            Center(
              child: Obx(
                () => Text(
                  displayTime.value,
                  style: const TextStyle(
                    color: Colors.white,
                    fontSize: 60,
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),
            ),
```

```
            ),

            const SizedBox(height: 20),
            const Text(
              'Device 1',
              style: TextStyle(color: Colors.white, fontSize: 30),
            ),
            const SizedBox(height: 20),

            //display sensor info
            StreamBuilder<List<BluetoothService>>(
              stream: deviceList.first.services,
              initialData: [],
              builder: (c, snapshot) {
                if (snapshot.data != null &&
                    snapshot.data!.isNotEmpty &&
                    snapshot.data![3].characteristics.isNotEmpty) {
                  // set characteristics
                  accelerometerCharacteristic1 =
                      snapshot.data![3].characteristics[0];

                  gyroscopeCharacteristic1 =
                      snapshot.data![3].characteristics[1];

                  magnetometerCharacteristic1 =
                      snapshot.data![3].characteristics[2];

                  return Column(
                    children: [
                      StreamBuilder<List<int>>(
                        stream: snapshot.data![3].characteristics[0].value,
                        builder: (ctx, newerSnapshot) {
                          if (newerSnapshot.data != null) {
                            accelerometerValues1 =
                                convertFromBinary(newerSnapshot.data!);

                            return Text(
                              'Acc: ' + accelerometerValues1.toString(),
                              style: const TextStyle(
                                color: Colors.white,
                                fontSize: 30,
                                fontWeight: FontWeight.w200,
                              ),
                            );
                          }
                          return CircularProgressIndicator();
                        },
                      ),
                      StreamBuilder<List<int>>(
                        stream: snapshot.data![3].characteristics[1].value,
                        builder: (ctx, newerSnapshot) {
                          if (newerSnapshot.data != null) {
                            gyroscopeValues1 =
                                convertFromBinary(newerSnapshot.data!);

                            return Text(
                              'Gyro: ' + gyroscopeValues1.toString(),
                              style: TextStyle(
                                color: Colors.white,
                                fontSize: 30,
                                fontWeight: FontWeight.w200,
```

```
                            ),
                          );
                        }
                        return CircularProgressIndicator();
                      },
                    ),
                    StreamBuilder<List<int>>(
                      stream: snapshot.data![3].characteristics[2].value,
                      builder: (ctx, newerSnapshot) {
                        if (newerSnapshot.data != null) {
                          magnetometerValues1 =
                              convertFromBinary(newerSnapshot.data!);

                          return Text(
                            'Mag: ' + magnetometerValues1.toString(),
                            style: TextStyle(
                              color: Colors.white,
                              fontSize: 30,
                              fontWeight: FontWeight.w200,
                            ),
                          );
                        }
                        return CircularProgressIndicator();
                      },
                    ),
                  ],
                );
              }

              return const Text('Loading');
            },
          ),

          const SizedBox(height: 50),
          const Text(
            'Device 2',
            style: TextStyle(color: Colors.white, fontSize: 30),
          ),
          const SizedBox(height: 20),
          StreamBuilder<List<BluetoothService>>(
            stream: deviceList.last.services,
            initialData: [],
            builder: (c, snapshot) {
              if (snapshot.data != null &&
                  snapshot.data!.isNotEmpty &&
                  snapshot.data![3].characteristics.isNotEmpty) {
                // set characteristics
                accelerometerCharacteristic2 =
                    snapshot.data![3].characteristics[0];

                gyroscopeCharacteristic2 =
                    snapshot.data![3].characteristics[1];

                magnetometerCharacteristic2 =
                    snapshot.data![3].characteristics[2];

                return Column(
                  children: [
                    StreamBuilder<List<int>>(
                      stream: snapshot.data![3].characteristics[0].value,
                      builder: (ctx, newerSnapshot) {
```

```
                        if (newerSnapshot.data != null) {
                          accelerometerValues2 =
                              convertFromBinary(newerSnapshot.data!);

                          return Text(
                            'Acc: ' + accelerometerValues2.toString(),
                            style: const TextStyle(
                              color: Colors.white,
                              fontSize: 30,
                              fontWeight: FontWeight.w200,
                            ),
                          );
                        }

                        return CircularProgressIndicator();
                      },
                    ),
                    StreamBuilder<List<int>>(
                      stream: snapshot.data![3].characteristics[1].value,
                      builder: (ctx, newerSnapshot) {
                        if (newerSnapshot.data != null) {
                          gyroscopeValues2 =
                              convertFromBinary(newerSnapshot.data!);

                          return Text(
                            'Gyro: ' + gyroscopeValues2.toString(),
                            style: TextStyle(
                              color: Colors.white,
                              fontSize: 30,
                              fontWeight: FontWeight.w200,
                            ),
                          );
                        }
                        return CircularProgressIndicator();
                      },
                    ),
                    StreamBuilder<List<int>>(
                      stream: snapshot.data![3].characteristics[2].value,
                      builder: (ctx, newerSnapshot) {
                        if (newerSnapshot.data != null) {
                          magnetometerValues2 =
                              convertFromBinary(newerSnapshot.data!);

                          return Text(
                            'Mag: ' + magnetometerValues2.toString(),
                            style: TextStyle(
                              color: Colors.white,
                              fontSize: 30,
                              fontWeight: FontWeight.w200,
                            ),
                          );
                        }
                        return CircularProgressIndicator();
                      },
                    ),
                  ],
                );
              }

              return const Text('Loading');
            },
```

```
            ),
          ],
        ),
      ),
    );
  }
}
```

9.3 models

9.3.1 user.dart

```dart
import 'package:hive/hive.dart';
import 'package:hive_intro/models/activity.dart';

part 'user.g.dart';

@HiveType(typeId: 0)
class User {
  @HiveField(0)
  String? name;

  @HiveField(1)
  int? age;

  @HiveField(2)
  int? height;

  @HiveField(3)
  double? weight;

  @HiveField(4)
  List<Activity> activities;

  //class constructor
  User({
    this.name,
    this.age,
    this.height,
    this.weight,
    required this.activities,
  });
}
```

9.3.2 users.g.dart

```dart
// GENERATED CODE - DO NOT MODIFY BY HAND
part of 'user.dart';

// **************************************************************************
// TypeAdapterGenerator
// **************************************************************************

class UserAdapter extends TypeAdapter<User> {
  @override
  final int typeId = 0;

  @override
```

```
  User read(BinaryReader reader) {
    final numOfFields = reader.readByte();
    final fields = <int, dynamic>{
      for (int i = 0; i < numOfFields; i++) reader.readByte(): reader.read(),
    };
    return User(
      name: fields[0] as String?,
      age: fields[1] as int?,
      height: fields[2] as int?,
      weight: fields[3] as double?,
      activities: (fields[4] as List).cast<Activity>(),
    );
  }

  @override
  void write(BinaryWriter writer, User obj) {
    writer
      ..writeByte(5)
      ..writeByte(0)
      ..write(obj.name)
      ..writeByte(1)
      ..write(obj.age)
      ..writeByte(2)
      ..write(obj.height)
      ..writeByte(3)
      ..write(obj.weight)
      ..writeByte(4)
      ..write(obj.activities);
  }

  @override
  int get hashCode => typeId.hashCode;

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is UserAdapter &&
          runtimeType == other.runtimeType &&
          typeId == other.typeId;
}
```

9.3.3 activity.dart

```
import 'package:hive/hive.dart';
import 'package:hive_intro/models/movement.dart';

part 'activity.g.dart';

@HiveType(typeId: 1)
class Activity {
  @HiveField(0)
  List<Movement> mov = [];

  @HiveField(1)
  late DateTime date;

  @HiveField(2)
  late String duration;

  @HiveField(3)
  late int attempt;
```

```

  @HiveField(4)
  late int scored;
}
```

9.3.4 activity.g.dart

```
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'activity.dart';

// **************************************************************************
// TypeAdapterGenerator
// **************************************************************************

class ActivityAdapter extends TypeAdapter<Activity> {
  @override
  final int typeId = 1;

  @override
  Activity read(BinaryReader reader) {
    final numOfFields = reader.readByte();
    final fields = <int, dynamic>{
      for (int i = 0; i < numOfFields; i++) reader.readByte(): reader.read(),
    };
    return Activity()
      ..mov = (fields[0] as List).cast<Movement>()
      ..date = fields[1] as DateTime
      ..duration = fields[2] as String
      ..attempt = fields[3] as int
      ..scored = fields[4] as int;
  }

  @override
  void write(BinaryWriter writer, Activity obj) {
    writer
      ..writeByte(5)
      ..writeByte(0)
      ..write(obj.mov)
      ..writeByte(1)
      ..write(obj.date)
      ..writeByte(2)
      ..write(obj.duration)
      ..writeByte(3)
      ..write(obj.attempt)
      ..writeByte(4)
      ..write(obj.scored);
  }

  @override
  int get hashCode => typeId.hashCode;

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is ActivityAdapter &&
          runtimeType == other.runtimeType &&
          typeId == other.typeId;
}
```

9.3.5 movement.dart

```dart
import 'package:hive/hive.dart';

part 'movement.g.dart';

@HiveType(typeId: 2)
class Movement {
  @HiveField(0)
  late DateTime timestamp;

  @HiveField(1)
  late int accX;

  @HiveField(2)
  late int accY;

  @HiveField(3)
  late int accZ;

  @HiveField(4)
  late int gyrX;

  @HiveField(5)
  late int gyrY;

  @HiveField(6)
  late int gyrZ;

  @HiveField(7)
  late int magX;

  @HiveField(8)
  late int magY;

  @HiveField(9)
  late int magZ;

  @HiveField(10)
  late int calories;

  @HiveField(11)
  late int heartRate;

  @HiveField(12)
  late int accX2;

  @HiveField(13)
  late int accY2;

  @HiveField(14)
  late int accZ2;

  @HiveField(15)
  late int gyrX2;

  @HiveField(16)
  late int gyrY2;

  @HiveField(17)
  late int gyrZ2;

```

```
  @HiveField(18)
  late int magX2;

  @HiveField(19)
  late int magY2;

  @HiveField(20)
  late int magZ2;
}
```

9.3.6 movement.g.dart

```
// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'movement.dart';

// **************************************************************************
// TypeAdapterGenerator
// **************************************************************************

class MovementAdapter extends TypeAdapter<Movement> {
  @override
  final int typeId = 2;

  @override
  Movement read(BinaryReader reader) {
    final numOfFields = reader.readByte();
    final fields = <int, dynamic>{
      for (int i = 0; i < numOfFields; i++) reader.readByte(): reader.read(),
    };
    return Movement()
      ..timestamp = fields[0] as DateTime
      ..accX = fields[1] as int
      ..accY = fields[2] as int
      ..accZ = fields[3] as int
      ..gyrX = fields[4] as int
      ..gyrY = fields[5] as int
      ..gyrZ = fields[6] as int
      ..magX = fields[7] as int
      ..magY = fields[8] as int
      ..magZ = fields[9] as int
      ..calories = fields[10] as int
      ..heartRate = fields[11] as int
      ..accX2 = fields[12] as int
      ..accY2 = fields[13] as int
      ..accZ2 = fields[14] as int
      ..gyrX2 = fields[15] as int
      ..gyrY2 = fields[16] as int
      ..gyrZ2 = fields[17] as int
      ..magX2 = fields[18] as int
      ..magY2 = fields[19] as int
      ..magZ2 = fields[20] as int;
  }

  @override
  void write(BinaryWriter writer, Movement obj) {
    writer
      ..writeByte(21)
      ..writeByte(0)
      ..write(obj.timestamp)
      ..writeByte(1)
```

```
      ..write(obj.accX)
      ..writeByte(2)
      ..write(obj.accY)
      ..writeByte(3)
      ..write(obj.accZ)
      ..writeByte(4)
      ..write(obj.gyrX)
      ..writeByte(5)
      ..write(obj.gyrY)
      ..writeByte(6)
      ..write(obj.gyrZ)
      ..writeByte(7)
      ..write(obj.magX)
      ..writeByte(8)
      ..write(obj.magY)
      ..writeByte(9)
      ..write(obj.magZ)
      ..writeByte(10)
      ..write(obj.calories)
      ..writeByte(11)
      ..write(obj.heartRate)
      ..writeByte(12)
      ..write(obj.accX2)
      ..writeByte(13)
      ..write(obj.accY2)
      ..writeByte(14)
      ..write(obj.accZ2)
      ..writeByte(15)
      ..write(obj.gyrX2)
      ..writeByte(16)
      ..write(obj.gyrY2)
      ..writeByte(17)
      ..write(obj.gyrZ2)
      ..writeByte(18)
      ..write(obj.magX2)
      ..writeByte(19)
      ..write(obj.magY2)
      ..writeByte(20)
      ..write(obj.magZ2);
  }

  @override
  int get hashCode => typeId.hashCode;

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is MovementAdapter &&
          runtimeType == other.runtimeType &&
          typeId == other.typeId;
}
```

9.4 widgets

9.4.1 constants.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

```

```
Color kYellow = const Color(0xffF8DC6A);
Color kOrange = const Color(0xffF29A42);
Color kBlue = const Color(0xff293D4F);

final ButtonStyle flatButtonStyle = TextButton.styleFrom(
  primary: Colors.black87,
  backgroundColor: kYellow,
  minimumSize: const Size(350, 60),
  padding: const EdgeInsets.symmetric(horizontal: 16.0),
  shape: const RoundedRectangleBorder(
    borderRadius: BorderRadius.all(
      Radius.circular(10.0),
    ),
  ),
);

final ButtonStyle orangeFlatButtonStyle = TextButton.styleFrom(
  primary: Colors.black87,
  backgroundColor: kOrange,
  minimumSize: const Size(350, 60),
  padding: const EdgeInsets.symmetric(horizontal: 16.0),
  shape: const RoundedRectangleBorder(
    borderRadius: BorderRadius.all(
      Radius.circular(10.0),
    ),
  ),
);

TextStyle titleStyle = TextStyle(
  fontWeight: FontWeight.bold,
  fontSize: 16,
  color: kBlue,
);
```

9.4.2 helper.dart

```

//function to convert the sensor readings to decimal values
List convertFromBinary(List<int> hexList) {
  print('here is the original list' + hexList.toString());

  //inverts the order of each pair of data to display values properly
  int firstValue = (hexList[0] + (hexList[1] * 256));
  int secondValue = (hexList[2] + (hexList[3] * 256));
  int thirdValue = (hexList[4] + (hexList[5] * 256));

  firstValue = readSignedInt(firstValue);
  secondValue = readSignedInt(secondValue);
  thirdValue = readSignedInt(thirdValue);

  return [
    (firstValue),
    (secondValue),
    (thirdValue),
  ];
}

//function to read the signed ints coded in 2's complement
int readSignedInt(m) {
  int value = m;

```

```
  //checks if this is a negative number
  if ((value & 0x8000) > 0) {

    value = value | 0xFFFFFFFFFFFF0000;
  }

  return value;
}
```

9.4.3 hive-db.dart

```
import 'package:hive/hive.dart';
import 'package:hive_intro/models/activity.dart';
import 'package:hive_intro/models/movement.dart';
import 'package:hive_intro/models/user.dart';

class HiveDB {
  // all boxes: users, individual user box

  // user stuff
  // low level api's
  addUserInList(String name) async {
    var users = await Hive.openBox<String>('users');
    users.add(name);
  }

  createUserBox(User user) async {
    var box = await Hive.openBox<User>(user.name!);

    box.add(
      User(
        name: user.name!,
        age: user.age!,
        height: user.height!,
        weight: user.weight!,
        activities: [],
      ),
    );
  }

  /// Adds a new user and creates a new box
  void addNewUser(User user) {
    addUserInList(user.name!);
    createUserBox(user);
  }

  Future<void> updateUserNameInList({
    required int index,
    required String newName,
  }) async {
    var users = await Hive.openBox<String>('users');

    users.putAt(index, newName);

    print('updated in user list');
  }

  void updateUser(User _newUser, int index) async {
    var userBox = await Hive.openBox<User>(_newUser.name!);

    userBox.put(0, _newUser);
```

```
51
52      updateUserNameInList(index: index, newName: _newUser.name!);
53
54      print('update user function exited');
55    }
56
57    void trashABox(String name) async {
58      Hive.box('name').close();
59      await Hive.deleteBoxFromDisk(name);
60    }
61
62    /// enter name of user whose activity we want
63    Future<Activity?> getLastActivity(String name) async {
64      var userBox = await Hive.openBox<User>(name);
65
66      var user = userBox.getAt(0);
67
68      if (user!.activities.isEmpty) {
69        return null;
70      } else {
71        return user.activities.last;
72      }
73    }
74
75    Future<User> returnUserFromAName(String userName) async {
76      var userBox = await Hive.openBox<User>(userName);
77
78      print('name was $userName and got box $userName');
79
80      return userBox.getAt(0)!;
81    }
82
83    Future<void> createActivity({
84      required String name,
85      // required String deviceID,
86    }) async {
87      var userBox = await Hive.openBox<User>(name);
88
89      User user = userBox.getAt(0)!;
90
91      user.activities.add(Activity()..date = DateTime.now()
92          // ..deviceId = deviceID,
93          );
94    }
95
96    /// Update activity list with new movement
97    updateActivityWithMovement({
98      required String name,
99      required Movement movement,
100   }) async {
101     var userBox = await Hive.openBox<User>(name);
102
103     User user = userBox.getAt(0)!;
104
105     if (user.activities.isNotEmpty) {
106       Activity activity = user.activities.last;
107
108       activity.mov.add(movement);
109     } else {
110       print('no activity found');
111     }
```

```
}

updateActivityWithTimestamp({
  required String name,
  required String duration,
}) async {
  var userBox = await Hive.openBox<User>(name);

  User user = userBox.getAt(0)!;

  if (user.activities.isNotEmpty) {
    Activity activity = user.activities.last;

    activity.duration = duration;
    print('updated duration');
  } else {
    print('no activity found');
  }
}

/// [ballWentInHoop]: pass in true if ball goes into hoop, false by default.
/// Ensure String [name] of user is passed in.
void shotMade(String name, {bool ballWentInHoop = false}) async {
  var userBox = await Hive.openBox<User>(name);

  var user = userBox.getAt(0);
  Activity _activity = user!.activities.last;
  _activity.attempt++;

  ballWentInHoop ? _activity.scored++ : null;
}

/// Enter name of box to remove
removeUserBox(String name) async {
  await Hive.deleteBoxFromDisk(name);
  print('deleted $name');
}

removeActivity({
  required String name,
  required int index,
}) async {
  var userBox = await Hive.openBox<User>(name);

  var user = userBox.getAt(0);
  user!.activities.removeAt(index);
}

Future<void> printAll() async {
  // to see all boxes
  var box = await Hive.openBox<String>('users');
  print(box.values);
}

Future<void> printSomeUserStuff(String name) async {
  // to print specific boxes
  var box = await Hive.openBox<User>(name);

  var person = box.getAt(0);
  print(person!.weight);
}
```

```
}
```

9.4.4 ble-widgets.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_blue/flutter_blue.dart';

class ScanResultTile extends StatelessWidget {
  const ScanResultTile({Key? key, required this.result, required this.onTap})
      : super(key: key);

  final ScanResult result;
  final VoidCallback onTap;

  Widget _buildTitle(BuildContext context) {
    if (result.device.name.length > 0) {
      return Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Text(
            result.device.name,
            overflow: TextOverflow.ellipsis,
          ),
          Text(
            result.device.id.toString(),
            style: Theme.of(context).textTheme.caption,
          )
        ],
      );
    } else {
      return Text(result.device.id.toString());
    }
  }

  Widget _buildAdvRow(BuildContext context, String title, String value) {
    return Padding(
      padding: EdgeInsets.symmetric(horizontal: 16.0, vertical: 4.0),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Text(title, style: Theme.of(context).textTheme.caption),
          SizedBox(
            width: 12.0,
          ),
          Expanded(
            child: Text(
              value,
              style: Theme.of(context)
                  .textTheme
                  .caption!
                  .apply(color: Colors.black),
              softWrap: true,
            ),
          ),
        ],
      ),
    );
  }

  String getNiceHexArray(List<int> bytes) {
```

```
    return '[${bytes.map((i) => i.toRadixString(16).padLeft(2, '0')).join(', ')}]'
        .toUpperCase();
  }

  String? getNiceManufacturerData(Map<int, List<int>> data) {
    if (data.isEmpty) {
      return null;
    }
    List<String> res = [];
    data.forEach((id, bytes) {
      res.add(
          '${id.toRadixString(16).toUpperCase()}: ${getNiceHexArray(bytes)}');
    });
    return res.join(', ');
  }

  String? getNiceServiceData(Map<String, List<int>> data) {
    if (data.isEmpty) {
      return null;
    }
    List<String> res = [];
    data.forEach((id, bytes) {
      res.add('${id.toUpperCase()}: ${getNiceHexArray(bytes)}');
    });
    return res.join(', ');
  }

  @override
  Widget build(BuildContext context) {
    return ExpansionTile(
      title: _buildTitle(context),
      leading: Text(result.rssi.toString()),
      trailing: RaisedButton(
        child: const Text('CONNECT'),
        color: Colors.black,
        textColor: Colors.white,
        onPressed: (result.advertisementData.connectable) ? onTap : null,
      ),
      children: <Widget>[
        _buildAdvRow(
            context, 'Complete Local Name', result.advertisementData.localName),
        _buildAdvRow(context, 'Tx Power Level',
            '${result.advertisementData.txPowerLevel ?? 'N/A'}'),
        _buildAdvRow(
            context,
            'Manufacturer Data',
            getNiceManufacturerData(
                    result.advertisementData.manufacturerData) ??
                'N/A'),
        _buildAdvRow(
            context,
            'Service UUIDs',
            (result.advertisementData.serviceUuids.isNotEmpty)
                ? result.advertisementData.serviceUuids.join(', ').toUpperCase()
                : 'N/A'),
        _buildAdvRow(context, 'Service Data',
            getNiceServiceData(result.advertisementData.serviceData) ?? 'N/A'),
      ],
    );
  }
}
```

```

class ServiceTile extends StatelessWidget {
  final BluetoothService service;
  final List<CharacteristicTile> characteristicTiles;
  const ServiceTile(
      {Key? key, required this.service, required this.characteristicTiles})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    if (characteristicTiles.isNotEmpty) {
      return ExpansionTile(
        title: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            const Text('Service'),
            Text(
              '0x${service.uuid.toString().toUpperCase().substring(4, 8)}',
              style: Theme.of(context)
                  .textTheme
                  .bodyText2!
                  .copyWith(color: Theme.of(context).textTheme.caption!.color),
            )
          ],
        ),
        children: characteristicTiles,
      );
    } else {
      return ListTile(
        title: const Text('Service'),
        subtitle:
            Text('0x${service.uuid.toString().toUpperCase().substring(4, 8)}'),
      );
    }
  }
}

class CharacteristicTile extends StatelessWidget {
  final BluetoothCharacteristic characteristic;
  final List<DescriptorTile> descriptorTiles;
  final VoidCallback onReadPressed;
  final VoidCallback onWritePressed;
  final VoidCallback onNotificationPressed;

  const CharacteristicTile(
      {Key? key,
      required this.characteristic,
      required this.descriptorTiles,
      required this.onReadPressed,
      required this.onWritePressed,
      required this.onNotificationPressed})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
```

9.4.5 custom-form-widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
```

```dart

import '../constants.dart';

class CustomFormWidget extends StatelessWidget {
  const CustomFormWidget({
    Key? key,
    required this.controller,
    required this.icon,
    this.willAcceptNumbersOnly = true,
    required this.content,
  }) : super(key: key);

  final String content;
  final IconData icon;
  final bool  willAcceptNumbersOnly;
  final TextEditingController controller;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
      child: TextFormField(
        inputFormatters: willAcceptNumbersOnly
            ? [FilteringTextInputFormatter.digitsOnly]
            : null,
        keyboardType:
            willAcceptNumbersOnly ? TextInputType.number : TextInputType.text,
        controller: controller,
        decoration: InputDecoration(
          filled: true,
          fillColor: kYellow,
          border: const OutlineInputBorder(
            borderRadius: BorderRadius.all(
              Radius.circular(10.0),
            ),
          ),
          focusedBorder: const OutlineInputBorder(
            borderSide: BorderSide(color: Colors.transparent),
            borderRadius: BorderRadius.all(
              Radius.circular(10.0),
            ),
          ),
          focusColor: kBlue,
          labelText: content,
          prefixIcon: Icon(icon),
        ),
        // The validator receives the text that the user has entered.
        validator: (value) {
          if (value == null || value.isEmpty) {
            return "Field cannot be empty";
          }
          return null;
        },
      ),
    );
  }
}
```

Part II

Solicitation document

Chapter 10

Solicitation document

This chapter will describe the conditions related to the realisation of the wearable prototype for the identification and evaluation of free throws in basketball.

10.1 Object

The product to be implemented consists of a wearable prototype, formed by one wearable device capable of obtaining the measurements to obtain the orientation of the arm of the player when shooting a free throw and one accelerometer sensor that captures the outcome of the shot, together with a mobile application for storing and processing the information and a MATLAB tool project for analysing the data.

10.2 Material requirements

For the realisation of the prototype, the wearable devices must be the Hexiwear from the manufacturer MikroElektronika. A mobile device with a Bluetooth version higher than 4.1 must also be available for Bluetooth Low Energy communication and an Android version higher than 7.0 or iOS above 11.0 for the correct functioning of the mobile application. On the other hand, it will be necessary to have a computer with sufficient specifications to be able to work with the MATLAB computing tool, to be able to obtain the data from the mobile device and to be able to work with Android Studio. If the user wishes to debug and test the app on an iPhone device, the personal computer shall be manufactured by Apple as well. Finally, it will be necessary to access a basketball court and to have a standardized ball to perform the shots.

10.3 Execution requirements

Once the materials are available, the wearable prototype can be built. To do this, the first thing to do is to check the correct functioning of the wearable device, so it is necessary to install the application provided by MikroElecktronika to visualise the data obtained by the device's sensors. Once the correct functioning of the device has been checked, the developed mobile application must be implemented to obtain the values collected by the wearable device via Bluetooth Low Energy and these values must be saved in a database so that they can later be sent to a computer as CSV files. Finally, a MATLAB project must be implemented to obtain the values from the CSV files and classify them according to the sensor from which the data originates. The MATLAB project must also be able to apply the Madgwick algorithm to the sensor values and the results obtained must be passed to navigation angles or also called Tait-Bryan for the representation of these. This process in MATLAB must be repeated for another movement, which will act as a reference movement, to obtain the degree of similarity between the movements through the correlation coefficient in two dimensions and, together with conditionals, identify and evaluate the movement based on the reference movement, by equalising the length of the samples of the two movements employing cross-correlation.

Concerning quality control, the ISO/IEC 20000 standard for the management of mobile application services and the ISO/IEC 27000 standard for the preservation of user information in the mobile application must be complied with.

10.4 Testing and service adjustments

Once the prototype has been released to the market, ISO/IEC 15504 should be followed for the possible improvement of the services offered by the mobile application. In addition, the identification and assessment of shots should be improved by carrying out more tests to obtain a higher percentage of success for the prototype.

Part III

Budget

Chapter 11

Budget

The purpose of this section is to establish the estimated costs that would be involved in carrying out this project.

11.1 Budget items

It should be noted that the project would be carried out by one person, with an estimated completion time of 6 months without necessarily following a continuous working day of 8 hours. This budget takes into account the costs of materials, software licences and labour to carry out the project.

Equipment costs refer to all material as well as computer equipment used in the realisation of the project. Concerning software licences, the budget has been carried out on the assumption that student licences are available, however, if this were not the case, the price would increase compared to the current total price, but the rise would not be critical. Regarding amortization, it has been considered of 20% for the hardware and 33% for the software and taking into account the hours dedicated to the project (250h) over the total hours of work in a year (estimating 1700h/year). The expression to obtain the cost value is the following

$$Cost = \frac{\%\, of\ amortization * Price * Number\, of\, hours}{Total\, work\, hours\, in\, a\, year} \tag{11.1}$$

Personnel costs are understood as the costs to be paid for employing human resources. In other words, it includes not only the cost of employees' salaries but also social security payments, insurance and other related expenses. For this project, it is necessary to hire a person with the qualifications of an industrial engineer, taking into account that this is a level 2 worker. In addition, a proportional percentage of 35% of the salary corresponding to the aforementioned personnel-related expenses has been added to the calculation.

Finally, a percentage of indirect costs (5%) has been added, which refers to unforeseen costs such as tariffs for materials or energy and maintenance and also the profit has been added, which refers to the percentage to make the project economically viable for implementation.

11.2 Bill of materials (BOM)

Ref	Unit	Description	Price (€)
h01	u.	Hexiwear	48.4
h02	u.	Hexiwear Blue Pack	18.15
h03	u.	Hexiwear strap (net sensor)	5
h04	u.	Apple iMac 2017 Retina 4K 21'5" HDD	1000
h05	u.	iPhone 11 128 GB	739
h06	u.	Basketball ball	15
s01	u.	Android Studio	0
s02	u.	Visual Studio Code	0
s03	u.	Xcode	0
s04	u.	MATLAB (student)	0
s05	u.	Microsoft Office (student)	0
s06	u.	Draw.io	0
s07	u.	Apple Developer membership (yearly)	99
e01	h.	Industrial technical engineer	13.16

11.3 Unitary prices

Ref	Unit	Description	Price (€)	Cost (€)	Quantity	Total (€)
h01	u.	Hexiwear	48.40	1.42	2	2.85
h02	u.	Hexiwear Blue Pack	18.15	0.53	1	0.53
h03	u.	Hexiwear strap (net sensor)	5.00	0.15	1	0.15
h04	u.	Apple iMac 2017 Retina 4K 21'5" HDD	1000.00	29.41	1	29.41
h05	u.	iPhone 11 128 GB	739.00	21.74	1	21.74
h06	u.	Basketball ball	15.00	0.44	1	0.44
s01	u.	Android Studio	-	-	1	-
s02	u.	Visual Studio Code	-	-	1	-
s03	u.	Xcode	-	-	1	-
s04	u.	MATLAB (student)	-	-	1	-
s05	u.	Microsoft Office (student)	-	-	1	-
s06	u.	Draw.io	-	-	1	-
s07	u.	Apple Developer membership (yearly)	99.00	4.80	1	4.80
e01	h.	Industrial technical engineer	13.16	13.16	250	3290.00
	%	Indirect costs	6250.25		5	312.51
		Total resources				**3662.43 €**

11.4 Budget summary

Concept	**Amount €)**
Total resources	3662.43
Benefits (20%)	732.49
Execution budget	4394.92
VAT (21%)	922.93
Total	**5317.85 €**

Bibliography

[1] */ Block diagram of the Kalman filter, which involves four steps: (1)... / Download Scientific Diagram.* URL: https://www.researchgate.net/figure/Block-diagram-of-the-Kalman-filter-which-involves-four-steps-1-predicting-current_fig2_335426887 (visited on 08/17/2021) (cit. on p. 27).

[2] *Automatic, Real-Time Basketball Stats and Analytics | ShotTracker.* URL: https://shottracker.com/ (visited on 08/11/2021) (cit. on p. 7).

[3] Scott Baker. "Classification of Common Basketball Actions using Player Tracking Data from the ShotTracker System". PhD thesis. University of Colorado at Boulder, 2020 (cit. on p. 7).

[4] Alejandro Barranco GutiÃ©rrez. "Ancho de banda experimental del movimiento de la muÃ±*ecadelamano*". In: Oct. 8, 2008 (cit. on p. 32).

[5] *Basketball Participation Report 2020.* URL: https://www.sfia.org/reports/807_Basketball-Participation-Report-2020 (visited on 08/10/2021) (cit. on p. 5).

[6] Alfred Benavent Pellicer. "Wearable de medida de rendimiento en deportes de raqueta". Accepted: 2019-10-17T07:16:20Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Oct. 17, 2019.

[7] Lionel Brits. *Euler angles, vector of File:Euler.png.* Jan. 9, 2008 (cit. on p. 25).

[8] BjÃ¶rn Eggert, Marion Mundt, and Bernd Markert. "IMU-BASED ACTIVITY RECOGNITION OF THE BASKETBALL JUMP SHOT". In: *ISBS Proceedings Archive* 38.1 (2020), p. 344 (cit. on pp. 7, 32).

[9] *ESP32 Bluetooth Low Energy (BLE) on Arduino IDE | Random Nerd Tutorials.* May 16, 2019. URL: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/ (visited on 08/16/2021) (cit. on pp. 23, 24).

[10] *EspaÃ±a, enelTop*10*depasesconmayorusodewearables, Formaci*^{3}*nyestudios|Interactiva*. InteractivaDigital.com. URL: https://interactivadigital.com/formacion-y-estudios-marketing-digital/espana-en-el-top-10-de-paises-con-mayor-uso-de-wearables/ (visited on 09/06/2021) (cit. on p. 9).

[11] *Figure 9 illustrates the principle of gimbal lock. The outer blue frame...* ResearchGate. URL: https://www.researchgate.net/figure/llustrates-the-principle-of-gimbal-lock-The-outer-blue-frame-represents-the-x-axis-the_fig4_338835648 (visited on 08/18/2021) (cit. on p. 25).

[12] *File:Taitbrianzyx.svg - Wikimedia Commons*. URL: https://commons.wikimedia.org/wiki/File:Taitbrianzyx.svg (visited on 08/18/2021) (cit. on p. 25).

[13] *Flutter - Beautiful native apps in record time*. URL: https://flutter.dev/ (visited on 08/15/2021) (cit. on p. 19).

[14] *flutter_blue | Flutter Package*. Dart packages. URL: https://pub.dev/packages/flutter_blue (visited on 09/04/2021) (cit. on p. 30).

[15] NÃ©stor GarcÃa GarcÃa. "DiseÃ±*odeundispositivowearableconfunci*^{3}*ndepod*^{3}*metrodeprecisi*^{3}*nparas*
Accepted: 2020-10-27T11:39:16Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Oct. 27, 2020.

[16] *Hexiwear*. MikroElektronika. URL: http://www.mikroe.com/hexiwear (visited on 08/12/2021) (cit. on pp. 10, 18, 19, 31).

[17] *hive | Dart Package*. Dart packages. URL: https://pub.dev/packages/hive (visited on 08/26/2021) (cit. on pp. 33, 34).

[18] *Jugadores de baloncesto federados por regiÃ*^{3}*nenEspa* ± *a*. Statista. URL: https://es.statista.com/estadisticas/1051652/baloncesto-numero-de-federados-en-espana-por-comunidad-autonoma/ (visited on 08/10/2021) (cit. on p. 5).

[19] Ruijie Ma et al. "Basketball movements recognition using a wrist wearable inertial measurement unit". In: *2018 IEEE 1st International Conference on Micro/Nano Sensors for AI, Healthcare, and Robotics (NSENS)*. IEEE, 2018, pp. 73–76 (cit. on pp. 7, 32).

[20] Sebastian O H Madgwick. "An eï¬*cientorientation*¬*lterforinertialandinertial/magneticsensorarrays*". In: (), p. 32 (cit. on pp. 26, 28, 38).

[21] Jonathan C. Maglott, Junkai Xu, and Peter B. Shull. "Differences in arm motion timing characteristics for basketball free throw and jump shooting via a body-worn sensorized sleeve". In: *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN). ISSN: 2376-8894. May 2017, pp. 31–34. DOI: 10.1109/BSN.2017.7936000 (cit. on p. 6).

[22] Sergi MeliÃ de la AsunciÃ³*n*. "Prototipo de dispositivo de medida de rendimiento en running basado en acelerÃ³*metrotriaxialycomunicaci*³*nadispositivom*³*vil*". Accepted: 2016-12-29T08:47:35Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Dec. 29, 2016.

[23] *Mobile OS market share 2021*. Statista. URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ (visited on 08/08/2021) (cit. on pp. 13–15).

[24] *MPU-9250 | TDK*. URL: https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/ (visited on 08/12/2021) (cit. on p. 13).

[25] *Quaternions â x-io Technologies*. URL: https://x-io.co.uk/quaternions/ (visited on 09/04/2021) (cit. on pp. 38, 39).

[26] JuliÃ¡n RamÃrez Espinal. "DiseÃ±*odeunwearableparadeportesdecontacto*". Accepted: 2016-12-21T08:34:08Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Dec. 21, 2016.

[27] Manju Rana and Vikas Mittal. "Wearable sensors for real-time kinematics analysis in sports: a review". In: *IEEE Sensors Journal* 21.2 (2020). Publisher: IEEE, pp. 1187–1207 (cit. on p. 7).

[28] Shreyank Shankar et al. "Performance measurement and analysis of shooting form of basketball players using a wearable IoT system". In: *2018 IEEE 8th International Advance Computing Conference (IACC)*. IEEE, 2018, pp. 26–32 (cit. on pp. 7, 32, 58).

[29] Esther Soriano Viguer. "Prototipo de dispositivo wearable de medida de rendimiento en deportes acrobÃ¡ticos basado en unidad de medida inercial y comunicaciÃ³*nadispositivom*³*vil*". Accepted: 2019-09-10T09:29:42Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Sept. 10, 2019.

[30] *STEVAL-WESU1 - Wearable sensor unit reference design for fast time to market - STMicroelectronics*. URL: https://www.st.com/en/evaluation-tools/steval-wesu1.html (visited on 06/24/2021) (cit. on p. 11).

[31] Matthew Straeten, Payman Rajai, and Mohammed Jalal Ahamed. "Method and implementation of micro Inertial Measurement Unit (IMU) in sensing basketball dynamics". In: *Sensors and Actuators A: Physical* 293 (July 2019), pp. 7–13. ISSN: 09244247. DOI: 10.1016/j.sna.2019.03.042 (cit. on p. 7).

[32] *Strategies to improve your free-throw shooting*. Winning Hoops. URL: https://winninghoops.com/article/strategies-to-improve-free-throw-shooting/ (visited on 08/11/2021) (cit. on p. 6).

[33] *TIDC-CC2650STK-SENSORTAG SimpleLinkâ¢ multi-standard CC2650 SensorTagâ¢ kit reference design | TI.com*. URL: https://www.ti.com/tool/TIDC-CC2650STK-SENSORTAG#1 (visited on 08/12/2021) (cit. on p. 12).

[34] Unknown. *AT THE LINE: Shooting The Traditional Free-Throw*. AT THE LINE. June 11, 2014. URL: https://fromthecharitystripe.blogspot.com/2014/06/shooting-traditional-free-throw.html (visited on 08/11/2021) (cit. on p. 7).

[35] Fabio Varesano. *Simple gravity compensation for 9 DOM IMUs | Varesano.net*. URL: http://www.varesano.net/blog/fabio/simple-gravity-compensation-9-dom-imus (visited on 09/04/2021) (cit. on p. 40).

[36] *Your First Bluetooth Low Energy App with Flutter*. URL: https://lupyuen.github.io/pinetime-rust-mynewt/articles/flutter#flutter-for-ios (visited on 08/10/2021) (cit. on p. 30).

[37] Javier Zamora GirbÃ©s. "Wearable de medida de rendimiento en deportes de contacto". Accepted: 2017-09-05T16:32:56Z. Proyecto/Trabajo fin de carrera/grado. Universitat PolitÃ¨cnica de ValÃ¨ncia, Sept. 5, 2017.