Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Master in Software Engineering

Master Thesis

Design of an IoT Monitoring System for the Old People at Home

Author: Juan Enrique Medina Miguelañez

Madrid, July 2023

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis
Master in Software Engineering

Title: Design of an IoT Monitoring System for the Elderly at Home
July 2023

Author: Juan Enrique Medina Miguelañez

Supervisor:

Elena Villalba Mora
Lenguajes y Sistemas Informaticos
e Ingeniería de Software

ETSI Informáticos
UPM

Co-supervisor:

Jaime Ramírez Rodríguez
Lenguajes y Sistemas Informaticos
e Ingeniería de Software

ETSI Informáticos
UPM

Abstract

Live expectancy has increased around the world. We live longer than those years before. However, even though live expectancy is larger than before, that does not mean we have increased as well our years with good health to enjoy these last years of our live. We need as society to make aware about improving life quality of these years and make sure they are not ruled by physical and mental decline.

One way of preventing this is to ensure inhabitants are aware of healthy ageing, which is promoting healthy habits such as good nutrition, be physically and mentally active. Nonetheless, although is important to maintain good habits, it is also important to develop test and factors that facilitate to predict and prevent the first signs of frailty and decline, to old with good health and better range of autonomy.

One of the most important problems at the time to develop these tests, is that they require the intervention of the geriatric professionals and caretakers to put them into action. That derives in a lot of time and, of course, great spent of money.

A solution could be home monitoring systems that are able to measure without intervention of professionals and caretakers. Along with the advantages of not having to force geriatricians to take measures, home monitoring systems can provide more regular measurements. Although they already exist some kind of sensors that can perform these tasks, such as pedometers, they lack some vital qualities, the most important, transparency and ubiquity.

ActiveUp Project was born from this necessity, to perform daily tasks during the day without noticing the different devices and sensors that can test the parameters and features of the patients.

In this document we approach the task of creating a wearable which can gather gait speed measurements and store them in an internal memory address to send them to the cloud, to allow a professional or a caretaker to observe the evolution of the patient.

Table of Contents

1 Introduction

1.1 Motivation

Nowadays healthy ageing is a current issue for society. We have increased our live expectancy, so we live longer than years and centuries before. However, that does not mean we have not increased the years with good health in proportion to this increase. It is necessary to remark that this increase must be followed by another increase in years when the older people are in good health, and not being overwhelmed by physical and mental decline.

The way to achieve these goals is to promote and keep functional capacity, to detect frailty and incapacity.
Therefore, it is essential to develop tests to predict the reasons for frailty to avoid or at least decrease the next step, which is incapacity. This allows us to age in better conditions and more independently, instead of being dependent on other people.

One of these inconveniences that this brings to us is the necessity of involvement from professionals and caretakers, which requires an increase in the health budget. To solve this problem, one of the solutions is to make their homes a test environment, with sensors and controllers that do not require an intervention of any professional, allowing them to keep their lifetime and not resulting in an increase of a spent of money.

Although there are already sensors that allow us to take measurements of some parameters, they lack some vital features such as insufficient clearance and ubiquity.

This master thesis is encompassed in a R&D project called ActiveUP. The aim of this project is to study the healthy ageing of older people based on specific parameters, such as gait speed, leg strength, weight loss or fatigue, during their daily lifetime, in a ubiquitous and transparent system, focusing on the gait speed in this case. This project proposes, as part of a home monitoring system, a wearable including accelerometers attached to the patient as a good approach for tracking signals from the patient in a transparent, ubiquitous, and personalized way.

For tracking these signals, we will focus on the different parts what we will need: the collection of signals from the sensors and transforming them into different data that is useful for us.

1.2 Objectives and goals

The main goal of this master thesis is to develop a wearable that can monitor the movement of people, to discover when they show signs of ill health or frailty, to avoid the next step, which is incapacity.

We must fulfil the following features for the wearable:

- The system must provide a service that does not forbid the patient to perform its daily lifetime.
- It must track all information in a period predefined in the requirements.
- The experts, such as doctors or professional caregivers, must be able to interpret these data to see the evolution of the patient, and if the patient begins to show the first signs of frailty.

1.3 Document structure

In this section we define the structure of the document to make a summary of the project:

The chapter 2 exposes the current situation of the field regarding the Ageing and Health, along with the main gadgets and tools used to reach our goals in the project.

The chapter 3 is an overview of the past and current projects for monitoring old people from a remote position, and what upgrades and improvements each project has been adding to the previous project, including the architecture of ActiveUP, which is the project we explain in this memory.

The chapter 4 is a detailed specification of requirements, including Functional Requirements and Non-Functional Requirements

The chapter 5 gives the reader an overall idea of the details of each component of the system. It also includes the software development of the main device, the Wearable, and each of the classes we implemented for it.

The chapter 6 regards to the testing of the device, including the unitary testing of each function, method and class, and the functional testing, which comprises the fulfilment of every Functional Requirement we defined in section 4.

The chapter 7 includes the deployment of the project in retirement homes with possible future patients and individual homes.

The chapter 8 includes an overview of the projects, highlighting the strong and weak points of the product and gathering the general opinions and suggestions from the users and patients, by quizzes and questionnaires.

The chapter 9 includes possible future projects and products, including commercialization of the product and possible upgrades considering the opinion of the users and patients.

The chapter 10 includes the bibliography and external tools such as links and web pages to obtain the information to make this document.

2 State of the Art

2.1 Healthy and active ageing

Ageing is a natural and gradual process of a living organism that leads to a higher risk of frailty, disease, and finally death. Over time living organisms suffer molecular and cellular damage that causes a decrease in physical and mental abilities. Though we can understand ageing as a natural process, it is not only connected with the passing of time. Many more factors have to do with this problem, such as genetics, lifestyle, chronic diseases, mental activity, familiar environment, or social support.

Therefore, the ageing concept is different for each one of us and is seen in a different way in any case. We can endure this ageing in a positive way if we are willing to take a healthy lifestyle. Healthy an active Ageing are models that try to focus in a better and more positive way.

- According to the OMS definition, Healthy Ageing is the process of promoting and encouraging the functional capacity that avoids allows comfort at ageing. It consists of maintaining healthy habits during the daily lifetime to reduce and delay the negative effects of ageing so that people can lead an active and independent life.
- According to the OMS definition, Active Ageing is the process of leverage having good physical, mental and social health. It consists of increasing years of life and their quality and diminishing the risk of incapacity. There are 3 main factors to consider for Active Ageing: Social Involvement for Personal Comfort, Health to delay dependence and Security to Ensure Protection.

These two models are well integrated with each other, as both give at the same time, they are ways to endure the ageing, and share the same goals to have more quality and live expectancy. They both approach the connection between physical and mental activity, health, and positive ageing.

Since the end of the last century, OMS has been working on developing good habits to promote these two models. To achieve this goal, it has not only to do with the involvement of older people, but the means that society possess for that. Thus, it is necessary to highlight that society must provide these means so older people can keep their lifetime with the restrictions that age and loss of abilities.

Figure 1 Frailty in the Old Population

Frailty can be defined as a physiologic state of increased vulnerability to stressors, resulting from decreased physiologic reserves or dysregulation of multiple physiologic systems. The prevailing opinion is that a physiological decline, associated with disease or age, which individually does not reach clinical importance, since it affects multiple systems and has a cumulative nature, can be detected as frailty. Frailty is pointed out by different authors as a strong predictor of disability, hospitalization, falls, loss of mobility and cardiovascular disease, with frail individuals being more vulnerable to adverse events [1].

Age is important to the health of old people, but is not the most relevant factor, for determining this. Functional capacity is more important. To determine this, we must have clear concepts of what is the frailty. Preventing is essential to avoid the most damaging effects of mental and physical decline along the ageing.

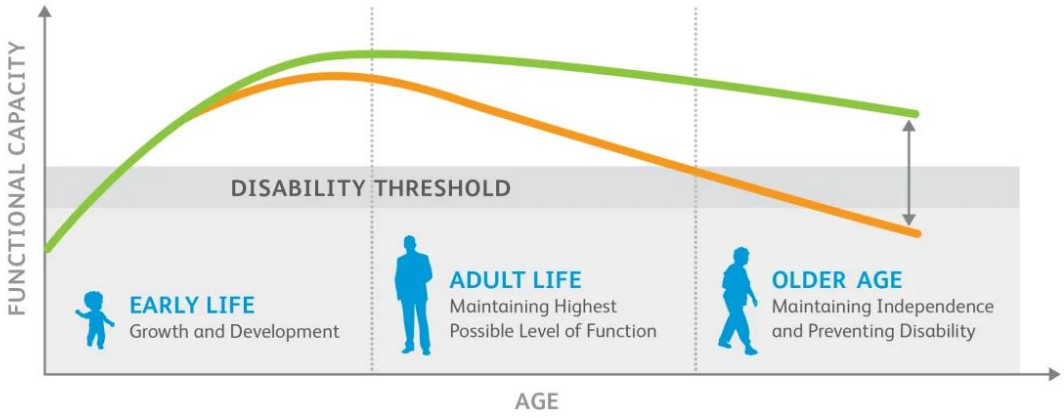


Figure 2 Frailty Evolution Along Lifetime

It is important to outline that frailty is a non-constant state, but volatile and with different directions. We can split the state of old people as 4 phases:

- Robustness: The patient can perform all activities of its lifetime without any support from external people.
- Prefrailty: The patient can perform most of the daily activities, but some of them need the support of external people.
- Frailty: The patient has difficulties performing most of its daily activities and needs the support of external people.

- Incapacity: The patient is completely dependent on external people.

In the robustness state we can prevent frailty. In the prefrailty state we can delay de frailty. In the frailty state we can prevent or at least delay the incapacity state.

According to Linda Fried, there is five main factors to classify from one status to another [2]. These factors are the following ones:

- Weight Loss.
- Exhaustion.
- Muscular fatigue.
- Low physical activity.
- Gait Speed Loss.

The listing of one state or another are the following:

- Robustness: The patient shows none of those signs.
- Prefrailty: The patient shows one or two signs.
- Frailty: The patient shows three or more signs.

Preventing these risk factors will allow ageing with better health, so it is important to practice physical activity and good nutrition. It is also important to prevent and delay frailty.

To delay, prevent or even revert the frailty, it is necessary to monitor the activity and the evolution of functional decline of each patient. This is part of the CGA, Comprehensive Geriatric Assessment, which is a diagnostic process aimed at old people to assess a monitoring and treatment plan.

Figure 3 Example of Tensiometer

One of the most common issues is that checking old people to detect signs of functional decline demands the intervention of professionals and caretakers and much time, resulting in the fact that most of the time it is eluded. Most of the monitoring process takes 6 months. However, 6 months is a long time, and it takes time to develop signs of frailty, so a more common monitoring would be much handier, and it helps detecting the first signs of fragility for each patient.

A useful tool to achieve this goal is the use of automatic sensors to track functional capacity within the home, because they do not require the

intervention of external people [3]. Besides, there are some tests to assess this, that have been evaluated and proved by professionals.

One of these tests consists in measure the gait speed by walking straight a determined distance. The most common distance is 4 m, but there is also 6 m, in case there is enough room to maneuver, or 2.4m in case there is not enough room. It works by measuring the time the patient takes by going across the distance between one side and another. Dividing the distance between the beginning and the end, by the time it takes to cross, we obtain the average gait speed.

Another test is called Sit-Stand-Sit, which consists of watching how many times the patient can sit and stand in a defined period. It is made for analyzing the strength of the legs and knees and the endurance [4].

Currently there are sensors able to measure the value of these functional variables of any patient to assess the functional capacity, as shown in Picture 4. However, one of the main inconveniences is that they are not so ubiquitous and transparent, which is one of the most indispensable conditions established. Here is the definition of ubiquitous and transparent:

- Transparent: When the sensor is able not to interfere with the daily lifetime of the patient.
- Ubiquitous: When the sensor can go unnoticed and belong to the environment.

The activities before, though consisting of doing exercise for the real lifetime, are performed manually and not automatically. They need to prepare the sensors and the application within the smartphone and follow instructions, which do not make them transparent and ubiquitous. There also exist some body sensors to control the activity of the daily lifetime, such as smartphones, tablets, pedometers, smart necklaces, electronic tags, but even though they are transparent, not all of them fulfil the condition of ubiquity.

One of the most popular devices consists of an IMU (Inertial Measurement Unit), which consists of an electronical device that measures and reports a specific force, angular rate and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes and sometimes even magnetometers. Accelerometers are considered one of the best ways to monitor the activity of a patient in a precise and objective way. It could be used for gathering gait speed, strength, physical activity level, movement, etc.

A home monitoring system is an ideal method to track the functional capacity of old people allowing them to live in better conditions for a longer period without needing to be assisted by a professional or a caretaker.

2.2 Technological concepts

2.2.1 IoT: MQTT

IoT (Internet of Things) is a concept that groups and interconnects devices and objects through a network, where all of them are visible and interact with one another. These objects could be anything, from sensors to daily home appliances, or even outfits. Everything can be connected to internet and work

without human intervention, so the goal is to manage a communication machine to machine or, better known, M2M [5].

Nowadays IoT is very popular because of all the possible applications and possibilities to improve the daily lifetime of people as well as business environments, where IoT was set a long time ago. For instance, one of these applications is the smart homes that are in use right now. Imagine that the weather is cold, a smart house would be able to activate central heating to warm the environment, making home much more comfortable without needing human intervention. Another popular use is in the industry (known as 4th Industrial Revolution), where is already applied in factories where devices and sensors connected to network allow to gather data and generate alarms and messages to send to different users to take necessary actions and activate protocols to intervene without human assistance, to correct and prevent them.

IoT is designed to help you easily implement the IoT protocols you need to communicate between networked devices. Many common protocols are supported, including AMQP, MQTT, MQTT-SN, STOMP, CoAP and more. All of them have the common features:

- Small Footprint: Each component is optimized for installation on small IoT-enabled devices.
- Uniform and Extensive Design: Very easy to use, with a uniform, intuitive, and extensible design. Common component interfaces across platforms and technologies.
- Fully Integrated Components: Native software components for any supported development technology - with no dependencies on external libraries.
- Blazing Fast Performance: Based on an optimized asynchronous socket architecture that has been actively refined for over three decades.
- Outstanding Technical Support: Backed by an expert team of support professionals. Free Email Support for everyone. Premium Support is also available for a fee.
- Additional Features: Detailed documentation, hundreds of sample applications, fully indexed help files, royalty-free licensing, and more.

Above all these protocols, we will focus on MQTT, which is the one we are going to use in this project.

MQTT is a messenger service based on standards, used to communicate from one device to another. Smart sensors and other devices from IoT usually must receive and transmit data from one restricted network with restricted bandwidth to another. These devices use MQTT as tool to transmit data, for it is an easy and efficient way to implement. MQTT has the following features and benefits:

- Light and Efficient: The implementation of MQTT on the IoT device requires minimal resources, so it can be used even on small microcontrollers. For example, a minimal MQTT control message might have as few as two bytes of data. MQTT message headers are also small to optimize network bandwidth.
- Scalable: The implementation of MQTT requires a minimal amount of code that consumes very little power in operations. The protocol

also has built-in features to support communication with many IoT devices. Therefore, you can implement the MQTT protocol to connect with millions of these devices.

- Reliable: Many IoT devices connect via unreliable cellular networks with low bandwidth and high latency. MQTT has built-in features that reduce the time it takes for the IoT device to reconnect with the cloud.
- Secure: MQTT makes it easy for developers to encrypt messages and authenticate devices and users using modern authentication protocols.
- Implemented: Several languages, such as Python, have extensive support for the implementation of the MQTT protocol. Therefore, developers can quickly implement it with minimal coding in any type of application [6].

MQTT protocol consists of publication and subscription model. According to classical network communication, clients and servers communicate directly with each other, with clients making requests and servers processing requests and sending responses. However, MQTT provides a publication and subscription to decouple the message remnant to the receiver. Instead of this, there is a third component, a messenger, which controls the communication between the publish and the subscribe component. The meaning of this messenger is filtering the entering messages from the publication part, ensuring the messages only for the corresponding subscription part, and connecting and disconnecting this publication and subscription parts.

MQTT protocol has the following components:

- Client: It could be any device, a sensor or a microcontroller that uses any MQTT library. If the client sends messages is a publisher, and if receives messages it is a subscriber.
- Agent: is the back-end service provider that handles the messages between different clients. The role of this service provider is to receive and filter these messages, identifying the subscribers and send those messages to them. It also handles the authentication of the clients, sends the messages to other systems to analyze, and recovers lost messages.
- Connection: is the link between the client and the agent. The client tries the connection with the agent and the agent responds with a successful or a failure message. Both require a TPC/IP heap to communicate [7].

Next, the functioning of the MQTT protocol could be the following:

1. A client can try to establish a connection with the MQTT agent.
2. Once connected, the client can publish some specific messages, subscribe to another specific message, or do both.
3. When the agent receives a message, it sends it back to the corresponding subscribers.

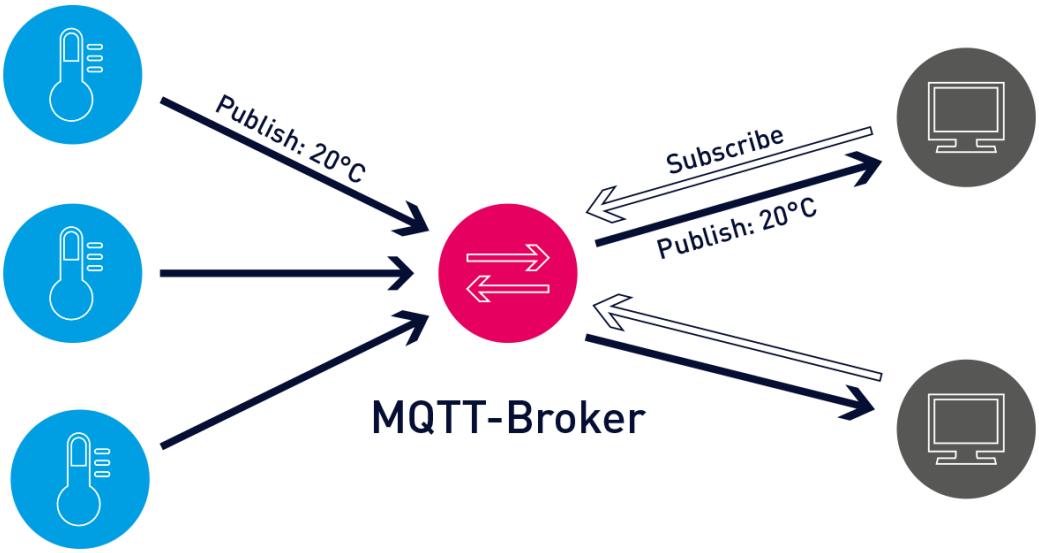


Figure 4 Brief schedule of MQTT Protocol

Now we analyze the details:

- Topic: is the key word that MQTT agents use by the time to sort messages to the different MQTT clients. Topics are hierarchically arranged, like a directory with different files. For instance, consider a smart building with different floors and these floors each have several homes. Each home has several rooms at the same time, and each room has different electronic devices or electrical appliances. One possible organization of the topics are the following:
  building/3rdFloor/5thDoor/kitchen/Glass_Ceramic
  building/7thFloor/2ndDoor/livingRoom/TV

- Publication: clients publish messages within the corresponding topic in a byte array format. The data message could be in different formats, such as text format, binary format, XML files or JSON files. For instance, it is possible that a temperature sensor for the living room publishes its data to the livingRoom topic.
- Subscription: clients send a request for a subscription to one specific topic. This request contains a unique identifier and a subscription list. For instance, one possible subscriber to the livingRoom topic, could be the air conditioner.

MQTT uses SSL (Secure Socket Layer) to protect secret data transmitted by MQTT devices. It can implement identity, authentication and authorization among clients and agents by SSL certificates and passwords. Normally MQTT agents authenticate clients by their passwords, as well as unique identifications assigned to each client.

2.2.2 Arduino

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Arduino

boards are available commercially from the official website or through authorized distributors [8].

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs [9].

2.2.2.1 Arduino IDE

The Arduino integrated development environment (IDE) is an cross-platform application that is written in the Java programming language. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus [10].

A new IDE named 2.0 was released as alpha version in October 2019, beta in March 2021 and a stable version in September 2022. This system still uses Arduino Command Line Interface, but new upgrades include more professional development environment, even a debugging system for tracking the code sentence by sentence, autocompletion support, just as other professional environments, such as Eclipse, NetBeans, or Visual Studio Code, to check possible syntax errors, and a Git support, in case we want to save the source code in a remote repository. It also includes a new board manager and a new library manager in case we need to search for another board and library on a remote web.

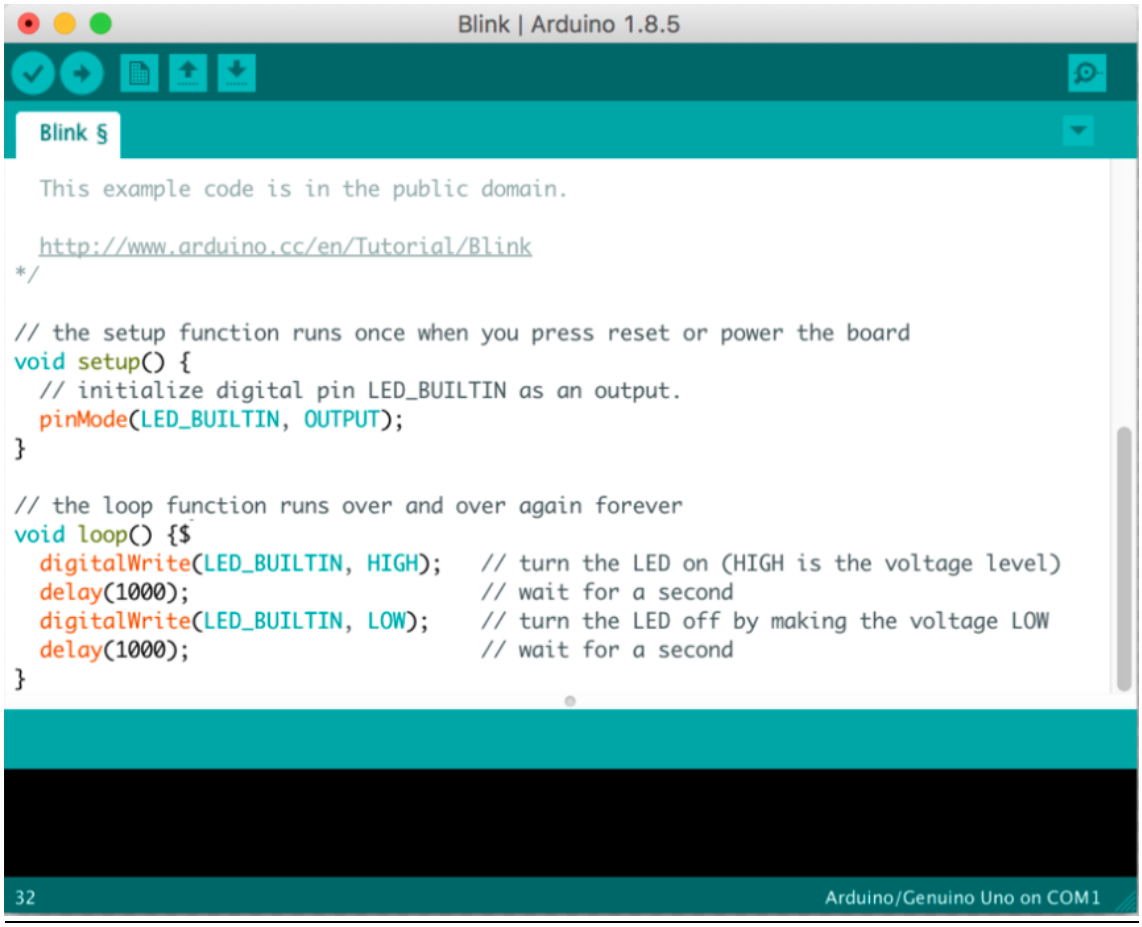


Figure 5 Development Interface for Arduino Project

2.2.2.2 Arduino Language

The Arduino IDE supports the languages C and C++using special rules of code structuring. The Arduino IDE supplies a software library for the wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU Toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the firmware of the board [11].

A minimal Arduino project consists of two main functions:

- Setup: This function is the first one to be called, when the sketch starts after a power-up or a reset event. It is used to initialized global and local variables, input and output pins and other basic libraries. It is analogous to the main function in the C or C++ project.
- Loop: After the setup function ends, loop function is repeated indefinitely in the main program, until a power-off or a reset event. A reset event could be either hardware or software, hardware events are triggered with a reset button pressed or a reset pit set as HIGH, while software events are forcing the execution sentence point at first direction of memory.

Here is an example of a simple program, that makes a LED blink every 0,5s:

```
const int LED = 2;

void setup() {

  pinMode(LED, OUTPUT);
}


void loop() {
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}
```

Figure 6 Example of Arduino Project (Blinking LED)

2.2.3 Communication protocols

2.2.3.1 Bluetooth

Bluetooth is an industrial specification for Wireless Personal Area Network (WPAN) designed to transmit files and data through a radio frequency link in the 2.4 GHz ISM Band. With this specification, the use of wires to communicate different devices is eliminated. The goals this specification pretends to achieve are the following:

- Ease the communication between devices.
- Eliminate wires and connectors.
- Allow to create personal Wireless Networks and ease synchronization between devices.

Versions:

- 1.X: First generation of Bluetooth devices (until 2003), the maximum power is 100mW or 20dBm and it has a maximum range of 100m. The Bandwidth is 1Mbps.
- 2.X: Second generation of Bluetooth devices (2004 to 2009), the maximum power is 5mW or 4dBm and it has a maximum range of 10m. The Bandwidth is 3Mbps.
- 3.X: Third generation of Bluetooth devices (2009 to 2010), the maximum power is 1mW or 0dBm and it has a maximum range of 3-5m. The Bandwidth is up to 24Mbps.
- 4.X: Fourth generation of Bluetooth devices (2010 to 2017), the maximum power is 0,5mW or -3 dBm and it has a maximum range of 1-3m. The Bandwidth is up to 32Mbps. Bluetooth Low Energy (BLE) is

a subset of this generation of Bluetooth devices, the main difference is that the consume is almost nothing when the device is not paired.
- 5.X: Fifth generation of Bluetooth devices (2017 until now), the maximum power is 0,5mW or -3 dBm and it has a maximum range of 1-3m. The Bandwidth is up to 50Mbps.

Hardware:

The components of a Bluetooth module are the following:

- Radio device: this is responsible for modulating and transmitting the signal.
- Digital controller: composed of a CPU, a Digital Signal Processor (DSP) called Link Controller (LC) and interfaces with the host device. Link Controller oversees processing Bandwidth, the Transference Function and coding voice record and data. The CPU handles Bluetooth instructions on the host device to simplify its operation. To do this, software called Link Manager runs on the CPU whose function is to communicate with other device through LPM protocol.

Protocols:

- LMP: Used for the establishment and control of the radio link between two devices, that is implemented in the controller.
- L2CAP: Used to multiplex multiple logical connections between two devices that use different high-level protocols, providing segmentation and reassembly of packets. It provides packet sizes up to 64kB.
- SDP: Used to allow a device to discover any other service offered by other devices and their associated parameters. Every service has its own unique identifier.
- RFCOMM: Used for generating a data flow. The data flow is like TFC.
- BNEP: Used for transferring data from other protocol heap through L2CAP. The main task is transmitting IP in a local network.
- AVCTP: Used for transferring audio and video control commands. It is used for audio and video distribution.
- TCS: Is the bit-oriented protocol that defines call control signalling for the establishment of voice and data calls between Bluetooth devices [12].

2.2.3.2 Wi-Fi

Wi-Fi is a family of wireless network protocols that are used for local area networking and internet access, allowing nearby devices to exchange radio data by radio waves.

Wi-Fi uses IEEE-802 Protocol family and is designed to work with ethernet. Lots of devices can work either wirelessly access point or with wired devices. They can work with two possible bandwidths, 2.4GHz and 5GHz, and they are subdivided into multiple channels.

The uses for the Wi-Fi protocols are:

- Internet: As we said before, Wi-Fi technology may be used to provide local network and Internet access to devices that are within Wi-Fi range of one or more routers that are connected to the Internet. Wi-Fi provides services in private homes, businesses, as well as in public spaces. Airports, hotels, restaurants, offices, universities and many others, often provides a Wi-Fi Network to provide needed services, often using a captive portal webpage for access. Similarly, battery-powered routers may include a cellular Internet radio modem and a Wi-Fi access point.
- Geolocation: Wi-Fi positioning systems use the positions of Wi-Fi hotspots to identify a device's location.
- Motion Detection: Wi-Fi sensing is used in applications such as motion detection and gesture recognition. [13]

Figure 7 A Huawei Dual Band 2.4 and 5GHz Router Wi-Fi

Wi-Fi stations communicate themselves by sending data packages. This, like any other radio signal, is done by modulating and demodulating signal waves. As other LAN devices, stations come programmed by a unique 6 bytes MAC address. The MAC addresses are used to specify both the destination and the source of each data packet. Wi-Fi also support multiple channels of communication, but every channel is only half-duplex, this does mean that every channel can either transmit or receive data, but not doing both things at the same time. In any case, the channels can be time-shared, this also means that multiple users are allowed to share the same channel by dividing the signals into multiple time slots.

The data frame is constructed in common fields, present always in every frame, and specific fields, present in specific cases. The common fields are always the Frame Control and the Frame Check Sequence (FCS). The first has a size of 2 octets and the last 4 octets. Other possible frames are the addresses, the Sequence Control, the QoS Control, the HT Control and the Frame Body.

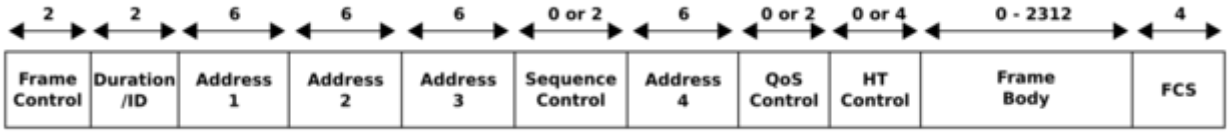


Figure 8 Wi-Fi Frame Control

The Frame Control has always 3 subfields within, which are the following: Protocol Version, the type and the subtype. Other possible subfields are:

- To-DS
- From-DS
- More-Fragments
- Retry
- Power Management
- More Data
- Protected Frame
- HTC/Order [14]

Generations:

- Wi-Fi 1 (1997): Radio Frequency 2.4 GHz, Maximum Rate: 1-2 Mbps
- Wi-Fi 2 (1999): Radio Frequency 2.4 GHz, Maximum Rate: 1-11 Mbps
- Wi-Fi 3 (2003): Radio Frequency 2.4/5 GHz, Maximum Rate: 6-54 Mbps
- Wi-Fi 4 (2008): Radio Frequency 2.4/5 GHz, Maximum Rate: 72-600 Mbps
- Wi-Fi 5 (2014): Radio Frequency 5 GHz, Maximum Rate: 433-6933 Mbps
- Wi-Fi 6 (2020): Radio Frequency 2.4/5 GHz, Maximum Rate: 574-9608 Mbps

2.2.3.3 UART

UART (Universal Asynchronous Receiver-Transmitter), more known as Serial Communication, is a communication protocol based on an integrated circuit used for serial communication between a central microcontroller or microprocessor and a peripheral. The hardware structure is very simple, it is composed only of two pins: receiver pin (Rx) and transmitter pin (Tx). The receiver pin of one device is connected to the transmitter of the other device, and the same goes for the opposite [15].

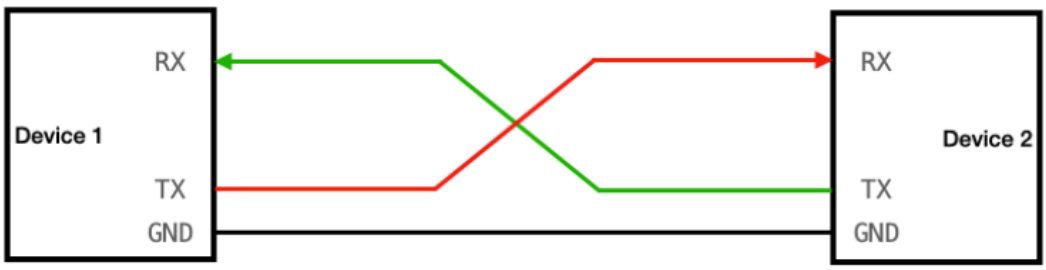


Figure 9 Schedule of UART Connection

The UART protocol takes bits of data and group them into packages of serial messages. The most common size of the packages is 10 bits, 7-8 bits are used for the data, one bit is used for the start signal and the last 1-2 bits are used for the stop package signal. It could be also a parity bit. The transmission begins when the signal pass from HIGH (idle) to LOW (start) state [16].

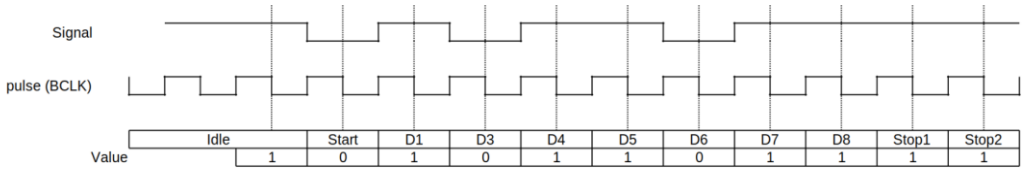


Figure 10 Example of UART Frame

The Serial Communications could be divided into two possible options:

- Half-Duplex: The device can receive or transmit, but it cannot do both things at the same time. This is the most common case.
- Full-Duplex: The device can receive and transmit at the same time.

Other important aspects of the UART Serial Port Communication are:

- Baud Rate: The common unit of measurement, which is used to set the speed of communication. Generally, are 9600 bauds (or bps).
- Data Bit Size: The size of the frame, normally is from 5 to 9 bits.
- Stop Bit Size: The size of the stop, normally is 1 or 2 bits.

2.2.3.4 I2C

I2C, also known as Inter-Integrated Circuit, is a serial and synchronous communication protocol based on package receiving and transmission among one master and several slaves, each one of them represented with a unique identifier.

I2C is widely used for the communication between processors and microcontrollers, and peripherals within a short distance. It is appropriate for peripherals where simplicity and low manufacturing costs are the most relevant features that a circuit must fulfil [17].

One of the most common applications of I2C design are:

- System management for PC systems via SMBus.
- Accessing real-time clocks and NVRAM chips that keep user settings.
- Accessing low-speed DACs and ADCs.
- Changing backlight, contrast, hue, colour balance settings etc in monitors (via Display Data Channel).
- Turning on and off the power supply of system components.
- Reading hardware monitoring and diagnostic sensors.
- Changing sound volume in intelligent speakers.
- Controlling small LCD or OLED displays.

Design

I2C is composed of two bidirectional wires: Serial Data Line (SDA) and Serial Clock Line (SCL or SCK). It is often to find attached to these wires two pull-up resistors connected to HIGH, to maintain a well-defined state when the circuit is static. The different components are connected to each wire for transmission.

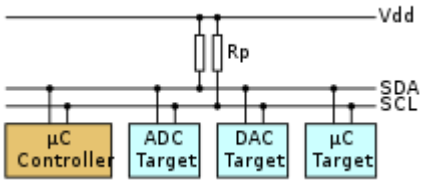


Figure 11 Basic Schedule of a I2C Functioning

The master handles the SCL wire, generating the signal needed to make each slave function. Before starting to work, the master maintains both SDA and SCL signals at HIGH value. Only when the master sets the SDA wire to LOW, the SCL wire activates, and the communication begins.

The first package is always referred to the I2C slave address, and the size could be between 7 and 10 bits, depending on the format. The following bit is corresponding to read or write command, normally read is set to LOW, and write set to HIGH. The address is sent to each slave, and the slave corresponding to the unique ID is the only to respond with the acknowledge bit.

After acknowledge bit set as HIGH, the following package is corresponding to the memory address, and again the acknowledge bit set as HIGH, and finally the data package, followed by the stop condition [18].

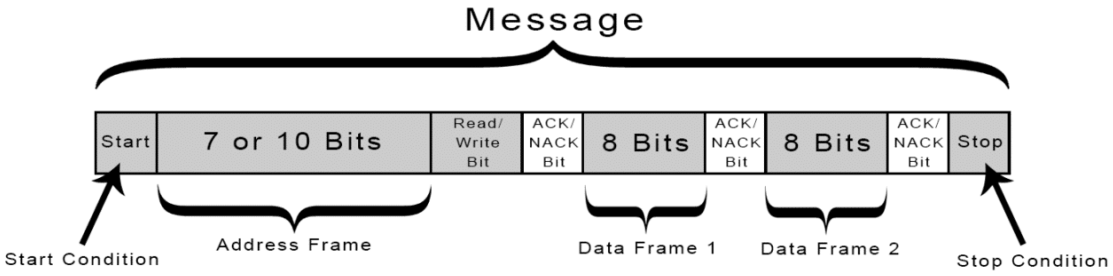


Figure 12 I2C Main Data Frame (1)

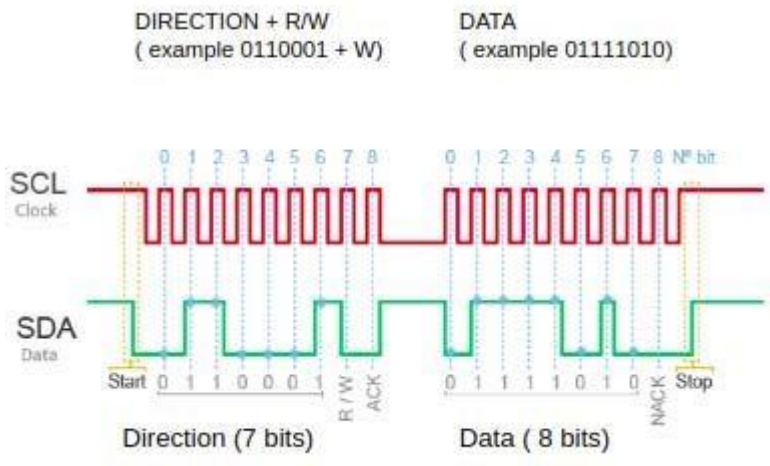


Figure 13 I2C Main Data Frame (2)

Modes:

- Standard Mode (Sm): 100 kbps
- Fast Mode (Fm): 400 kbps
- Fast Mode Plus (FM+): 1Mbps
- High-Speed Mode: (HM): 1,7-3,4 Mbps
- Ultra-Fast Mode: (UFm): 5 Mbps

Versions:

- Version 0: 100 kbps created.
- Version 1: Added 400 kbps fast-mode and addresses up to 10 bits.
- Version 2: Added 3,4 Mbps High-Speed Mode.
- Version 3: Added 1 Mbps Fast-Mode plus.
- Version 5: Added 5 Mbps Ultra-Fast-Mode unidirectional. From Version 5 onwards there are not significant changes.

2.2.3.5 SPI

SPI, also known as Serial Peripheral Interface, is a synchronous serial communication protocol designed for embedded systems to communicate processor or microcontroller with peripherals, usually in a short distance.

Like I2C, SPI also works with master-slave communication protocol, communicating master (usually a microcontroller or a processor) with different slaves (usually sensors, but also another controller or processor).

The main difference between I2C and SPI lays in the structure of the wires and protocols. SPI has a SCLK wire, corresponding to the clock, same as I2C, but SDA wire of I2C is substituted by two extra wires, MISO (Master Input Slave Output) and MOSI (Slave Input Master Output), and one wire per each additional slave attached to the master. That means, the formula to find the total of wires needed is:

$$nºwires = nºslaves + 3$$

Figure 14 SPI Scheme for One Slave

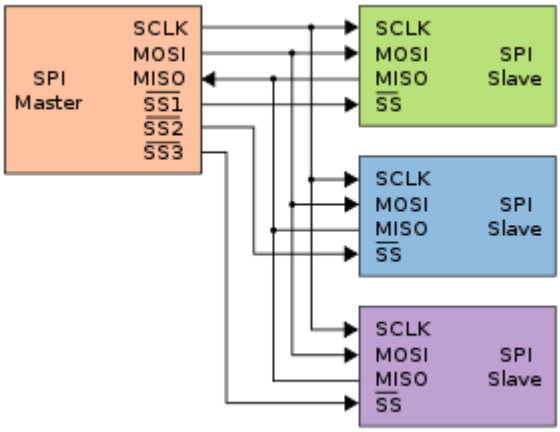


Figure 15 SPI Scheme for Three Different Slaves

The SCLK, MISO and MOSI wires are common for all the slaves. Besides, there is one SS wire per each slave, and every one of them is connected to one and only one of them. With this configuration, we ensure that the master has only one possible slave for each frame of data [19].

Regarding I2C, SPI has multiple advantages, the speed is significantly superior, there is no need for a frame for direction because the SS wire is in charge for that, and the rising edges are much minor. On the other side, SPI requires much more wires, and the layout could be much more complex, if there are many slaves connected to the master.

There are four different configurations based on two parameters of the signal clock: the Polarity (CPOL) and the Phase (CPHA):

- CPOL=0 and CPHA=0: Mode in which the clock state remains logic low, and information is sent on every transition from low to high, i.e., active high.
- CPOL=0 and CPHA=1: Mode in which the clock state remains logic low, and information is sent on every transition from high to low, i.e., active low.
- CPOL=1 and CPHA=0: A mode in which the clock state remains logic high, and information is sent on every transition from low to high, i.e., active high.
- CPOL=1 and CPHA=1: Mode in which the clock state remains logic high, and information is sent on every transition from high to low, i.e., active low [20].

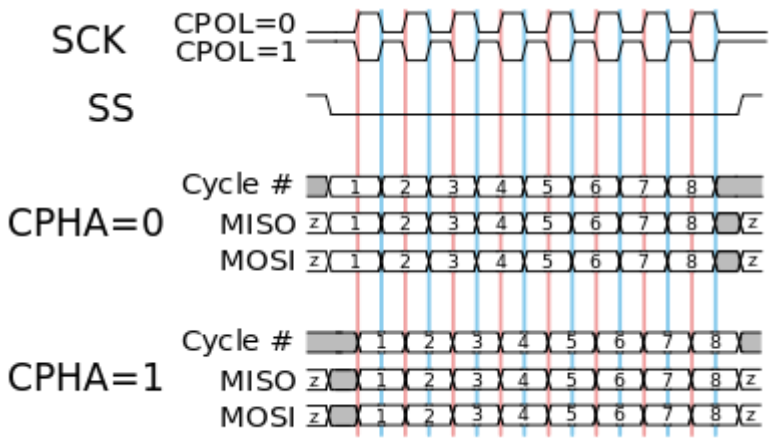


Figure 16 SPI Communication Protocol

3 Antecedents

3.1 Facet

The project consists of a monitoring system based on parameters that imitates the evaluation within conditions of usual clinical practice from professionals and caretakers. It is divided into four different elements that form the complete system, consisting of a Tablet, a Gait Speed folding stick, a Chair Stand support machine, and a wireless scale.

Figure 17 Complete Scheme of FACET Devices. It is composed of a Tablet (A), a Gait Speed Folding Stick (B), a Chair Stand Support Machine (C) and a Wireless Scale (D)

The tablet functioning consists of an app from where we can monitor the rest of the devices, based on the corresponding exercises for each week.
The Gait Speed Folding Stick (B) is a device made for measuring one of the most significant parameters in the evaluation of physical performance of old patients, which is Gait Speed. It is proved that there is a relationship between the required time to do the task, and hospitalization, residence admission, mortality, low quality life, physical and cognitive functional impairment and fall risk.
The proposed device is a couple of ultrasonic sensors, separated 2.4M from one to another, joined by a stick or a ribbon, which makes easy to install and uninstall within a patient house. Ultrasonic sensors have the advantage that they can measure distance between the first object they find, by sending an ultrasonic wave that rebounds in the object and returns to the ultrasonic sensor. Knowing the speed of sound (340 m/s), we can calculate the distance between the sensor and the object with the time between the sensor launches a wave and the rebounded wave returns by the following formula:

$$distance = time\_btwn\_wave\_detection \cdot \frac{340}{2} \ (m)$$

The correct use of this device is putting in a straight hall, with no objects between the sensor and the wall, and completely stretched. The device includes a people detection system that allows to detect a patient when he or she walk straight up onto the sensors. Once a patient is detected in one sensor, the device calculates the time from this detection to the detection of the other sensor.

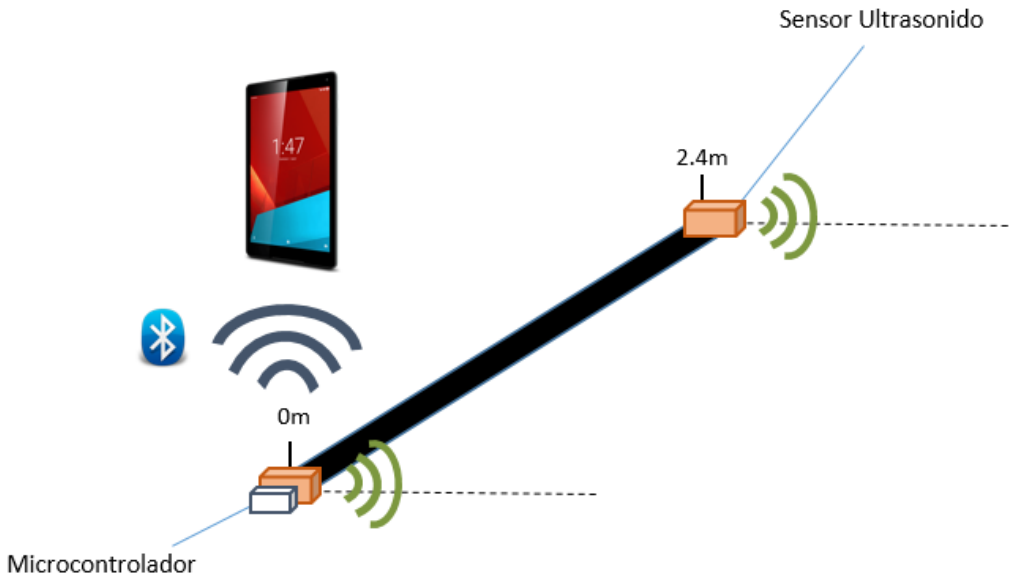


Figure 18 Gait Speed Functioning

To this point, determining the Gait Speed is easy, is a simple division of the distance between sensors and the time it takes to cross them, according to the following formula:

$$speed = \frac{space}{time} = \frac{2.4}{time}\ (m/s)$$

The Chair Stand Support Machine is a device meant for measuring another significant parameter in the evaluation of physical performance of old patients, which is the strength of lower limbs, based on the capacity of standing and sitting several times between a determined time. This is one of the most important tests and it is proved also that there is a huge relation between the capacity of subjects for standing and sitting from the chair with bigger success.

Figure 19 Chair Stand Functioning

The Chair Stand Device consists of an ultrasonic sensor, which is put onto a chair back, and is programmed to detect the distance between the back of the chair and the back of the patient. The device calculates the distance and the angle of the legs by an algorithm embedded in, and with the maximums and minimums can calculate how many cycles the patient can repeat, because the signals are almost periodical.

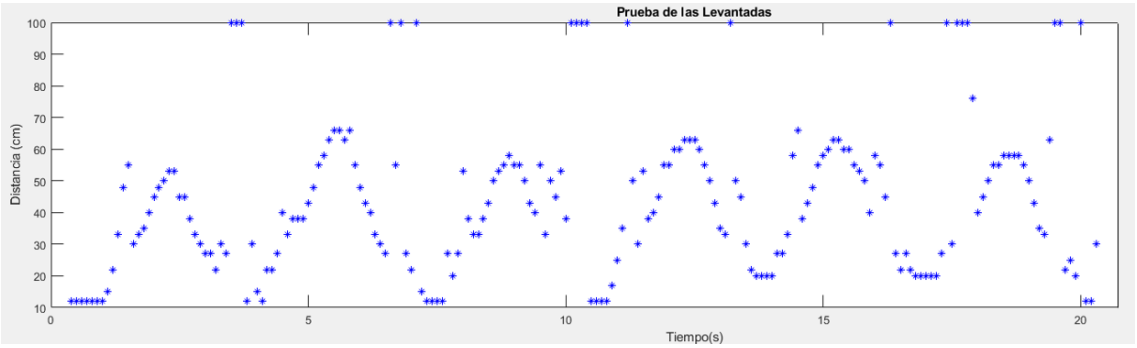


Figure 20 Chair Stand Signal Read from Device

The more cycles repeated or, in other words, the more frequency the signal possesses, the more stands the patient can perform in a single time.

Finally, the last device consists of a wireless scale. Weight loss is another relevant factor in the monitoring process of patients, being in fact one of the five criteria suggested by Linda Fried to assess whether a patient is fragile or not.

The functioning is simple, the patient gets on the scale and waits until his weight appears on the screen. Once the scale has measured the weight, the information is sent to the smartphone or tablet by Bluetooth.

Figure 21 Wireless Scale with Tablet

These three devices are connected to the smartphone or tablet by Bluetooth, so that they can send the relevant information. Therefore, the smartphone or tablet works as a data collector, as a patient platform to interact with the devices and as a source of information to the remote cloud. As we said before, the mobile device could be either smartphone or tablet, providing that it has Bluetooth and internet connection.

These mobile devices contain a software application that allows the patients to acquire independence and collect relevant information about fragility. On one side, they allow quick quizzes about parameters associated with fragility, such as the status today, how many falls the patient has suffered during the last days or the nutritional status, as many more. On the other hand, they act as a storing source, leveraging the data collected from the three sensors to process those data and transform them into relevant information that can be used by professionals and caretakers, using internet features, such as Wi-Fi or Data. Thus, the mobile device is set as the main tracking device, which allows the evaluation of the patient, and the strategy to follow based on the information collected by the sensors. For instance, according to the information received from the smartphone or tablet, as doctors, we can prescribe a new diet for the patient if we can see that this patient has lost too much weight during the last 2 weeks or the last month [21].

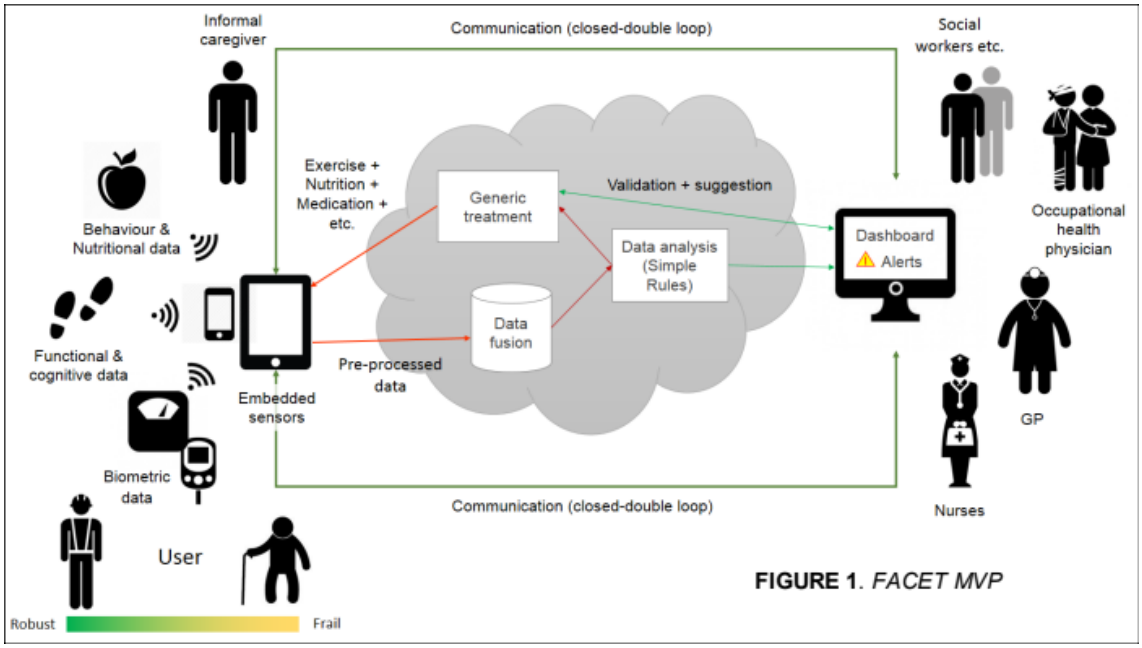


Figure 22 Facet Schedule

3.2 Positive

Positive projects consist of an evolution and upgrade of FACET, with the goals of providing a better diagnostic and remote monitoring of older adults, prevent disability among the older population, mitigate the social and clinical issues, detect possible events and threats, and involve patients in their own environment.

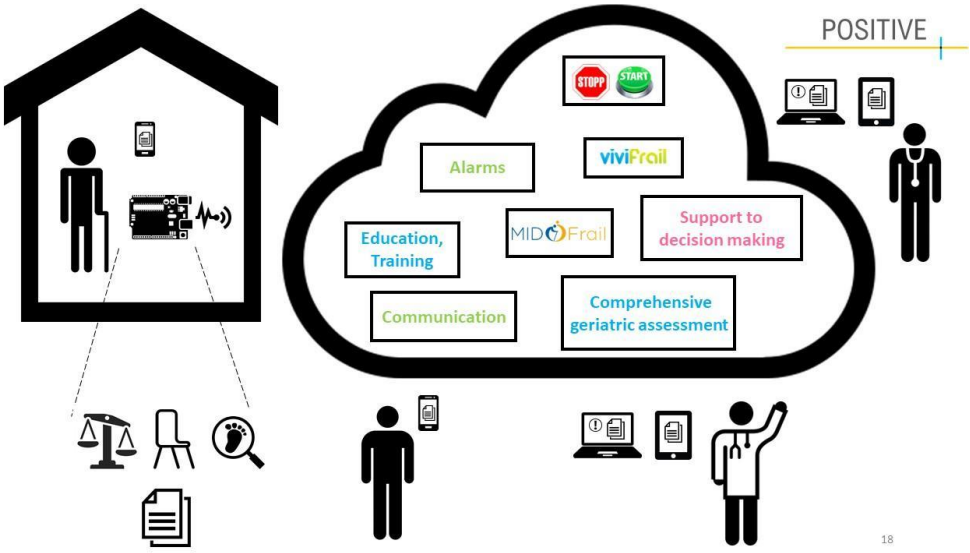


Figure 23 Positive Layout

As Facet, positive consist of several sensors installed within the home of the patient, along with the smartphone or tablet that contains the application responsible for taking all the data from the patients. The application already contains all algorithms needed to take the parameters [22].

In this project, the devices have suffered some modifications with respect of the previous project:

- The Gait Speed device has changed its form, substituting the previous stick for a more flexible chain, like the one for bicycles, but made of plastic.

Figure 24 Positive Gait Speed Device

The microcontroller is attached to one end, along with one ultrasonic sensor, and the other end contains the other ultrasonic sensor. Both ends are joined by a wire that attaches the other sensor with the microcontroller. This wire is covered by the chain and works as a 2.4m separator.

- The Chair Stand device is even more different than the previous version. Instead of using a chair to find the stand up and down, the new device contains an accelerometer and gyroscope and is attached to the leg, near the knee, reading constants angles from the patient to find how many stands can do in 30 seconds.

Figure 25 Positive Chair Stand Device

This device has 2 different ways of functioning: the calibration mode and the regular mode. The calibration mode finds the angles for the patient when he or she sits and stands. The normal mode measures how many stands and sits the patient can do.

- Besides having an application for smartphone or tablet, the system has a webpage, to allow the professionals and caretakers to review the patients and check the evolution of them.

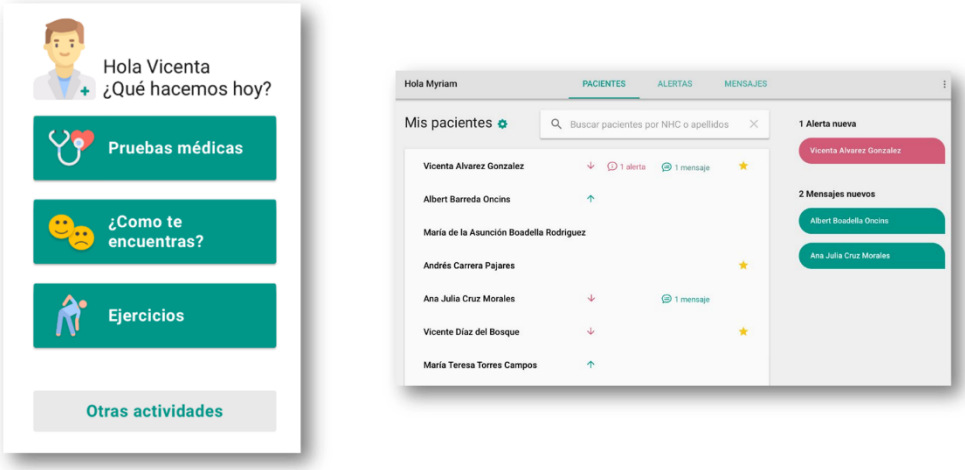


Figure 26 Application and Web Page

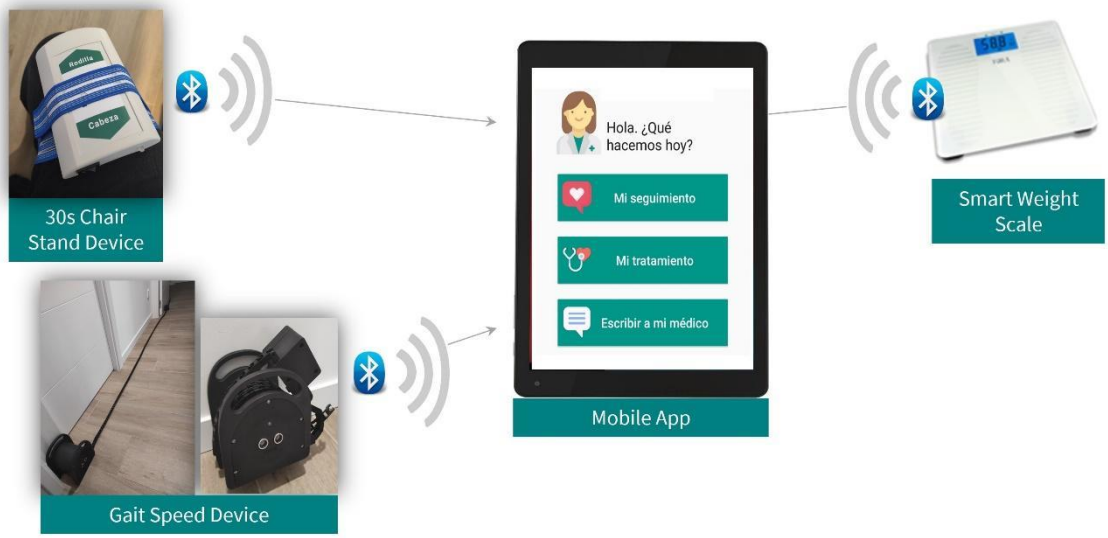


Figure 27 Global Home Monitoring System

3.3 ActiveUP

Regarding the increase of IoT monitoring systems, ActiveUp is born to take the leap to a project for simple monitoring and Internet of Things. The idea is creating a device that can measure most of the parameters before but considering transparency and ubiquity [23].

ActiveUp has some of the features of positive, but the most relevant evolution is a wearable system that can measure the inertial movements of the body. This

wearable is attached to a belt that wears at the waist. This device collects data regarded to the movements of the patient, and store them within an internal memory, to send them to the cloud to be analyzed in the future.

Basing on these features, the idea is using the same devices used for Positive Project regarding Gait Speed and Chair Stand Tests, to complement these measurements with a Wearable that is hanged to the waist by the user.

This wearable is composed by a microcontroller that can perform as a central processor, along with an accelerometer composed by an inertial sensor which is able to measure the acceleration and angular acceleration, a micro-SD card module containing a SD card to store the information collected by the accelerometer, and a RTC (Real Time Clock) able to record the current date and time and store them within the SD card too.

This Wearable can be configured via bluetooth by an external device, such as a smartphone or tablet, in order to store the required information within, including the home network and password, the ip needed for the broker, the user and password for the user and the time that the bluetooth is on.

Figure 27 First Wearable Prototype

The data stored within the wearable is sent to a middle path broker, which oversees processing the signals obtained by the wearable, and send to the remote cloud, to be seen by the doctors and caretakers. In this case, the broker is inside an external device, which is a Raspberry Pi model 4, located within home of patients.

Once the measurements are taken, they are sent to the Cloud Server by Internet. This Server Cloud consists of an API Rest in a static domain, which gathers the information within two databases, MySQL, and MongoDB. The inclusion of two databases instead of just one database, is because MySQL is a relational database, while MongoDB is a non-relational database. This means that MySQL contains a unique identifier that helps to organize data, but on the other hand, if there would be much request for the database, we need a faster response from the database, which in this case we will use MongoDB.

The storage within the cloud could be accessed by an app, that is pretty like the Positive. In this application, you could see your evolution, the progress (or regress) you are doing, and the next activities.

Comparing the results obtained by the wearable and the measurements from the devices before, we can develop an algorithm that can calculate average speed during a frame where the patient is moving, basing on different signals stored within this wearable.

Figure 28 Second Wearable Prototype

The architectural scheme of the project is based on several main components that interact with each other, the Wearable device, the Gait Speed device, the Chair Stand device, the Broker, the Server Cloud and the Smartphone or Tablet.

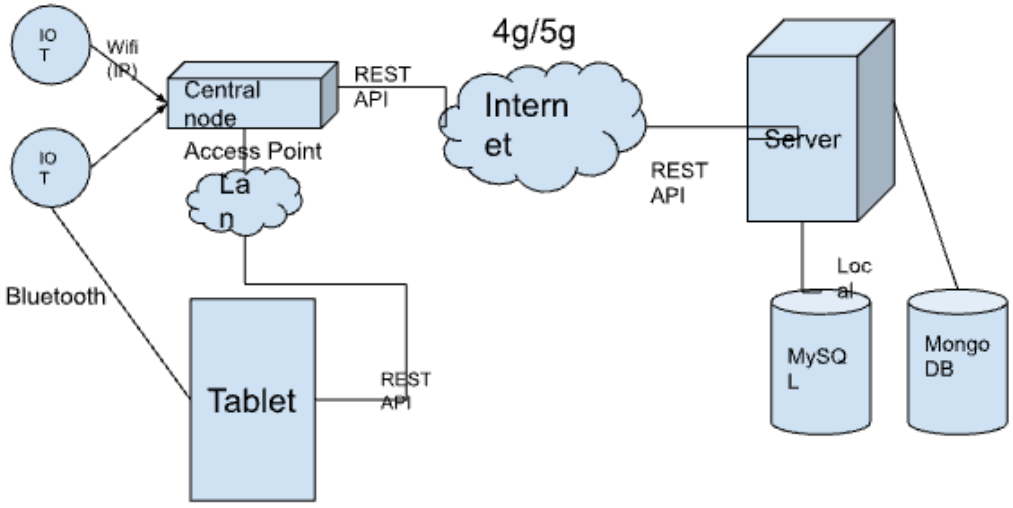


Figure 29 Main Architecture of ActiveUP

As we see in Figure 32, the devices in charge of the test, which means, the Wearable, Gait Speed and Chair Stand devices, are connected to the Central Node, by IOT protocols (MQTT). The IOT devices first connect to the home network and then they are assigned to an IP. In the case of the central node, it

is assigned to a static IP, so the devices know exactly where they can find the broker.

The broker itself is also connected to the smartphone or tablet by the home network, but in this case is not connected to the broker by an IOT protocol, but a Rest Api itself. This Rest Api is responsible for making requests about the Gait Speed and Chair Stand measurements taken by the patient.

4 Software Requirement Specification

4.1 Introduction

In this part of the document, we define the needed features to design a wearable able to collect inertial data and store it internally, to send them to the broker when the connection between wearable and the broker is available.

Next, we describe the minimal requirements that the hardware of the wearable must fulfil physically:

- The wearable must contain a battery able to supply enough power to maintain the device working during at least a day or two.
- The wearable must contain a commutator, to switch on and off the device and make it work when a plug is connected to it.
- The wearable must contain a processor able to execute the main commands to make the device a state machine with the rest of the components acting as peripherals of it. This processor must have at least one I2C connection, one SPI connection and 3 digital and 2 analogic GPIOs.
- The wearable must contain an accelerometer able to measure the movement of the patient in a predefined interval of time. This accelerometer must be able to communicate these measurements back to the processor.
- The wearable must contain a Real Time Clock (RTC) able to check the current date and time, to register when the measurements collected by the accelerometer are recorded.
- The wearable must contain a Micro SD Card Module able to receive a SD Card, to store the information regarding the accelerometer, date and time in one of the files.
- The wearable must contain a battery charger, to charge the battery when the power is depleted.
- The wearable must contain a RGB LED, able to show the possible states of the machine, to help the user to understand how the machine is working during all the time.
- The wearable must contain a couple of resistors and capacitors, to avoid overvoltage and overcurrent within the device, make needed voltage divider to convert from 5V to 3.3V and measure some features like analogic and digital features and parameters.
- The wearable must contain a push button, to change manually from one state to another in case we want to manipulate the functioning of the device.
- The wearable must have two different states well differentiated from each other. One for the setup mode, which is only executed once at the beginning of the sketch, and the other for the loop mode, which is executed indefinitely.

4.2 Functional requirements

FR1-Setup-State

- FR1.1 The Wearable must activate pins corresponding to the I2C, SPI, UART protocols. This means, the GPIO for these special pins must be disable and multiplexed to avoid conflict.
- FR1.2 The Wearable must activate pins corresponding to the digital IO of the push button, the SD card CS and the RGB input.
- FR1.3 The Wearable must activate pin corresponding to the analogic IO of the Battery Charge and the Plug Indicator.
- FR1.4 The Wearable must check whether there is an Accelerometer connected via I2C to the main processor or not. If it is, the processor can continue, otherwise the LED remains in red colour and blinks once per second.
- FR1.5 The Wearable must check whether there is a Real Time Clock (RTC) connected via I2C to the main processor or not. If it is, the processor can continue, otherwise the LED remains in red colour and blinks twice per second.
- FR1.6 The Wearable must check whether there is an attached micro–SD Card Module to the processor, and this module contains a Micro SD Card or not. If it fulfils both conditions before, the processor can continue, otherwise the LED remains in red colour and blinks four times per second.
- FR1.7 The Wearable must check whether the EEPROM Memory is empty or not, which means is the first time of general use. If the EEPROM contains the needed parameters within specific memory addresses, the processor can continue, otherwise the processor will activate Bluetooth and waits for the incoming data via Bluetooth. The needed data are the following:
    - FR1.7.1 A Home Network SSID needed to connect via Wi-Fi.
    - FR1.7.2 A Password for the SSID before.
    - FR1.7.3 An IP to address the broker.
    - FR1.7.4 A Name of the USER needed for passing to the broker to make a request to the cloud for the token.
    - FR1.7.5 A Password of the USER needed for passing to the broker to make a request to the cloud for the token.

  Each one of these parameters must be an array of characters, and none of them must exceed a specified length (in this case 32 bytes).

- FR1.8 The Wearable must make a calibration process before start working normally, to calculate the needed static offsets for each signal and correct possible errors related to manufacturing and position of the patient. To perform this, the processor takes a specified number of measurements from the accelerometer in a short, predefined interval, and after this it calculate the total average for each measurement. These averages are stored within dynamic memory to use them after each measurement.
- FR1.9 During this state, the RGB LED colour could be either red or yellow, but never green, to avoid the user to move.

FR2-Loop-State

- FR2.1 The Wearable must check whether the Plug Indicator is HIGH or LOW. If it is HIGH, the processor activates the status corresponding to the charging-battery state, regarding to see the amount of battery to check how much time is needed to plug the device to completely recharge the battery. If it is LOW, the processor activates the status corresponding to the Not-Charging-Battery status, regarding to collect data from the accelerometer.
- FR2.2 After completing any of these actions, the device must force the processor to a sleep mode, to save battery current and make the live of the battery possible.

FR3-Charging-Battery-State

- FR3.1 Whenever the battery is charging, the device must make the LED blink twice per second and middle second on and the other off. Only when the battery is full recharge, the LED stops blinking.
- FR3.2 When the battery is below 25%, the LED must turn red.
- FR3.3 When the battery is between 25% and 50% the LED must turn orange.
- FR3.4 When the battery is between 50% and 75% the LED must turn yellow.
- FR3.5 When the battery is between 75% and 100% the LED must turn green.

FR4-Not-Charging-Battery-State

- FR4.1 The device must find whether the button is pushed or nor. If it is pushed, the device switch to a special state according to FR4.2. Otherwise, it can continue to check the next condition (see FR4.3).
- FR4.2 The device must find how much time the button is pushed. If it is less than a specific and predefined time, the state pass from a regular or collecting data mode to a Bluetooth mode (Setting-Mode). Otherwise, the state pass from a regular mode to a Wi-Fi mode (Transferring-Mode).
- FR4.3 Once the device finds the button is not pushed, the device checks whether the previous hour of the timestamps recorded is the same. If it is, the device continues collecting measurements. Otherwise, the device pass from a regular or collecting data mode to a Wi-Fi mode (Transferring-Mode).

FR5-Regular-Mode

- FR5.1 The device checks whether it is a new day. If it is the same day, the device continues collecting data and stores them within the same file as before. Otherwise, the device creates a new CSV file, with a previous name predefined before, and the date when this file is created.
- FR5.2 If the SD card is above the 90% of its capacity, instead of creating a new CSV file, the device stops storing data within, in order to avoid the SD card being full, and the RGB LED blinks first with a large blink, and then with a short blink.
- FR5.3 If the conditions before are fulfilled (the day has not changed) and the second is the same too, the device must collect data from the

accelerometer, regarding of the accelerations and angular gyro accelerations from the accelerometer.

- FR5.4 The device subtracts the offset first calculated from the previous measurement, to correct errors regarded the calibration.
- FR5.5 The device applies a matrix transformation based on the rotation axis and the unitary vector.
- FR5.6 The device checks whether the second of the timestamp for the previous measurement is the same or not. If it is the same, the measurement is gathered within an array of characters. If it is different, it is transferred to the CSV file of the current day within the SD card.

FR6-Setting-Mode

- FR6.1 The device checks whether the Bluetooth has any user paired and connected to itself. If it is, the device waits until some information arrives from the user, depending on the time established for the user (by default 5 min) otherwise, it disconnects itself from the user.
- FR6.2 The device waits for a incoming message during a concrete period. If this time is overpassed, the device disconnects itself from Bluetooth and returns to the regular mode.
- FR6.3 Once the device has received any message, first checks whether the information arrived has come in a JSON format. If it is so, checks for relevant keywords, which are Class, data, and within this key, SSID, Password, ServerIP, MQTTUser and MQTTPassword. Otherwise discards this information. It could be sent also the time or the time zone for the RTC, but this information is not mandatory.
- FR6.4 If the JSON data contains any relevant key data, the device tries to store any of them within the memory part assigned to it.
- FR6.5 If the JSON data key contains a value that is within length, the data is stored within the memory address assigned to it, after erasing the previous information stored within. Otherwise, it is discarded.
- FR6.6 If the information is successfully stored within EEPROM, the counter for deactivating the Bluetooth is reset to 0.

FR7-Transfering-Mode

- FR7.1. The device turns the RGB LED into indigo colour and start trying to transfer the files generated. After this, the RGB LED turns into red colour.
- FR7.2 The device checks the connection with the home network. If the network is available and the password is granted, it continues to the next phase, which is finding the IP broker. Otherwise, it returns to the regular mode. In this case the led blinks once.
- FR7.3 The device checks the connection with the IP of the broker, via a ping command. If the IP is reachable, it continues to the next phase, which is trying to connect the ports regarded to the MQTT and FTP protocols. Otherwise, it returns to the regular mode. In this case the RGBLED blinks twice.
- FR7.4 The device checks the ports regarded to the MQTT and FTP protocols. If both ports are opened, the device starts trying to transfer data to the broker. Otherwise, it returns to the regular mode. In this case, the RGB LED blinks four times.

- FR7.5 The device starts checking all directories within the SD card. If any directory has an invalid name previously defined, the device erases it.
- FR7.6 The device starts checking all possible files within each directory. If any file has an invalid name previously defined, the device erases it.
- FR7.7 The device enters in a loop for transferring each file from the device to the FTP port broker. If the file has a larger size than a previous size partition previously defined, the device fractions the file and send each partition in the same order as its position in of the file.
- FR7.8 Once the device has transmitted all valid files, the device enters in a deep sleep mode to recharge battery and the processor rest (after activating Wi-Fi, the device has consumed a lot of energy). To quit from this deep sleep mode, the user must push the button.

4.3 Non-functional requirements

NFR1-Battery

- NFR1.1 The battery must hold the normal functioning of the device for at least 1 or 2 days.
- NFR1.2 The battery must not overheat excessively in order not to annoy the user.
- NFR1.3 The battery must allow the user to charge it without any controversy.

NFR2-Processor

- NFR2.1 The processor must have different functioning modes in order to save energy when not needed.
- NFR2.2 The processor must enter sleep mode every predetermined interval (in this case 50 ms) to rest and save current.
- NFR2.3 Every wake-up moment, the processor must gather data from the accelerometer and RTC by I2C, and SD card by SPI.
- NFR2.4 The processor must allow communication between RGB LED by PWM (Pulse Width Modulation)
- NFR2.5 The processor must be able to read analog lectures from the charge battery module.
- NFR2.6 The processor must be able to activate and deactivate Bluetooth protocols to allow the device to entering in a sleep mode.
- NFR2.7 The processor must be able to activate and deactivate Wi-Fi protocols to allow the device to entering in a sleep mode.

NFR3-RTC

- NFR3.1 The RTC must be able to set and reset the date and time in a binary format to pass the date and time to the microcontroller.
- NFR3.2 The RTC must be able to convert the date and time from a binary format to a datetime format.
- NFR3.3 The RTC must be able to pass the date and time to the microcontroller by I2C communication.

NFR4-Accelerometer

- NFR4.1 The accelerometer must be able to collect information about the acceleration and angular acceleration from an inertial sensor.
- NFR4.2 The accelerometer must be able to send the information to the microcontroller by a I2C protocol.
- *NFR5-Micro-SD Card Module*NFR5.1 The Micro-SD Card Module must be able to read from and write into a Micro-SD card embedded within the module, without restrictions from permissions and hardware design.
- NFR5.2 The Micro-SD Card Module must be able to find what type of Micro-SD card is being used.
- NFR5.3 The Micro-SD Card Module must be able to find the size of the Micro-SD card is being used.

NFR5-Type C Charger

- NFR6.1 The Type Charger must be able to accept only type-C Chargers.
- NFR6.2 The Type Charger must be able to check whether a Lipo-Battery is completely charged or not.
-

NFR6-RGB Led

- NFR7.1 The RGB LED must be able to blink at 50Hz, this means, it must be able to change the color at 50 times per second, to maintain the eye an impression of continuous light.
- NFR7.2 The RGB LED must be able to light 256 different intensities of red, green, and blue colors, making a total of 16777216 different colors.
- NFR7.3 The RGB LED must be able to light 256 different intensities of colors.

5 Development

5.1 Main components

5.1.1 Wearable

As exposed before, the Wearable consists of a Microcontroller connected to an accelerometer and a RTC by I2C wiring, along with a Micro-SD Card module connected by SPI wiring. Beside this, the Wearable contains a Type-C Charger to charge the battery, a RGB LED to indicate the patient in which state the Device is every moment, and a push button, in case we want to force the device to change the status manually.

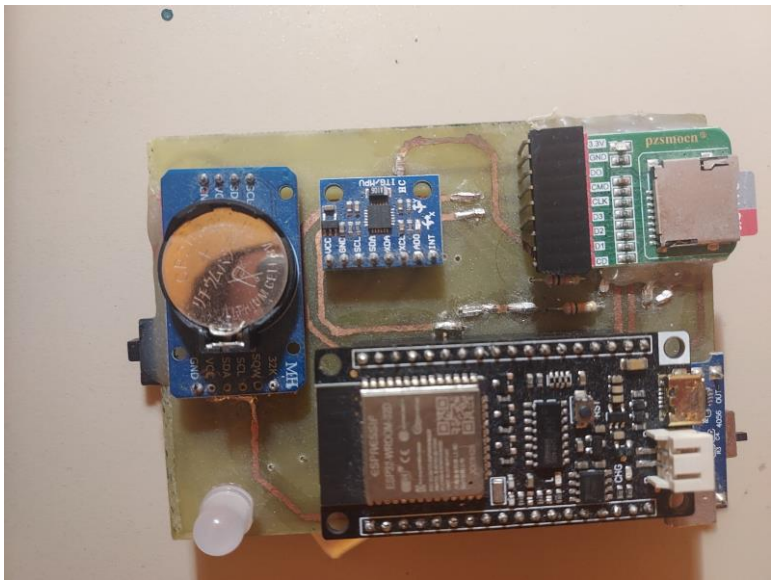


Figure 30 Internal Components of Wearable Device

The LED can swift the color for green when the device is in regular mode, to blue when is in setting mode and Bluetooth connected, and to indigo when is in transmission mode and Wi-Fi connected.

The regular mode means the device is collecting information from the accelerometer and the date and time from the RTC, and is storing it within a CSV file named with the date when it was created, in the SD Card. The measurements taken by the wearable are transformed first with a subtraction of an Offset previously taken, and then by a 3-Axis Rotation Matrix, one corresponding by each axis. The device enters after this in a light sleep mode, which means the processor is stopped and save battery in order not to deplete the battery.

The setting mode means the device activates the Bluetooth and is suspended waiting for another device to connect by Bluetooth, or until a previous defined timeout has passed. If this happens, the device returns to the regular mode and continues to collect information. If there is a connection with a smartphone or tablet, the device waits for a message and, when the message comes, it analyzes, first whether it is in JSON format, second the keys within the JSON, and then whether those keys contain a valid value. After this, the wearable blinks and waits for another message, or a manual disconnection.

The transmission mode means the device tries to connect with the broker, first checking the network is a valid network, if exists and the password is correct, second checking if the IP of the broker is reachable or unreachable, and third checking if the MQTT and FTP ports are enabled or disabled. After checking these three conditions meet, the device begins transferring every file that contains within the SD-Card and then erase from it. After all corresponding files are sent, the device enters in a deep sleep mode, which means the Processor stops completely and resets itself. The only way to continue collecting information is pushing the button.

There is also a fifth state, which is when the Device is off and there is a Charger plugged into the Type-C Charge module. In this case, the LED is blinking all the time until the battery is completely charged. The LED is red when is below 25% of battery, is orange when is between 25 and 50%, is yellow when is between 50 and 75%, and is green when is between 75 and 100%.

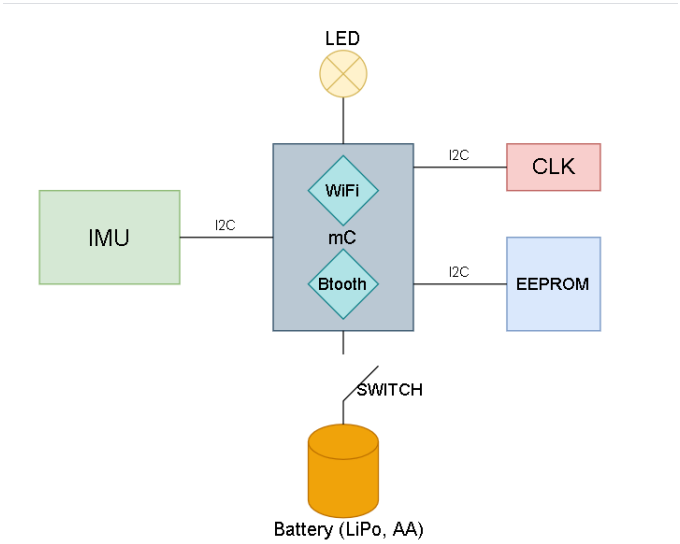


Figure 31 Wearable Schedule Based on Main Components

5.1.2 Broker

The broker consists of a Raspberry Pi Model 4, which contains within an Operating System based on GNU/Linux (in this case Ubuntu). To save resources, this OS does not have graphical interface.

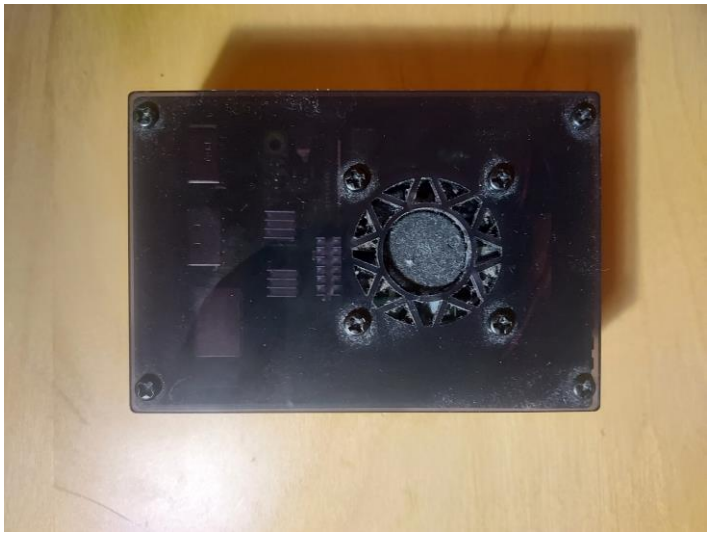


Figure 32 Raspberry Pi Model 4 - Broker

This OS has the following main user “ubuntu” and password “ageinglab”. Within this user, the directory where the files would be saved is “files”. Besides, there is also a directory where the code for the broker is installed. In this case, there is a docker launcher within this directory. This docker contains an agent of Broker called Mosquitto, which handles the messaging of MQTT messages, and oversees set the MQTT and FTP ports. It also contains an agent of broker called VSFTPD responsible for the FTP performance. The docker also activates the broker process in this case, to analyze the possibles messages that comes into the MQTT port and the files that comes into the FTP port.

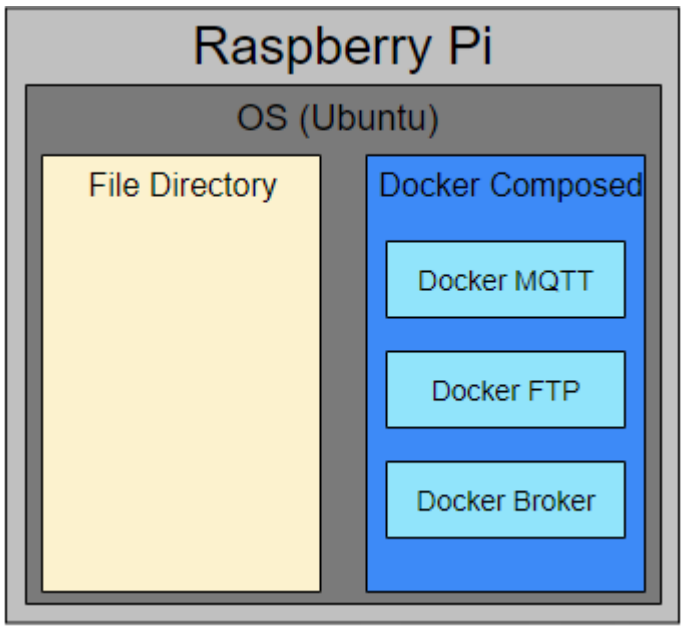


Figure 33 Main Schematic of Raspberry Pi Architecture

The docker broker contains several files within, that are necessary for the good performance of the Broker, the log of the broker, the configuration file, and the parameters from the home sensors (this will be applicable in future versions).

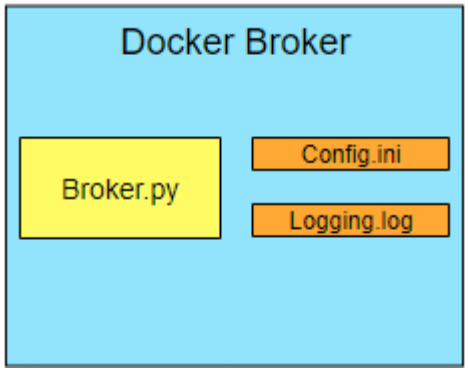


Figure 34 Main Scheme of Broker Docker - Mosquitto

When the Raspberry Pi boots, it states the minimum requirements for the good performance set within the init.d file. The docker composed must be here to be launched at the beginning of the process.

After that, if there is not the required file directory, the OS creates it, and launch the docker containing the broker and the other dockers responsible for launching the MQTT process (Mosquitto) and the FTP process (VSFTPD). This processes also handles the activation and deactivation of the MQTT port (1883) and FTP port (20 and 21). When the Docker of the Broker is launched, it immediately launches the executable based on a python script named Broker.py.

Once the process Broker.py is launched, first checks whether there are all relevant data contained within the Config.ini file, that is, user, password, local IP address, remote server IP address, log path and file, token path and file, patient ID, and JSON configurations. If it is something relevant missing, it immediately breaks the process printing the main motive for the stop.

When the relevant information is loaded and assuming there is no relevant information missing or errors related to the syntax, it immediately launches two additional threads, one regarding the control of the files that arrive to the broker, and the other for the connection by SFTP with the broker.

When a message arrives to the broker, it immediately analyzes the information, first checking whether the message is a valid JSON or not, second checking the keys arrived naturally and third checking the data for these keys. There are two types of valid JSON messages:

- A message for login: Is about a JSON request for making a login in the server. The broker makes an HTTP request of login and returns the response from the server. The structure of the message would be something like this:

```
{
    "message": "Random message",
    "profile": {
        "MQTTUser": "pruebashospital@email.com",
        "MQTTPassword": "pruebashospital"
    }
}
```

Figure 35 Valid Message for a Login to the Server

The JSON contains a message key, which could be any message the manufacturer will, and a profile object, which contain the user and password. The user could be an email or a simple ID, which will be what we use in this case.

- A message of transmission of a file: Is about a JSON announcing the name and the size of transmission of a file, and the state of this transmission, by FTP protocol. The structure of the message would be something like this:

```
{
    "status": 0,
    "file": "IMUWearable_20_11_2022.csv",
    "size": 123456
}
```

Figure 36 Valid Message for Transmission of CSV File

The JSON contains a status key of the file transmission, which indicates by a number (0-starts, 1-in process, 2-finished), the name of the file and the size of that file, in bytes.

If a transmission message arrives, the file is saved inside the directory, and the thread responsible for handling the files creates a block that avoids transmitting the file to the remote server.

The broker tries to send the files saved within to the server, when the ID of the patient is the same as the module of the hour of the day. For instance, a patient with the ID equals to 5, would transmit at 5:00 AM, and the patient with the ID 43, would transmit the file at 17:00 PM.

The process for sending the information to the cloud server is the following: the broker checks the connection by SFTP with the server with the remote IP address, the remote port and the RSA key, which is given at the beginning of the process. If something went wrong, the broker launches a message indicating the motive for the failure. If the broker sends the file with success, a message from the broker appears pointing this success. After that, the broker erases the file recently transferred to the cloud. The broker repeats this process until the directory where the files are saved is empty.

5.1.3 Cloud

The cloud server is in a remote address and consists of two parts:

- The storage for CSV files.
- The API REST for gathering measurements from the other devices (Gaitspeed and Chairstand) [24].

The storage for CSV files consists of a remote server (in this case a Raspberry Pi) coupled with an external device with high capacity to storage information from all the possible users (in this case an external HDD disk with capacity of 2 TB). This HDD disk is mounted as it was a part of the Raspberry Pi System.

The directory where the files sent to the server is called "ftp-files" and is located within the HDD disk. Inside, the organization of the directories are with different

folders, which they consist of the name Patient and the id of the Patient which they are saved.

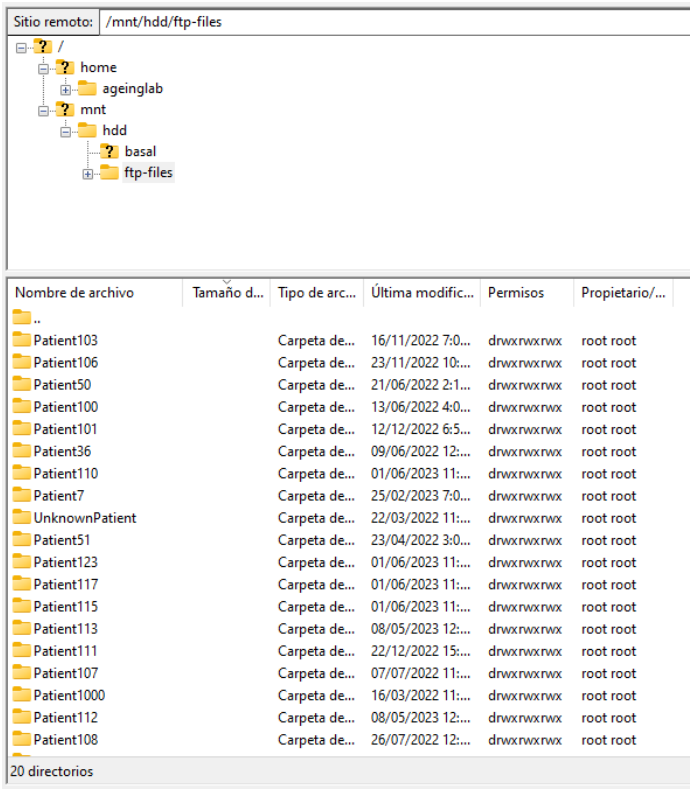


Figure 37 Organization of Files within Cloud Server

Inside these directories the CSV files recovered by the broker are contained in alphabetical order:

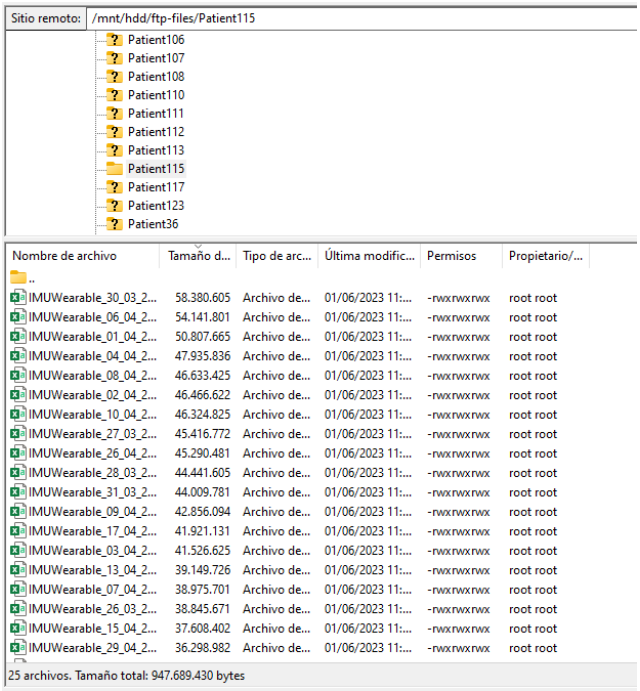


Figure 38 CSV Files within Patient 115 Route

The other part of the cloud is an API REST located in a remote address (in this case in the CESVIMA building) and is accessible by https protocol. This API is built on FastApi Framework and is composed of several URIs from we can make external request. For these URIs, we only focus on the ones that are related to the communication with the broker:

- Register: POST
    - Body:

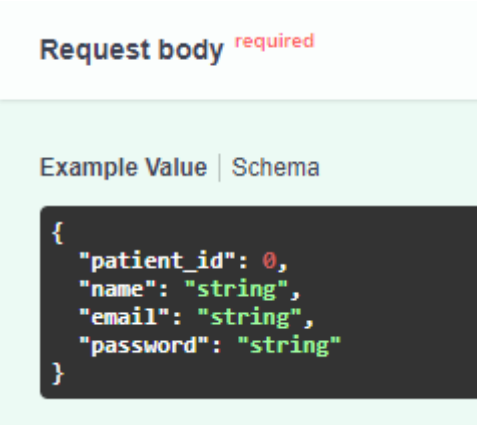


Figure 39 Register URI Body Request

- Responses:

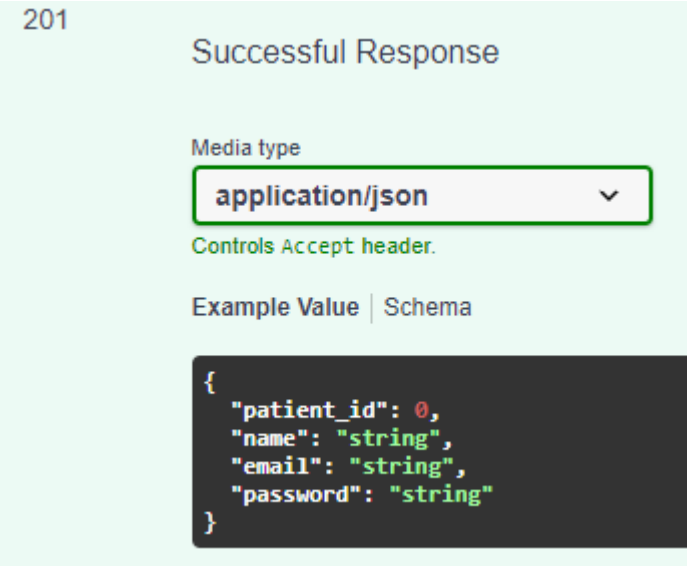


Figure 40 Register URI Success Response

Figure 41 Register URI Error Response

- Login: POST
  - Body:

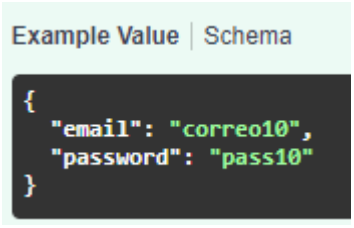


Figure 42 Login URI Body Request

  - Response:

Figure 43 Login URI Success Response

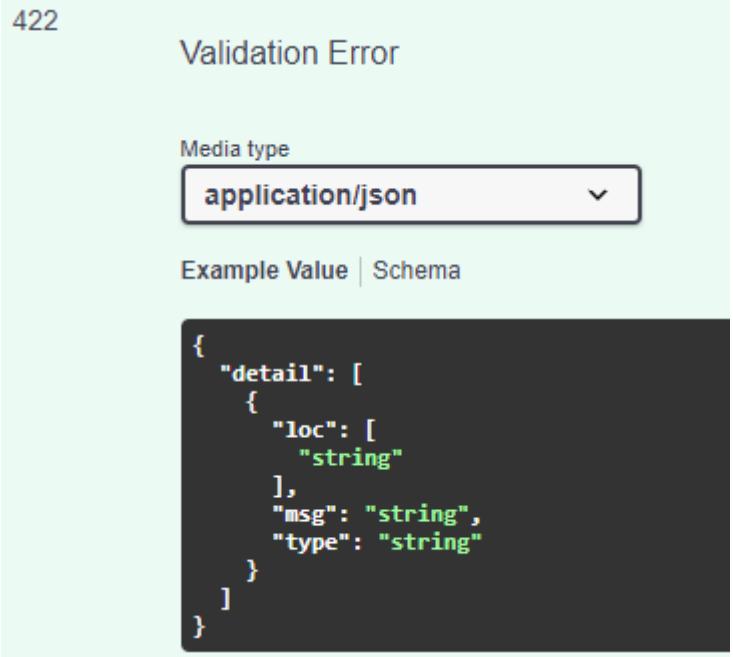


Figure 44 Login URI Error Response

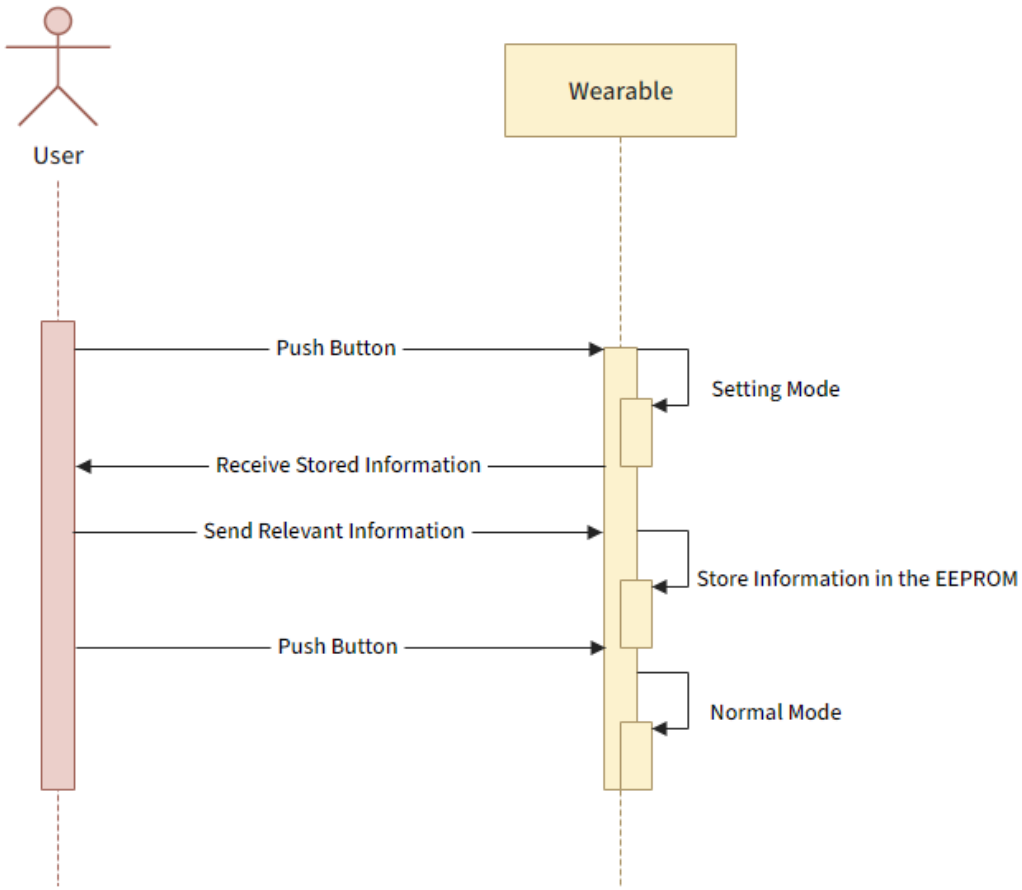
5.1.4 Tablet

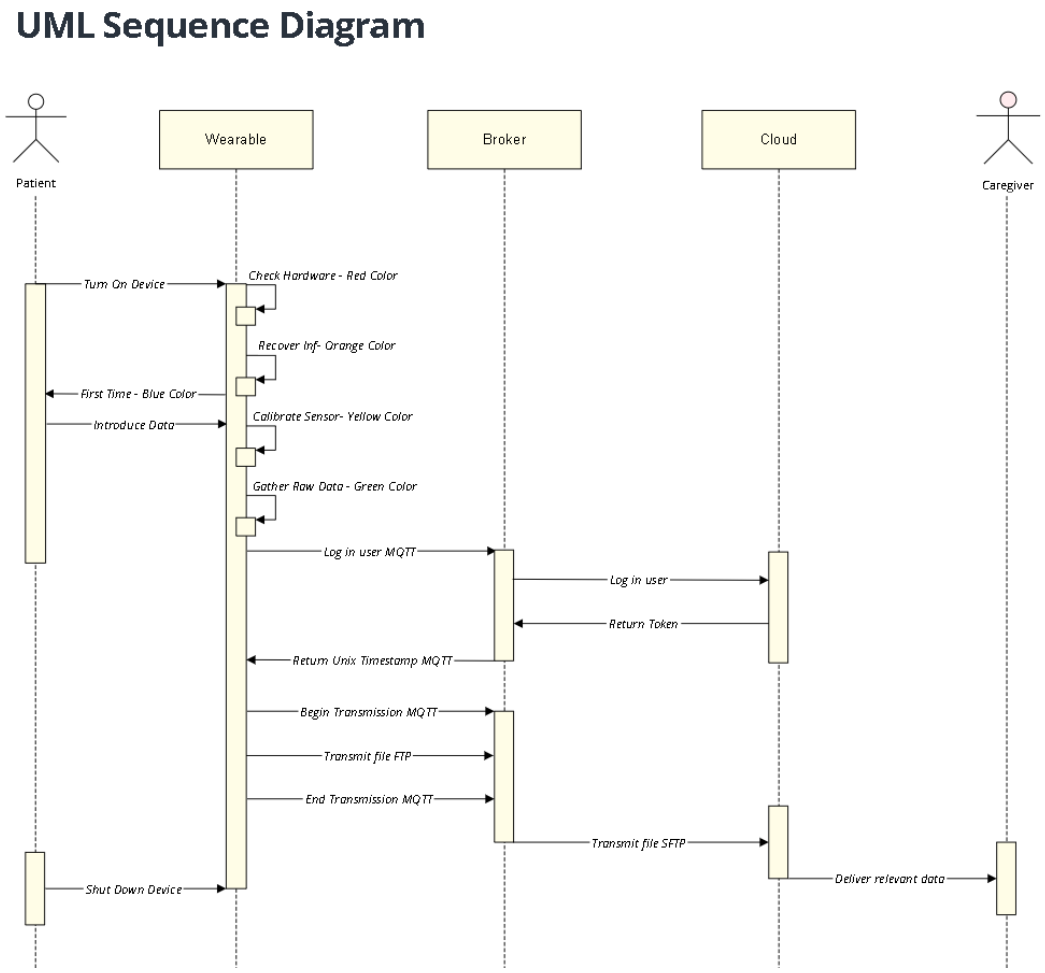
The main purpose of the tablet is an application where you can interact with all different devices, one of which is the Wearable. You can transfer all the relevant data, such as the home network, the password, the Broker IP address, the user and password and the time active for the Bluetooth module:

In the first use of the Wearable, the device will remain in setting mode until it receives the relevant information regarding the needs of the connection with the broker. After receiving all the information, it will store it internally and then behave normally.

5.2 Sequence diagram

Wearable-User:





5.3 Hardware development

5.3.1 Modules

5.3.1.1 Microcontroller

The module selected for enact as microcontroller is ESP32, a family of devices designed by Espressif Systems. In this case one trademark called Firebeetle. This trademark has been designed by an enterprise named DFRobot, which offers a great index of Open-Source products available for the community.

ESP32 is a low-cost and low-power microcontroller chip that has been integrated with Wi-Fi and Bluetooth modules. The ESP32 microcontroller employs an Xtensa Chip LX6 or 7 [25].

One of the main features of ESP32 is:

- Processors:
    - CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS

    - Ultra-low power (ULP) co-processor
- Memory: 320 KiB RAM, 448 KiB ROM
- Wireless connectivity:
    - Wi-Fi: 802.11 b/g/n
    - Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)
- Peripheral interfaces:
    - 34 × programmable GPIOs
    - 12-bit SAR ADC up to 18 channels
    - 2 × 8-bit DACs
    - 10 × touch sensors (capacitive sensing GPIOs)
    - 4 × SPI
    - 2 × I²S interfaces
    - 2 × I²C interfaces
    - 3 × UART
    - SD/SDIO/CE-ATA/MMC/eMMC host controller
    - SDIO/SPI slave controller
    - Ethernet MAC interface with dedicated DMA and planned IEEE 1588 Precision Time Protocol support[4]
    - CAN bus 2.0.
    - Infrared remote controller (TX/RX, up to 8 channels)
    - Pulse counter (capable of full quadrature decoding)
    - Motor PWM
    - LED PWM (up to 16 channels)
    - Ultra-low power analog pre-amplifier
- Security:
    - IEEE 802.11 standard security features all supported, including WPA, WPA2, WPA3 (depending on version) and WLAN Authentication and Privacy Infrastructure (WAPI)
    - Secure boot
    - Flash encryption
    - 1024-bit OTP, up to 768-bit for customers
    - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)
- Power management:
    - Internal low-dropout regulator
    - Individual power domain for RTC
    - 5 µA deep sleep current
    - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt.

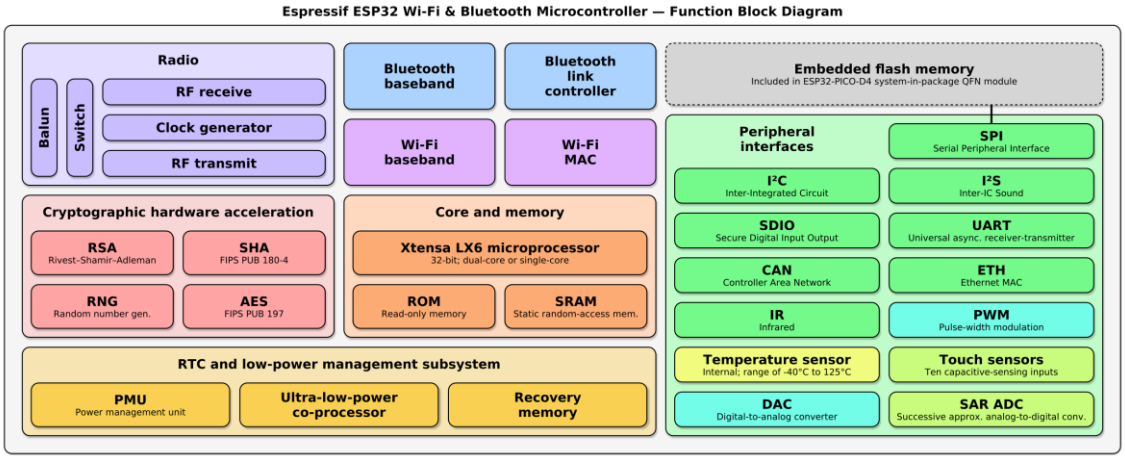


Figure 45 ESP32 Main Features and Architecture

The main advantage of ESP32 regarding other ESP32 devices is the low power consumption comparing other ESP32 devices, especially when the chip is in Sleep Mode. While in light sleep and deep sleep modes the current consumption of other devices is respectively by average 10 and 6 mA, ESP32 Firebeetle is 1mA for Light Sleep and 10 µA for a Deep Sleep mode, increasing considerably the live of time of the battery [26][27].

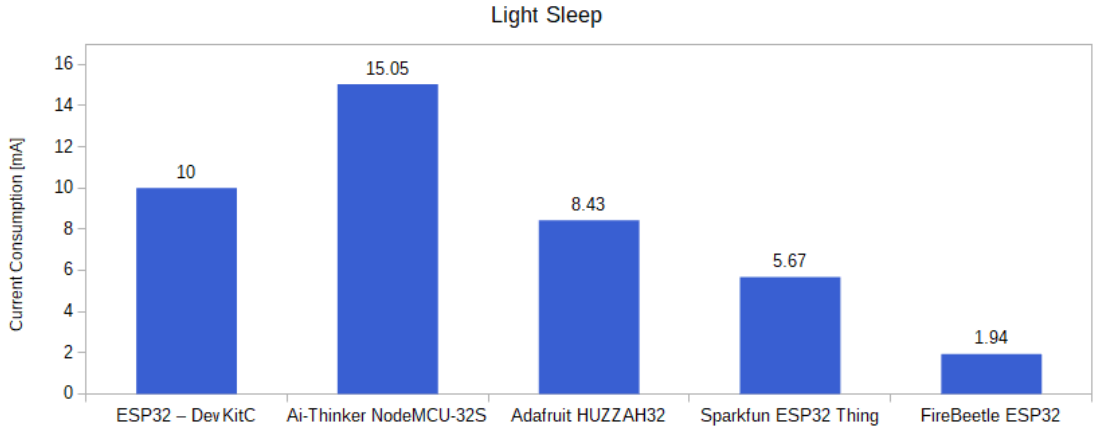


Figure 46 ESP32 Light Sleep Current Consumption

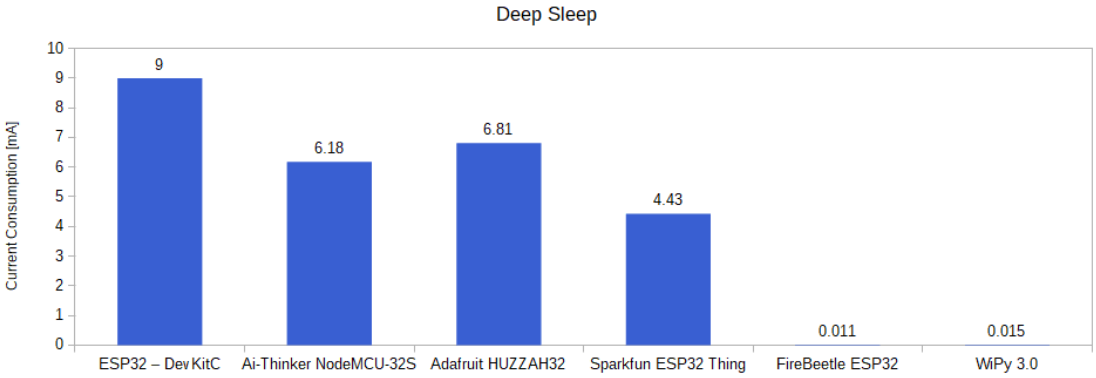


Figure 47 ESP32 Deep Sleep Current Consumption [28]

5.3.1.2 Accelerometer

The accelerometer is, after the Microcontroller, the main and most important part of the device. The model is MPU6050, which is an Inertial Measurement Unit with 6 Degrees of Freedom (DoF) due to its 3-axis accelerometer and 3 axis gyroscopes. It can also measure magnetic fields with a special sensor based on Hall effect [29].

Figure 48 MPU 6050 Accelerometer Module

The basic principle of working is the own definition of acceleration, which is the variation of speed during a minimum period. This accelerometer has a Micro Electromechanical System composed of a spring joint to two charges. Once the movement has increased to one side, the charge goes to the other side due to the inertial principle.

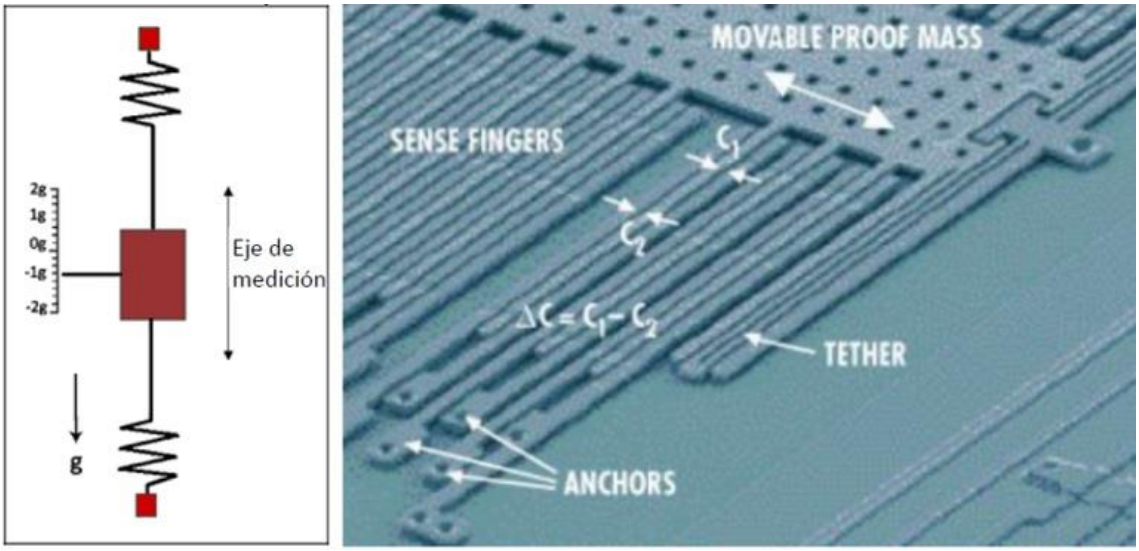


Figure 49 MPU Principle of Working

Regarding the angular acceleration, which means, the variation of angular speed per unit of time, the accelerometer is based on Coriolis effect. With a gyroscope we can measure the angular speed, and if the angular speed is integrated with respect to time, the angular displacement is obtained [30].

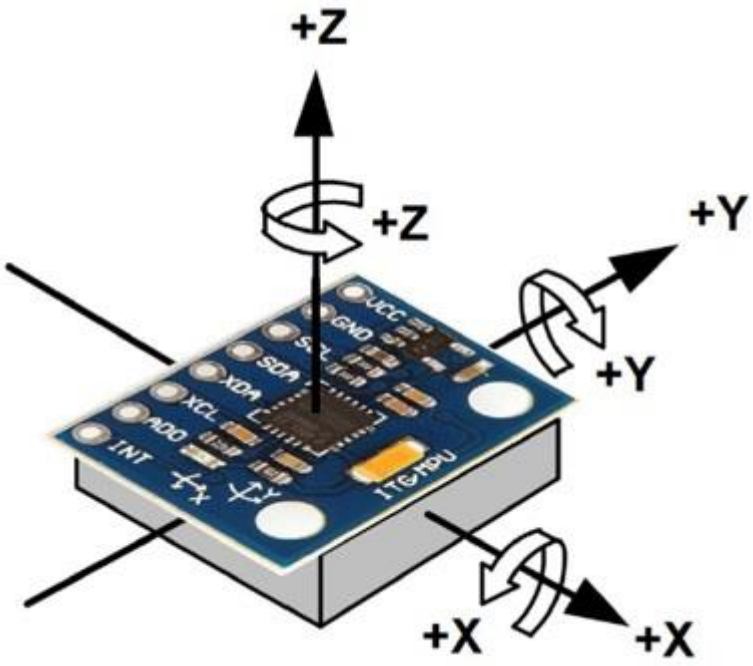


Figure 50 MPU6050 Module and its Axis

The communication with the microcontroller is made with an I2C wiring.

5.3.1.3 RTC

The RTC is the device responsible for taking the date and time during each measurement. The main difference with electronic clocks is that this kind of devices are able to measure real date and time, while the electronical clock are only able to count signal clocks [31].

The device chosen for the RTC is DS3231, which has the advantage with the previous version, DS1307, of more precision with the date and time. While DS1307 has variations of time due to high temperature, which makes a variation of 1 or 2 minutes per day, DS3231 has a correction of temperature diphase and therefore more precision (maximum 172ms per day) [32].

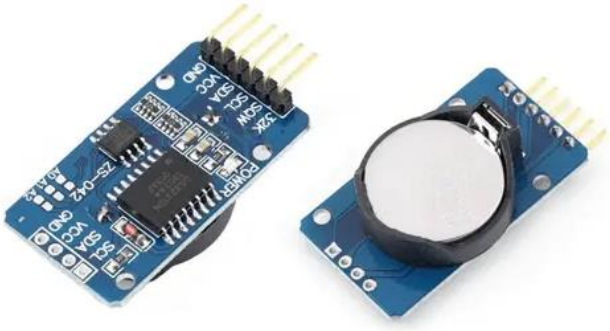
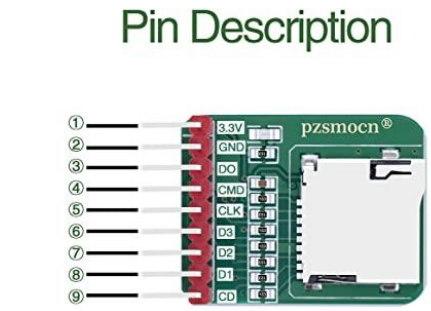


Figure 51 RTC Module

The device has an internal button battery that enables the module to keep working even after not receiving external power source supply. The communication with the main Microcontroller, is by I2C wiring.

5.3.1.4 Micro-SD card module

The SD card module is composed basically by a SPI wiring connection that connects the microcontroller with the module. In the final product, this module is substituted with a simple Micro-SD Card Shield, in order to make it simpler.



	Function (SD mode)	Function (SPI mode)
①	Power supply (3.3V)	Power supply (3.3V)
②	Power ground	Power ground
③	Data line 0	Master in and slave out (MISO)
④	Command line	Master out slave in (MOSI)
⑤	Clock	Clock (SCK)
⑥	Data line 3	Chip select / slave select (SS)
⑦	Data line 2	Reserved
⑧	Data line 1	Reserved
⑨	Reserved	Reserved

Figure 52 Micro-SD Card Shield[33]

5.3.1.5 Type-C charger

The Type-C charger module consist of an adapter of charger for the LiPo (Lithium-Polymer) battery, to allow it to recharge. It consists basically of a TP4056 chip, which is connected to two LEDs, one to indicate that the battery is being charged, and the other to show when the battery is fully charged.

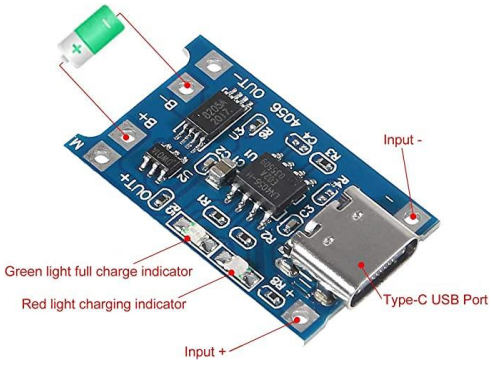


Figure 53 Type C Charger[34]

5.3.2 Components

5.3.2.1 Resistors

The resistors used for this device are the following:

- Resistor 10KΩ (x4)
- Resistor 20KΩ (x1)
- Resistor 330Ω (x1)

The 330Ω resistor is used for the Vin of the RGB Led input, one of the 10KΩ resistor is used for the push button connected to the ground, and the rest of them are used for voltage dividers.

5.3.2.2 Capacitors

One capacitor of 0.1μF is used for avoiding overvoltage to the GPIO regarding the entrance of the positive pin of battery.

5.3.2.3 Other elements

- Push Button: One side connected to the Vcc and the other connected to a resistor with Ground.
- RGB LED: Three pins, one connected to Vcc, one connected to the Ground and the third connected to one pin.
- Commutator: To switch on and off the device.

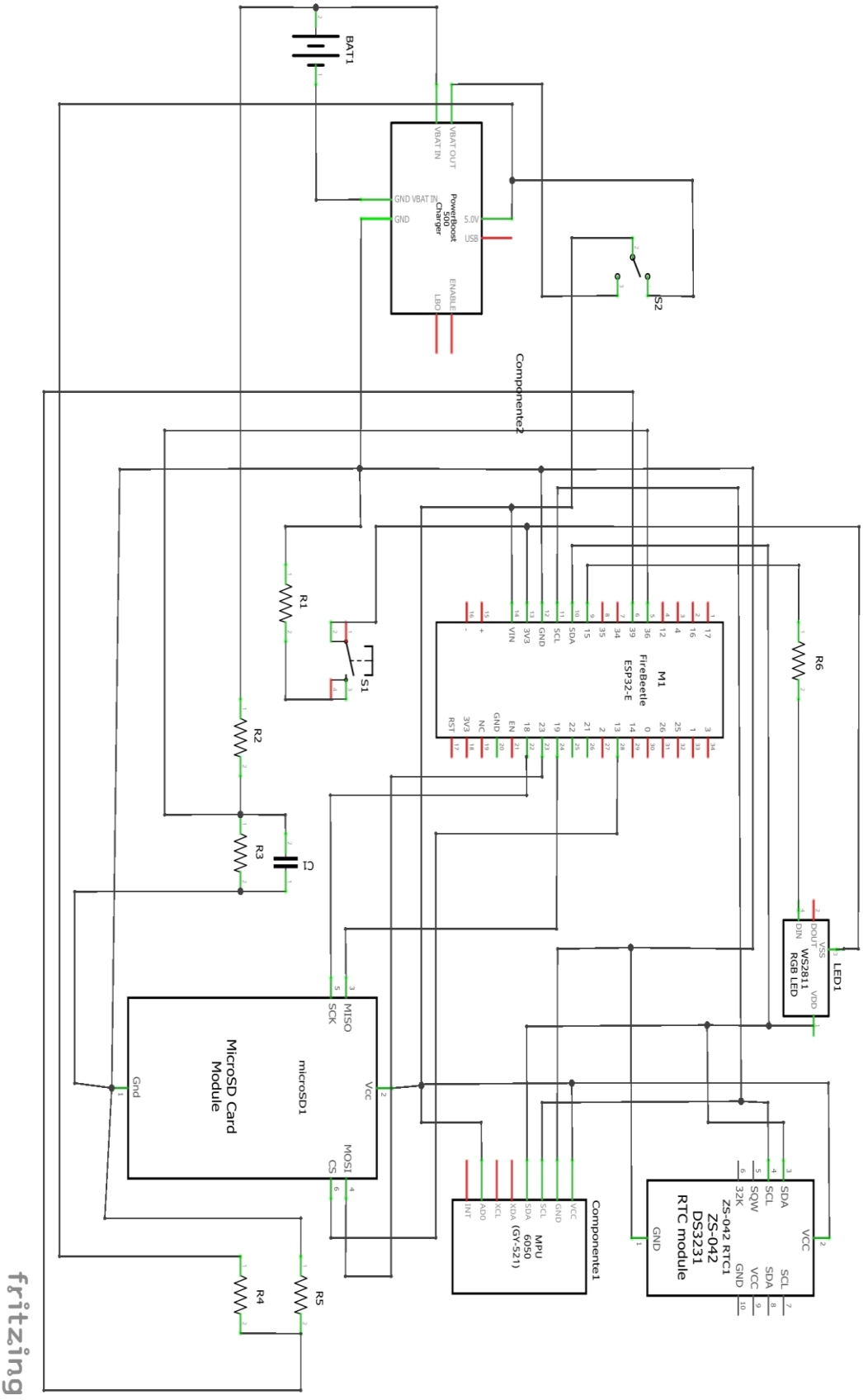
5.3.3 Connections

- Battery positive pin must be connected to the positive battery pin of TP4056 module.
- Battery negative pin must be connected to the negative battery pin of TP4056 module.
- Vin and Vout negative pins from TP4056 module must be connected to Gnd of ESP32, MPU6050, DS3231, SD Card module, and PL9823.
- Vout positive pin from TP4056 module must be connected to one of the two separated pins of the commutator.
- Vin positive pin from TP4056 module must be connected to the other separated pin of the commutator.
- The common pin of the commutator must be connected to Vin from ESP32, Vcc from MPU6050 and from DS3231, and AD0 from MPU6050 (Both MPU6050 and DS3231 have by default a slave ID of 0x68, so in order not to have problems of compatibility, we reset the slave ID of MPU6050 to 0x69).

- The SDA (IO21) pin from ESP32 must be connected to the SDA from MPU6050 and the DS3231.
- The SCL (IO22) pin from ESP32 must be connected to the SCL from MPU6050 and the DS3231.
- The MISO (IO19) pin from ESP32 must be connected to the DO (MISO) from MicroSD Card Module.
- The MOSI (IO23) pin from ESP32 must be connected to the DI (MOSI) from MicroSD Card Module.
- The SCK (IO18) pin from ESP32 must be connected to the SCLK from MicroSD Card Module.

- The GPIO13 pin from ESP32 is assigned to the CS for the MicroSD Card and so that must be connected to the D3.
- The GPIO5 pin from ESP32 is assigned to the pulse wave entrance of the RGB LED, so that must be connected to the Din of the PL9823 and a 330 Ohms resistor between them.
- The 3.3 V pin from ESP32 must be connected to the Vdd pin from PL9823 and one of the sides of the push button.
- The other side of the button is connected to the GPIO15/A4 and the Ground, and a 10K resistor between them, so when the button is pushed on, it gives 3.3V and when it is off, it gives 0V.
- The Battery positive pin must be connected to one voltage divider, composed of a 10K and a 20K. Between them is connected to the GPIO36/A0. The 20K resistor is also connected in parallel with a capacitor of 0.1uF.
- The Vin positive pin from TP4056 must be connected to a voltage divider, composed of two 10K resistors, and between them is connected to GPIO39/A1 [35].

5.3.4 PCB layout



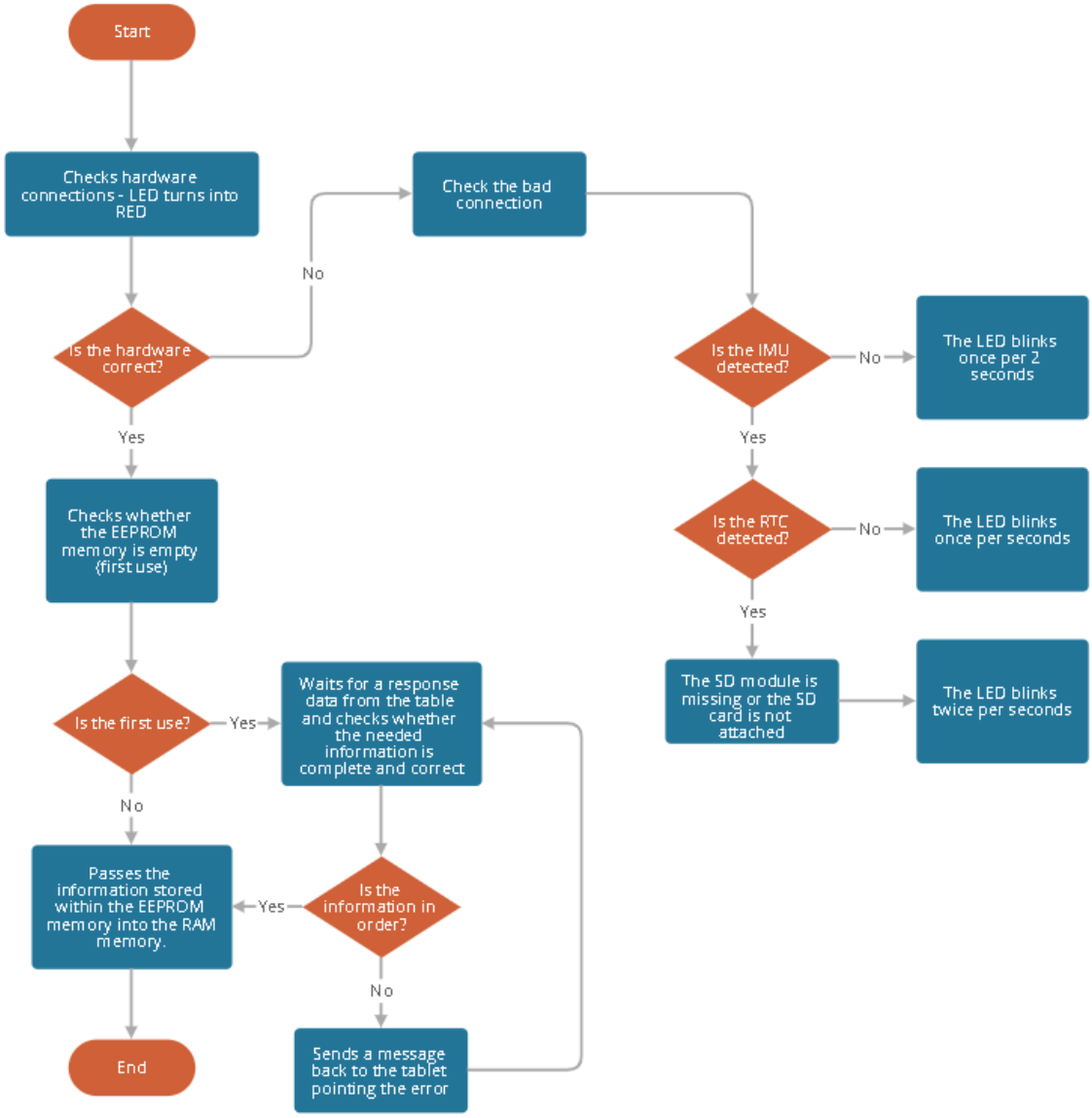
1Schematic of the Device

5.4 Software development

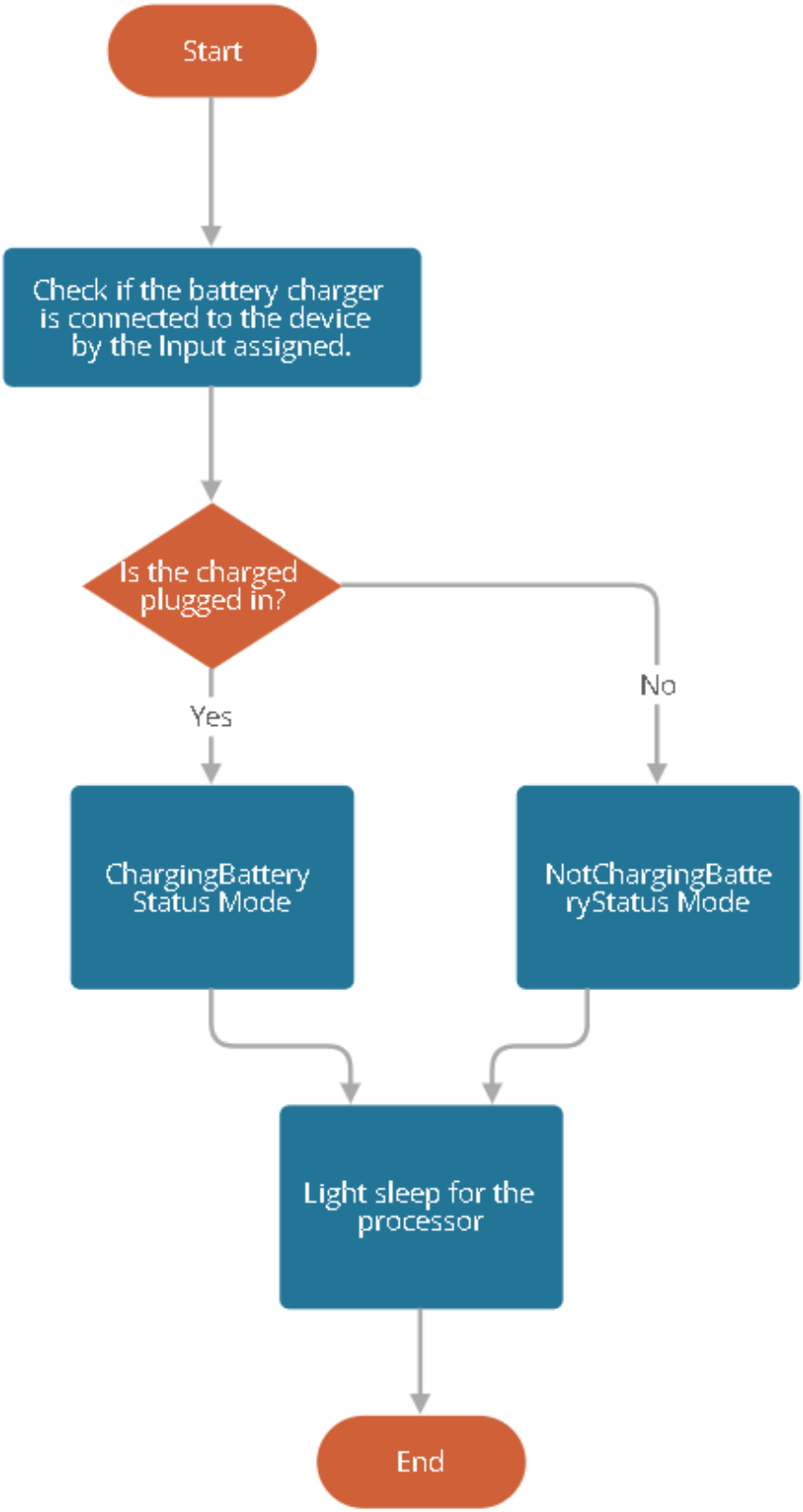
5.4.1 Flowcharts

[35]

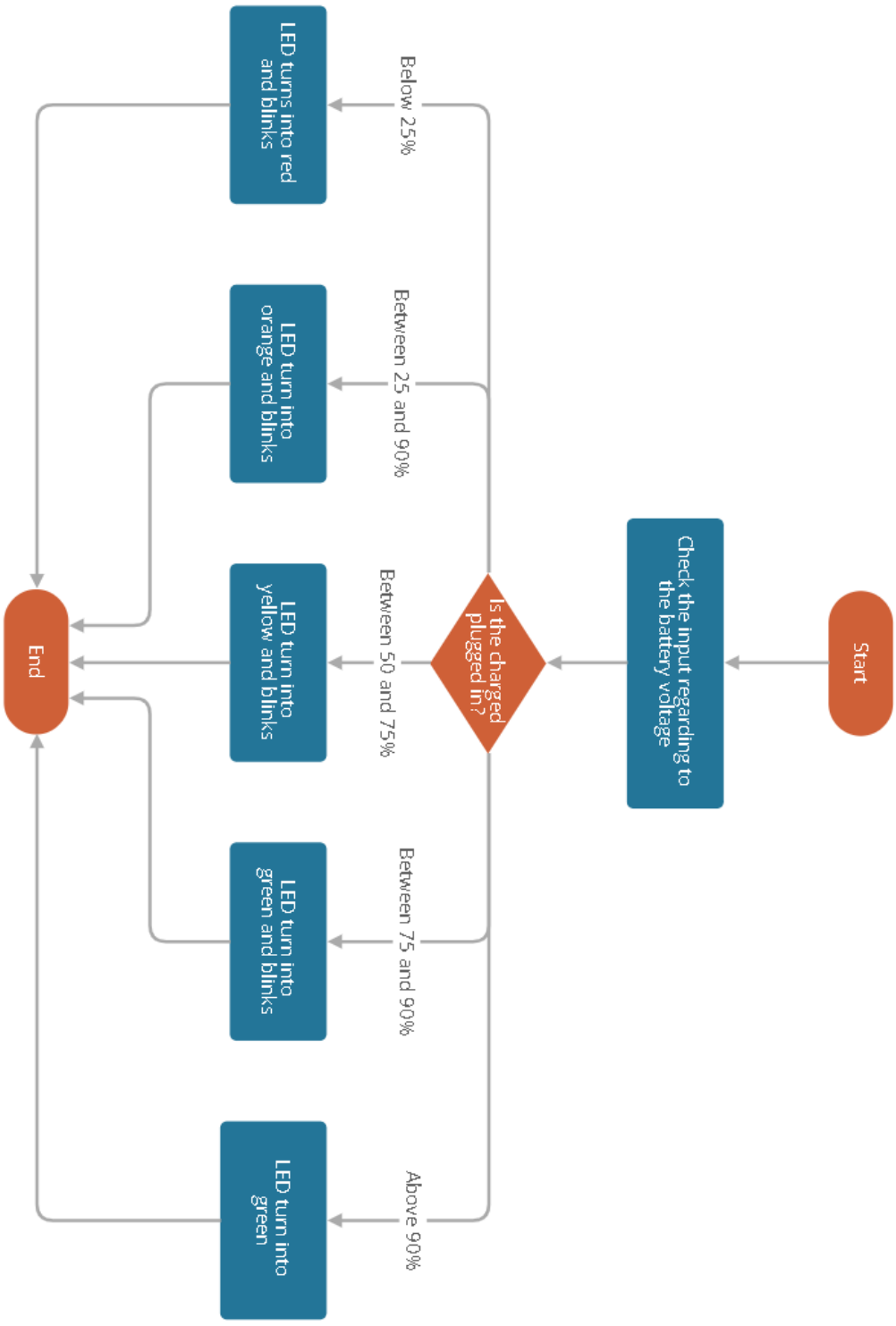
Setup Mode:



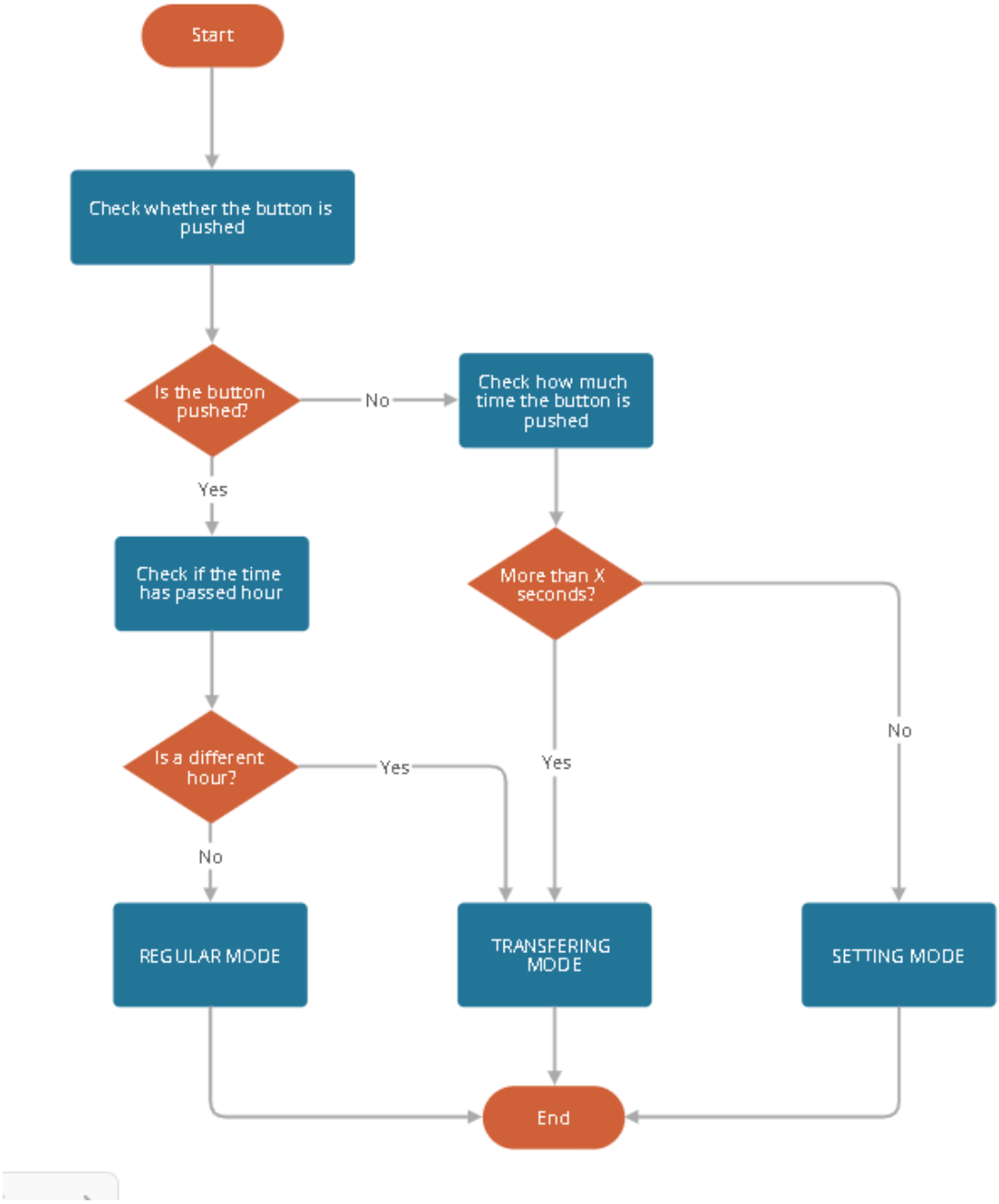
Loop Mode:


Start
Check if the battery charger is connected to the device by the Input assigned.
Is the charged plugged in?
Yes
No
ChargingBattery Status Mode
NotChargingBatteryStatus Mode
Light sleep for the processor
End

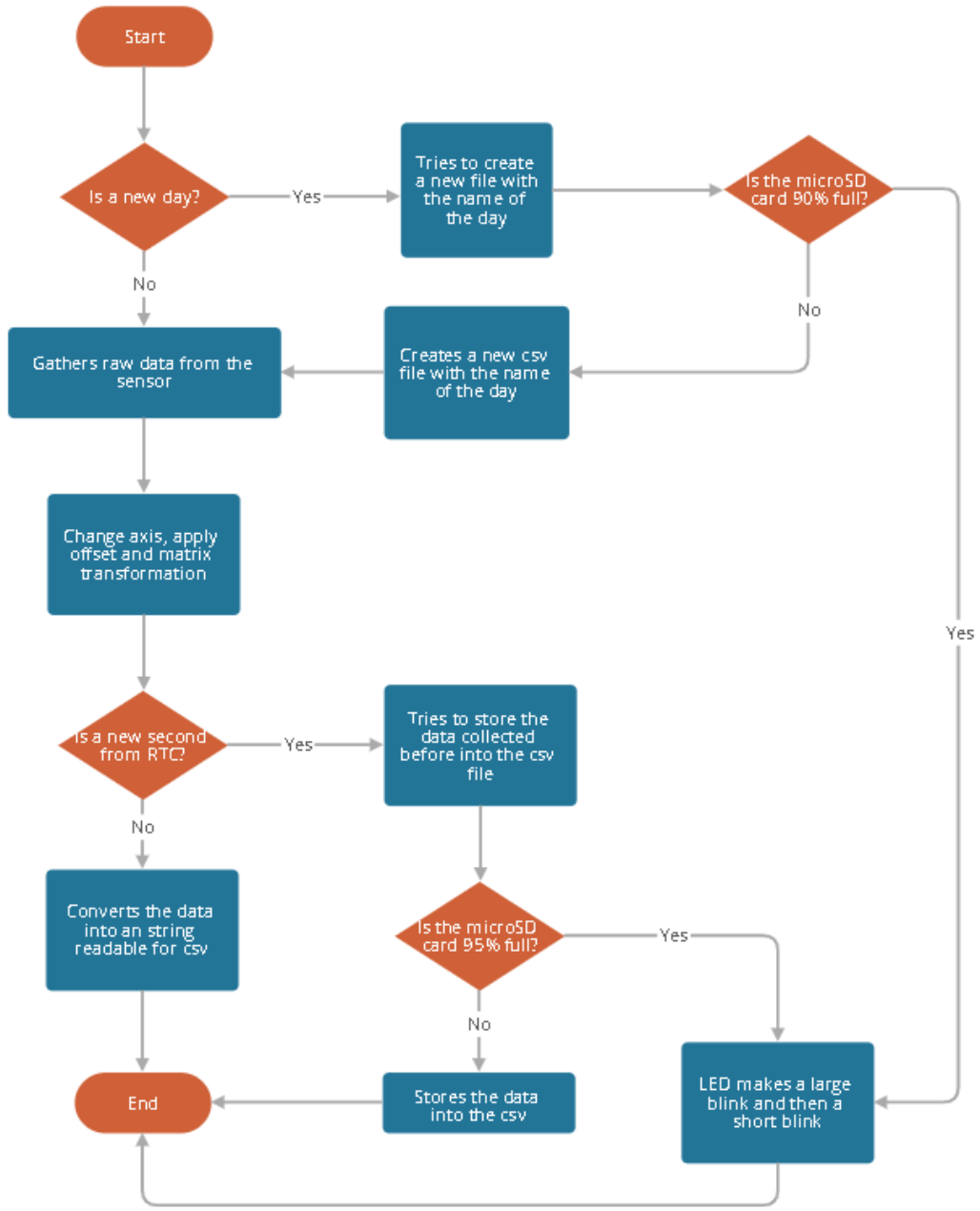
Charging Battery Mode:



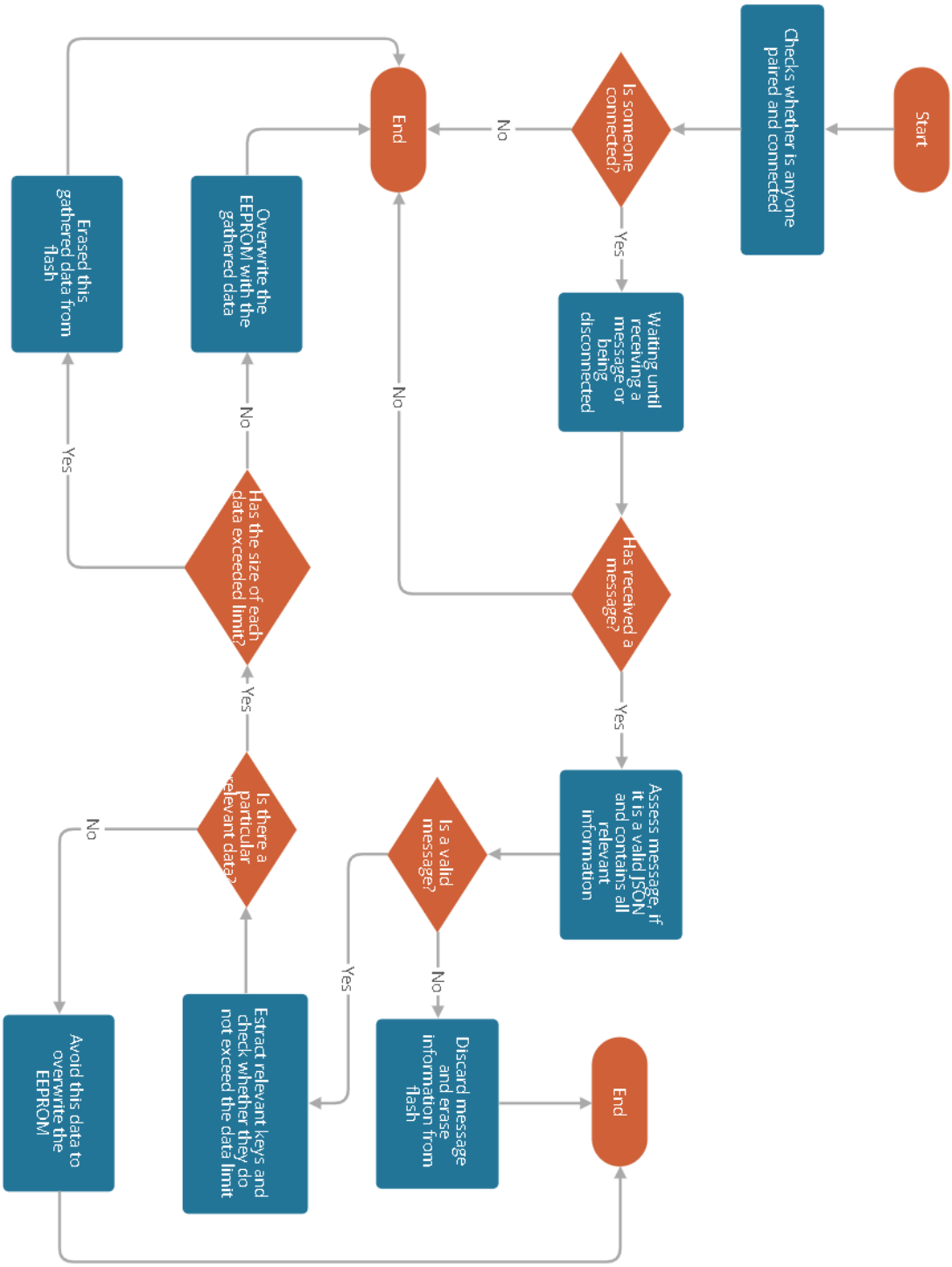
Not-Charging Battery Mode:



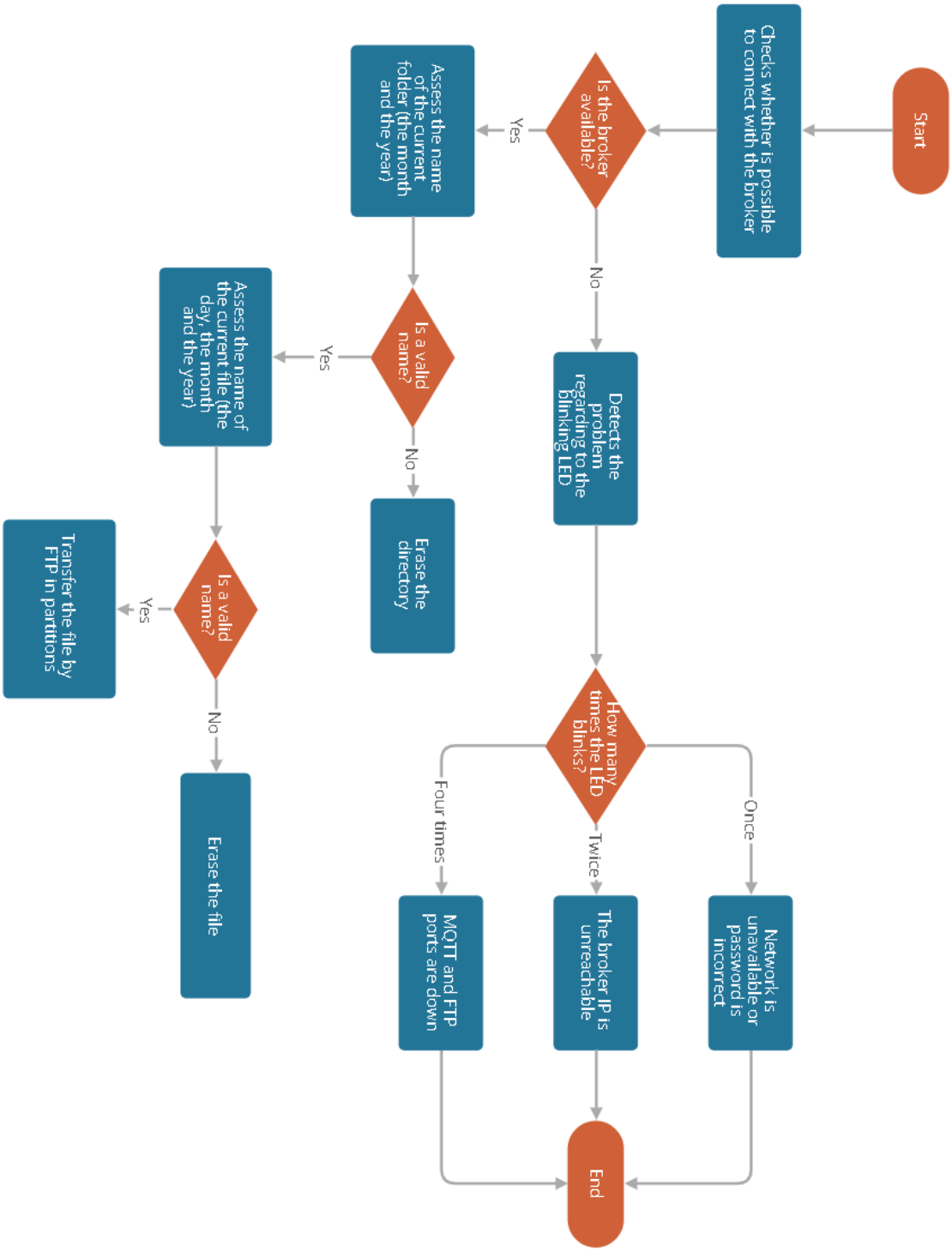
Regular Mode:



Setting Mode:

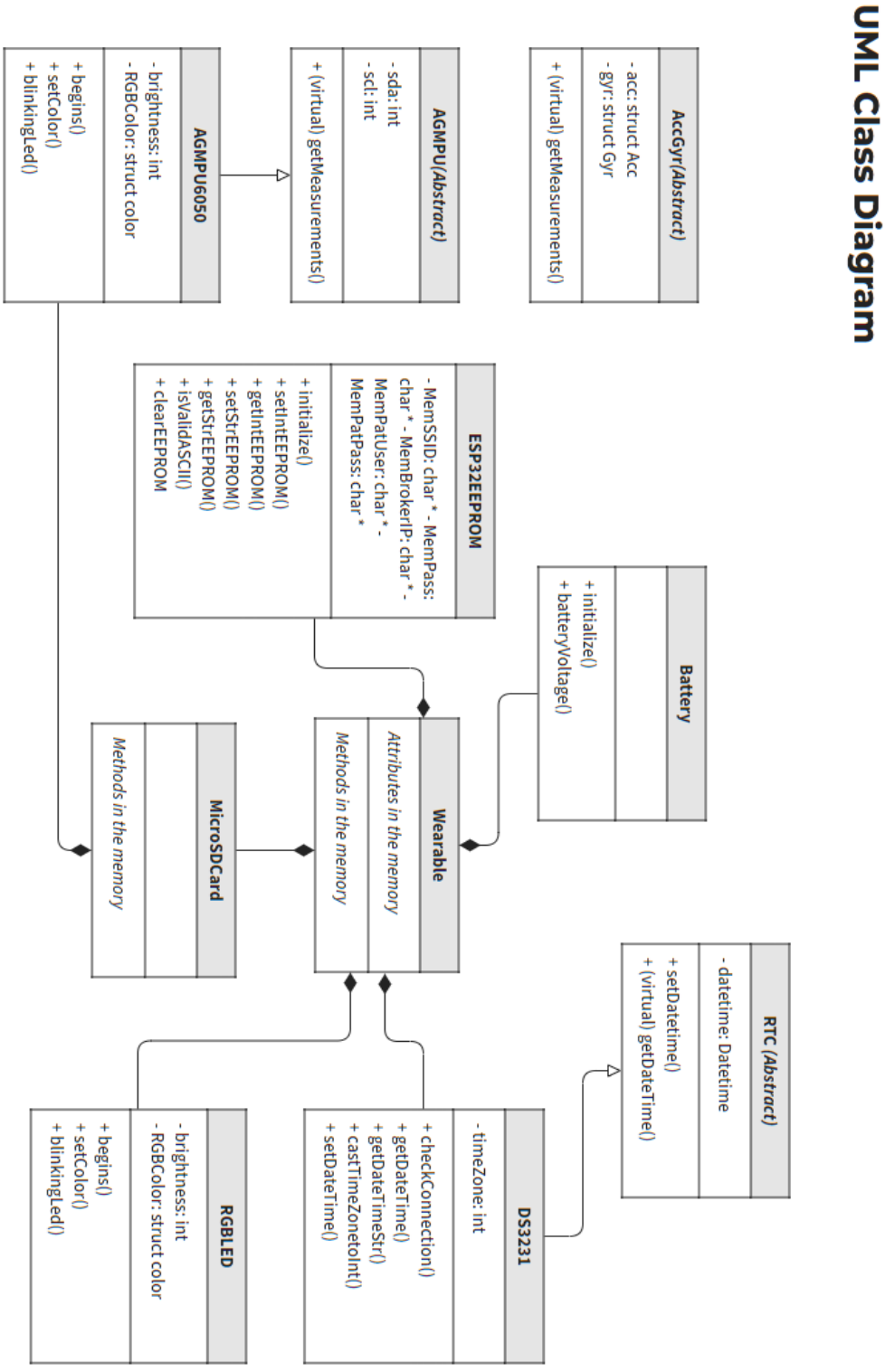


Transfering Mode:



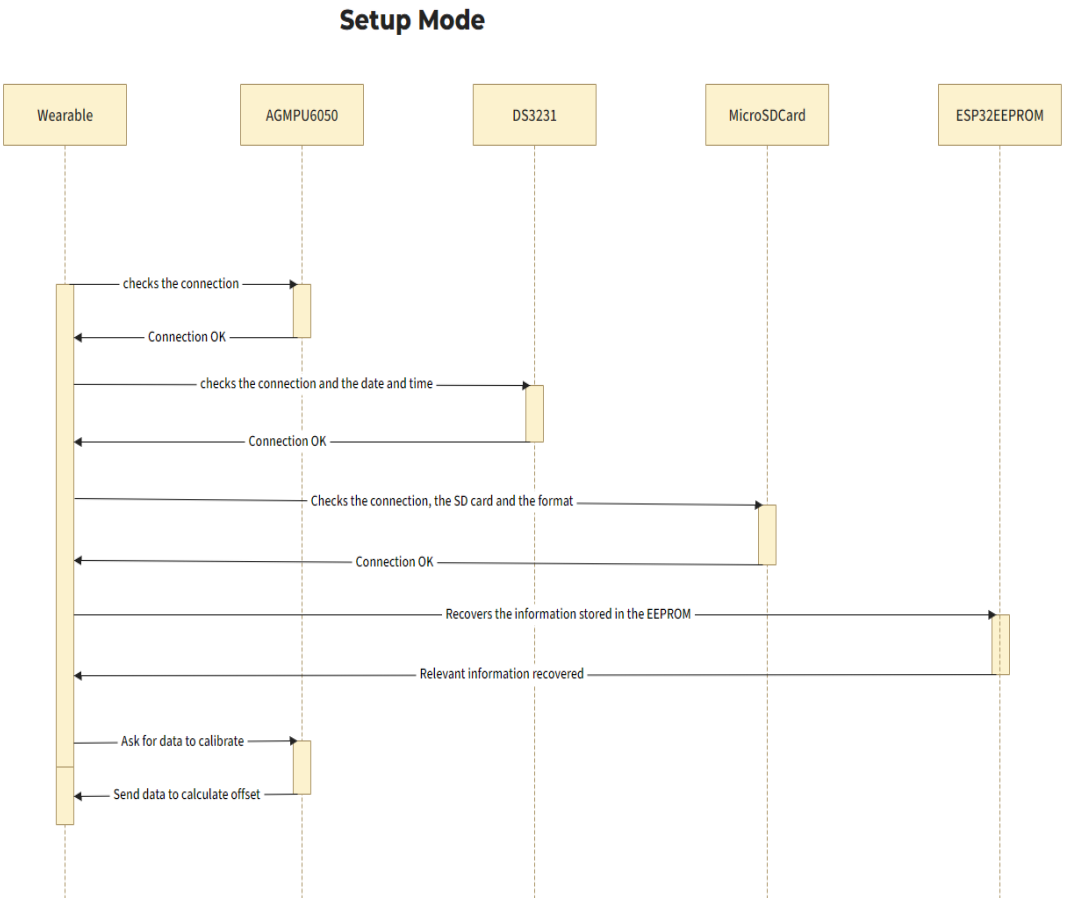
5.4.2 Class diagram

[36]

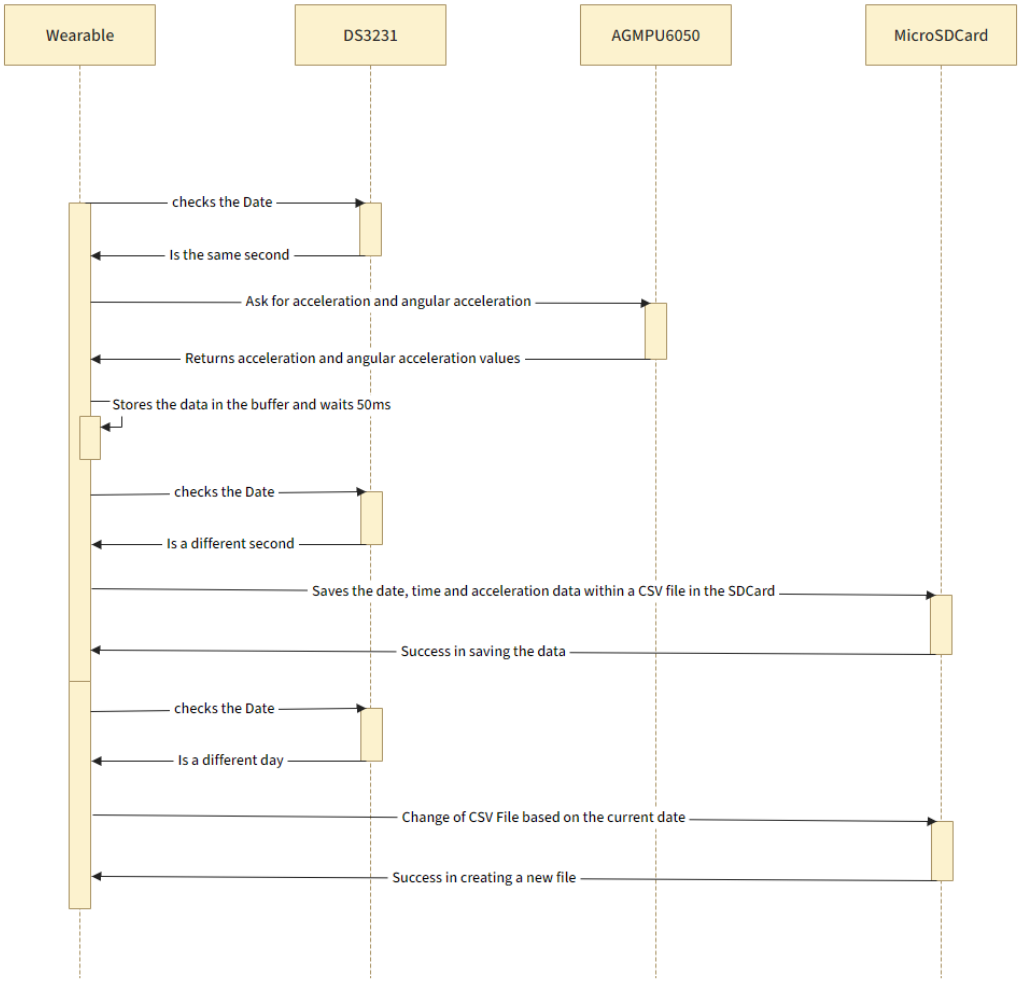


The definition of the class diagram has been based on the physical components of the wearable, the attributes are defined corresponding to the principal features of each component, and the methods are based on the behaviour of the component. For instance, the MicroSDCard Class has as attributes the GPIO pins and the total and used space on GB, and as methods has the order from the main microprocessor of creating directories, list all created directories, remove an existing directory, and creating, reading, writing, or deleting a specific file.

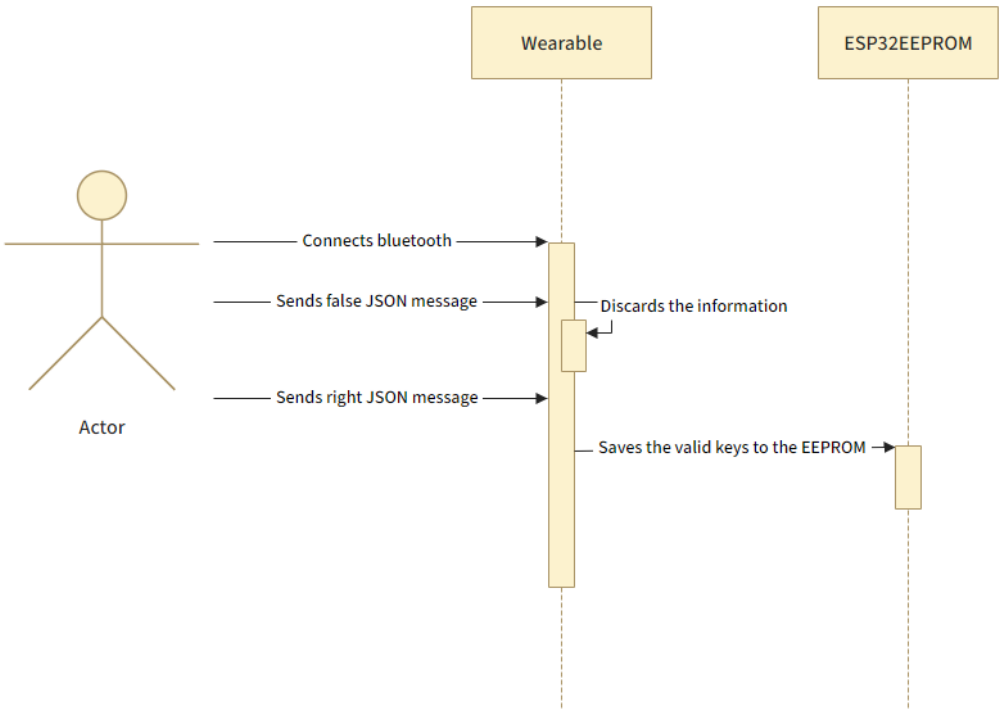
According to the interaction among different classes the sequence diagrams could be the following:



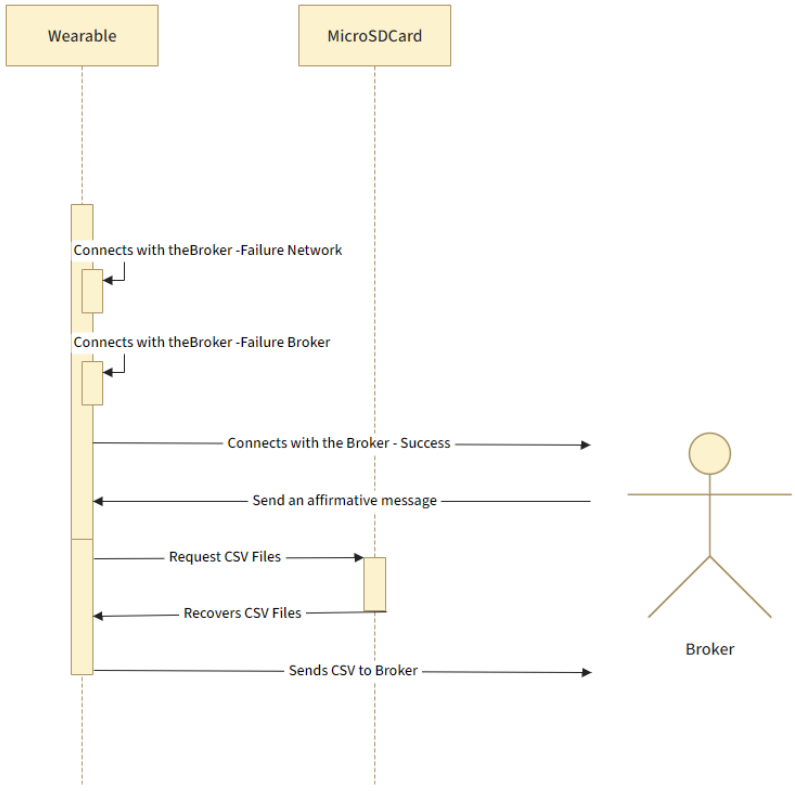
Regular Mode



Setting Mode



Transferring Mode



5.4.3 Source code

The source code is Arduino, which is made of C/C++ programming language, with some of Java methods and attributes built in the Arduino IDE Platform. The full project repository is available in this link:

[37]

5.4.3.1 Files

The files are divided into the .h files, where the classes are defined into this, and the .cpp/.ino, where the methods and attributes, are defined. Here there is the file structure of the project:

- AccGyr.h
- AGMPU.h
- AGMPU6050.h
- AGMPU6050.ino
- Basic.h
- Battery.h
- Battery.ino
- Button.h
- Button.ino
- DS3231.h
- DS3231.ino
- ESP32EEPROM.h
- ESP32EEPROM.ino
- ESP32Ping.cpp
- ESP32Ping.h
- Firmware.h
- I2Cdev.cpp
- I2Cdev.h
- IMU_Wearable_Class.ino
- MicroSDCard.h
- MicroSDCard.ino
- MPU6050.cpp
- MPU6050.h
- Ping.cpp
- Ping.h
- RGBLED.h
- RGBLED.ino
- RTC.h
- RTC.ino
- Wearable.h
- Wearable.ino

The files are grouped in levels of abstraction, where the independent files are grouped in level 0, the files dependent of files with level 0 are grouped in level 1, and so on. By doing this, the organization of the project will be much more organized and well-dependent.

- *LEVEL 0:*

File	Firmware.h
Description	File referred to the firmware of the device and the necessary declaration regarding

Necessary Files	NONE
Dependent Files	- Basics.h
Includes	- Arduino.h - Wire.h - SPI.h - BluetoothSerial.h - Esp_bt_main.h - Esp_bt_device.h - WiFi.h - PubSubClient.h - ESP32_FTPClient.h
Defines	- FIRMWARE_H
Classes	NONE

- *LEVEL 1:*

File	Basics.h
Description	File referred to the basic data for the correct operation of the device. It describes the necessary definitions to log the task within, the operating mode (Development, Testing and Production) and all needed information regarding that, including the GPIO inputs and outputs.
Necessary Files	- Firmware.h
Dependent Files	- AGMPU.h - Battery.h - Button.h - DS3231.h - ESP32EEPROM.h - MicroSDCard.h - RGBLED.h - RTC.h
Includes	NONE
Defines	- _WINDOWS_32_ - _WINDOWS_64_ - _LINUX_32_ - _LINUX_64_ - OS - BYTE_BUFFER_LENGTH - WORD_BUFFER_LENGTH - DWORD_BUFFER_LENGTH - QWORD_BUFFER_LENGTH - TINY_BUFFER_LENGTH - SMALL_BUFFER_LENGTH - MEDIUM_BUFFER_LENGTH

	- LARGE_BUFFER_LENGTH - HUGE_BUFFER_LENGTH - MAXIMUM_BUFFER_LENGTH - I2C_SDA - I2C_SCL - SPI_SCK - SPI_MISO - SPI_MOSI - SPI_CDSC - V_BATT_ADC - CHG_IND - RGB_LED - GPIO_PUSH_BUTTON - WEARABLE_NONE_LOGGING - WEARABLE_SERIAL_LOGGING - WEARABLE_BLUETOOTH_DEVELOPMENT_MASTER_LOGGING - WEARABLE_BLUETOOTH_DEVELOPMENT_SLAVE_LOGGING - WEARABLE_BLUETOOTH_TESTING_LOGGING - WEARABLE_MICRO_SD_LOGGING - WEARABLE_WIFI_LOGGING - LOGGING_MODE - BAUD_RATE - NAME_BLUETOOTH_DEVICE - FILE_NAME - WEARABLE_SET_MONITORING_LOG - WEARABLE_SET_SERIAL_RXTX - TRACE_LOGGING
Classes	NONE

- *LEVEL 2:*

File	AGMPU.h
Description	File referred to the abstract class AGMPU accelerometers and gyroscopes
Necessary Files	- Basics.h
Dependent Files	- AGMPU6050.h
Includes	NONE
Defines	- ACCGYR_AX - ACCGYR_AY - ACCGYR_AZ - ACCGYR_GX - ACCGYR_GY - ACCGYR_GZ
Classes	AccGyr (Abstract)

File	RTC.h
Description	Abstract class referred to the general use of the Real Time Clock (RTC)
Necessary Files	- Basics.h
Dependent Files	- DS3231.h - RTC.ino
Includes	- RTClib.h
Defines	- VAR_YEAR - VAR_MONTH - VAR_DAY - VAR_HOUR - VAR_MINUTE - VAR_SECOND
Classes	RTC (Abstract)

File	RTC.ino
Description	Definition of non-abstract methods for the RTC class
Necessary Files	- RTC.h
Methods	- setDateTime - getDateTime

- *LEVEL 3:*

File	AGMPU6050.h
Description	File referred to the abstract class AGMPU accelerometers and gyroscopes
Necessary Files	- Basics.h
Dependent Files	- Wearable.h
Includes	NONE
Defines	- ACCGYR_AX - ACCGYR_AY - ACCGYR_AZ - ACCGYR_GX - ACCGYR_GY - ACCGYR_GZ
Classes	AccGyr (Abstract)

File	AGMPU6050.ino
Description	Defines the public methods for the class MPU6050 to ensure the correct behaviour of the accelerometer
Necessary Files	- AGMPU6050.h
Methods	- initialize - checkConnection - getMeasurements

File	DS3231.h
Description	Class declaration regarding the main behaviour of the RTC model DS3231, including the current date and time stored within the controller
Necessary Files	- RTC.h
Dependent Files	- Wearable.h - DS3231.ino
Includes	NONE
Defines	- DS3231_I2C_CONNECTED - DS3231_I2C_DISCONNECTED - TIMEZONE_MINUS_12 - TIMEZONE_MINUS_11 - TIMEZONE_MINUS_10 - TIMEZONE_MINUS_0930 - TIMEZONE_MINUS_09 - TIMEZONE_MINUS_08 - TIMEZONE_MINUS_07 - TIMEZONE_MINUS_06 - TIMEZONE_MINUS_05 - TIMEZONE_MINUS_04 - TIMEZONE_MINUS_0330 - TIMEZONE_MINUS_03 - TIMEZONE_MINUS_02 - TIMEZONE_MINUS_01 - TIMEZONE_GMT - TIMEZONE_PLUS_01 - TIMEZONE_PLUS_02 - TIMEZONE_PLUS_03 - TIMEZONE_PLUS_0330 - TIMEZONE_PLUS_04 - TIMEZONE_PLUS_0430 - TIMEZONE_PLUS_05 - TIMEZONE_PLUS_0530 - TIMEZONE_PLUS_0545 - TIMEZONE_PLUS_06 - TIMEZONE_PLUS_0630 - TIMEZONE_PLUS_07 - TIMEZONE_PLUS_08 - TIMEZONE_PLUS_0845

	- TIMEZONE_PLUS_09 - TIMEZONE_PLUS_0930 - TIMEZONE_PLUS_10 - TIMEZONE_PLUS_1030 - TIMEZONE_PLUS_11 - TIMEZONE_PLUS_12 - TIMEZONE_PLUS_1245 - TIMEZONE_PLUS_13 - TIMEZONE_PLUS_14
Classes	DS3231

File	DS3231.ino
Description	Defines the public methods for the use of the DS3231
Necessary Files	- DS3231.h
Methods	- checkConnection - setDateTime - getDateTime - getDateTimeStr - catTimeZoneToInt

File	Battery.h
Description	Main class responsible for the functioning of the battery
Necessary Files	- Basics.h
Dependent Files	- Wearable.h - Battery.ino
Includes	NONE
Defines	- BATTERY_INITIALIZE_FUNCTION_MASK - BATTERY_INITIALIZE_TRACE1 - GET_BATTERY_VOLTAGE_FUNCTION_MASK - GET_BATTERY_VOLTAGE_TRACE1
Classes	Button

File	Battery.ino
Description	Defines the public methods for the use of the battery
Necessary Files	- Battery.h
Methods	- Initialize - batteryVoltage

File	Button.h
Description	Main class regarding of the functioning of the button

Necessary Files	- Basics.h
Dependent Files	- Wearable.h - Button.ino
Includes	NONE
Defines	- DEFAULT_WF_BTH_TIME_MS - RISING_EDGE - FALLING_EDGE - NO_EDGE - NOT_EXCEEDED_LIMIT - PREVIOUS_STATE_LOW - INVALID_INTEGER
Classes	Button

File	Button.ino
Description	Defines the public methods for the use of the button
Necessary Files	- Button.h
Methods	- Button (Constructor) - edgeDetection - pushedLimitButton

File	ESP32EEPROM.h
Description	Main class regarding the internal EEPROM and its methods for reading and writing
Necessary Files	- Basics.h
Dependent Files	- Wearable.h - ESP32EEPROM.ino
Includes	- EEPROM.h
Defines	- ESP32_EEPROM_SIZE - ESP32_EEPROM_DATA_LENGTH - NOT_READABLE_ASCII_CHAR - READABLE_ASCII_CHAR - EEPROM_ADDR_NET_SSID - EEPROM_ADDR_NET_PASS - EEPROM_ADDR_BROKER_IP - EEPROM_ADDR_BROKER_PASS - EEPROM_ADDR_BTH_TIME_ACTIVE - EEPROM_ADDR_OTA_UPDATE_CHECK
Classes	ESP32EEPROM

File	ESP32EEPROM.ino
Description	Defines the public methods for the use of the ESP32 internal EEPROM to read and write

Necessary Files	- ESP32EEPROM.h
Methods	- initialize - setIntEEPROM - getIntEEPROM - setStrEEPROM - getStrEEPROM - isValidASCII - clearEEPROM

File	MicroSDCard.h
Description	Main class regarding the SD-Card Module and the micro-SD card attached to it.
Necessary Files	- Basics.h
Dependent Files	- Wearable.h - MicroSDCard.ino
Includes	- FS.h - SD.h
Defines	- MICROSD_MMC - MICROSD_SD - MICROSD_SDHC - MICROSD_UNKNOWN - MICROSD_CARD_UPDATE_DIR
Classes	MicroSDCard

File	MicroSDCard.ino
Description	Defines the public methods for the correct use and behaviour of the Micro-SD Card Module
Necessary Files	- MicroSDCard.h
Methods	- initialize - listDir - createDir - deleteDir - getFileByIndex - readFile - appendFile - renameFile - checkFile - deleteFile

File	RGBLED.h
Description	Main class regarding the RGB LED WS2812 and its main features and behaviours

Necessary Files	- Basics.h
Dependent Files	- RGBLED.ino - Wearable.h
Includes	- Adafruit_NeoPixel.ino
Defines	- GPIO_RGB_LED - GPIO_RGB_BRIGHTNESS - RGB_LED_NONE_COLOR - RGB_LED_RED_COLOR - RGB_LED_ORANGE_COLOR - RGB_LED_YELLOW_COLOR - RGB_LED_GREEN_COLOR - RGB_LED_BLUE_COLOR - RGB_LED_INDIGO_COLOR - RED_COLOR_RED_BYTE - RED_COLOR_GREEN_BYTE - RED_COLOR_BLUE_BYTE - ORANGE_COLOR_RED_BYTE - ORANGE_COLOR_GREEN_BYTE - ORANGE_COLOR_BLUE_BYTE - YELLOW_COLOR_RED_BYTE - YELLOW_COLOR_GREEN_BYTE - YELLOW_COLOR_BLUE_BYTE - GREEN_COLOR_RED_BYTE - GREEN_COLOR_YELLOW_BYTE - GREEN_COLOR_BLUE_BYTE - BLUE_COLOR_RED_BYTE - BLUE_COLOR_GREEN_BYTE - BLUE_COLOR_BLUE_BYTE - INDIGO_COLOR_RED_BYTE - INDIGO_COLOR_GREEN_BYTE - INDIGO_COLOR_BLUE_BYTE
Classes	RGBLED

File	RGBLED.ino
Description	Defines the public methods for the correct use of the RGB LED, including the colour change and the brightness
Necessary Files	- RGBLED.h
Methods	- Begins - setColor - blinkingLed

- *LEVEL 4:*

File	Wearable.h
Description	Main Class Wearable definition responsible for the general functioning of the device
Necessary Files	- AGMPU6050.h - Battery.h - RGBLED.h - DS3231.h - Button.h - MicroSDCard.h - ESP32EEPROM.h
Dependent Files	- Wearable.ino - IMU_Wearable_Class.ino
Includes	- ArduinoJson.h - ESP32Ping.h
Defines	- NAME_WIFI_DEVICE - WIFI_MQTT_CONN_PORT - WIFI_FTP_TIMEOUT_CONN - WIFI_SSH_USER_REMOTE_CONN - WIFI_SSH_PASS_REMOTE_CONN - WEARABLE_FTP_REMOTE_UPDATE_MAX_SIZE_PART - WIFI_DEVICE_STATIC_IP_1 - WIFI_DEVICE_STATIC_IP_2 - WIFI_DEVICE_STATIC_IP_3 - WIFI_DEVICE_STATIC_IP_4 - WIFI_DEVICE_GATEWAY_1 - WIFI_DEVICE_GATEWAY_2 - WIFI_DEVICE_GATEWAY_3 - WIFI_DEVICE_GATEWAY_4 - WIFI_DEVICE_SUBNET_1 - WIFI_DEVICE_SUBNET_2 - WIFI_DEVICE_SUBNET_3 - WIFI_DEVICE_SUBNET_4 - WIFI_MQTT_QUEUE_PUB_LOGIN - WIFI_MQTT_QUEUE_PUB_UPDATES - WIFI_MQTT_QUEUE_SUB_LOGIN - WIFI_MQTT_QUEUE_SUB_DATETIME - WIFI_MQTT_TEST_MESSAGE - WIFI_UPDATE_VERSION_FILE - WIFI_JSON_KEY_VERSION - JSON_KEY_UPDATED - IMU_WEARABLE_FILE_TITLE - IMU_WEARABLE_FILE_NAME_COLUMNS
Classes	Wearable

File	Wearable.ino
Description	Defines the public methods for the Main Class Wearable
Necessary Files	- Wearable.h
Methods	- bluetoothCallback

	- isValidASCII - setup - loop - chargingBattery - notChargingBattery - regularMode - collectData - storeData - offsetTransform - coordinateAxisChange - settingMode - transferringMode - delayingAction - checkHardwareConnection - recoverEEPROMMemoryData - storeDataIntoEEPROMMemory - autocalibration - msgUpdateToJSON - castJSONToAttributes - castAttributesToJSON - castJSONToDateTime - checkOTAUpdate - recvUpdateFromBroker - WiFiConnect - WiFiDisconnect - wifiCallback - castJSONIPVerToString - sendMessageToBroker - recvMessageFromBroker - analyzeMsgUpdFromBroker - analyzeMsgUpdPartitionFromBroker - receiveBluetoothMsg - receiveBluetoothAsyncMsg - sendBluetoothMsg

- *LEVEL 5:*

File	IMU_Wearable_Class.ino
Description	Main File. Defines the setup and loops for the firmware
Necessary Files	- Wearable.h
Methods	NONE

5.4.3.2 Classes

- ***BATTERY:***

Class	**Battery**
File	Battery.h
Line	45-82
Description	Class that defines the behaviour of the Battery
Attributes	NONE
Methods	- Initialize - batteryVoltage
Abstract	NO

Method	Initialize
File	Battery.ino
Line	----------
Description	Initialize the GPIO pin V_BATT_ADC and CHG_IND to measure the battery voltage and the pin of the Charger Module
Inputs	NONE
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- BATTERY_INITIALIZE_FUNCTION_MASK - BATTERY_INITIALIZE_TRACE1

Method	batteryVoltage
File	Battery.ino
Line	----------
Description	Measures the voltage of the battery in order to see the total charge of it. The function takes a number between 0 and 4095 (0-0V; 4095-3.3V)
Inputs	NONE
Output	Float: The voltage measured in the voltage divider
Dependent Methods	NONE
Dependent Attributes	NONE

Traces	- BATTERY_VOLTAGE_FUNCTION_MASK - BATTERY_VOLTAGE_TRACE1

- ***ESP32EEPROM:***

Class	**ESP32EEPROM**
File	ESP32EEPROM.h
Line	122-143
Description	Class that defines the behaviour of the internal EEPROM of the ESP32
Attributes	- MemSSID (char *): Network SSID recovered from the EEPROM used to connect with the home Wi-Fi. - MemPass (char *): Network SSID recovered from the EEPROM used to connect with the home Wi-Fi. - MemBrokerIP (char *): IP regarding the broker (Raspberry Pi recovered from the EEPROM used to connect with it. - MemPatUser (char *): Username used in the remote server to login, via broker. - MemPatPass (char *): Password used in the remote server to login, via broker.
Methods	- Initialize - setIntEEPROM - getIntEEPROM - setStrEEPROM - getStrEEPROM - isValidASCII - clearEEPROM
Abstract	NO

Method	Initialize
File	ESP32EEPROM.ino
Line	40-49
Description	Initialize the internal EEPROM Memory from the ESP32 and reserves the size passed as argument
Inputs	- EEPRsize (int): EEPROM size intended to reserve.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - INITIALIZE_FUNCTION_MASK

Method	setIntEEPROM
File	ESP32EEPROM.ino
Line	71-97
Description	Tries to save the integer passed as argument within the address memory passed also as argument. For doing that, it saves each byte of the integer and stored them as individual bytes. If there is a failure trying to write, it sends a number minor than 0.
Inputs	- addr (int): EEPROM address intended to save. - Val (int *): Value intended to save within the address
Output	(int): - EEPROM_INT_WRITE_OK: The writing of the value within the EEPROM address occurred without errors. - EEPROM_INT_WRITE_INVALID_ADDRESS: The address passed as argument is not valid (minor than 0 or bigger than the maximum capacity)
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - SETINTEEPROM_FUNCTION_MASK

Method	getIntEEPROM
File	ESP32EEPROM.ino
Line	117-147
Description	Tries to recover the integer passed as argument from the address memory passed also as argument. For doing that, it recovers each byte of the integer within the EEPROM as individual bytes and after that makes an integer pointer to point the less significant of them. If there is a failure trying to read, it sends a number minor than 0.
Inputs	- addr (int): EEPROM address intended to recover. - Val (int *): Value intended to recover within the address
Output	(int): - EEPROM_INT_READ_OK: The reading of the value within the EEPROM address occurred without errors. - EEPROM_INT_READ_INVALID_ADDRESS: The address passed as argument is not valid (minor than 0 or bigger than the maximum capacity)
Dependent Methods	NONE

Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - GETINTEEPROM_FUNCTION_MASK

Method	setStrEEPROM
File	ESP32EEPROM.ino
Line	168-215
Description	Tries to save the string (char *) passed as argument within the address memory passed also as argument. For doing that, it converts each character from the array in a byte data format, based on ASCII code. After that, these ASCII characters are stored within the address passed as argument. If there is a failure trying to write, it sends a number minor than 0.
Inputs	- addr (int): EEPROM address intended to save. - Str (const char *): String intended to store within the address.
Output	(int): - EEPROM_STR_WRITE_OK: The writing of the string within the EEPROM address occurred without errors. - EEPROM_STR_WRITE_ERROR_INVALID_CHAR: One of the characters passed in the array is not a readable ASCII character (minor than 32 or bigger than 126). - EEPROM_STR_WRITE_ERROR_LENGTH_OVERFLOW: The length of the array of characters is bigger than the maximum established by default (in this case 32). - EEPROM_STR_WRITE_ERROR_INVALID_ADDRESS: The address passed as argument is not valid (minor than 0 or bigger than the maximum capacity)
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - SETSTREEPROM_FUNCTION_MASK

Method	getStrEEPROM
File	ESP32EEPROM.ino
Line	235-291
Description	Tries to recover the string (char *) passed as argument, with the length also passed as argument within the address memory passed as argument. For doing that, it recovers each byte data from the EEPROM memory and then transform them into readable ASCII character. After this it joins each character into

	the array format. If there is a failure trying to read, it sends a number minor than 0.
Inputs	- addr (int): EEPROM address intended to read from. - Str (const char *): String intended to recover within the address. - Strlength: Length of the string.
Output	(int): - EEPROM_STR_WRITE_OK: The reading of the string within the EEPROM address occurred without errors. - EEPROM_STR_WRITE_ERROR_EMPTY_ADDRESS: The address passed as argument is empty. - EEPROM_STR_WRITE_ERROR_LENGTH_OVERFLOW: The length of the array of characters is bigger than the maximum established by default (in this case 32). - EEPROM_STR_WRITE_ERROR_INVALID_ADDRESS: The address passed as argument is not valid (minor than 0 or bigger than the maximum capacity)
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - GETSTREEPROM_FUNCTION_MASK

Method	isValidASCII
File	ESP32EEPROM.ino
Line	386-410
Description	Evaluates whether the character passed as argument is a valid ASCII character or not.
Inputs	- c (char): character intended to evaluate.
Output	(int): - READABLE_ASCII_CHAR: The char is readable is ASCII format. - NOT_READABLE_ASCII_CHAR: The char is not readable in ASCII format.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- ESP32EEPROM_LOG_MASK - ISVALIDASCII_FUNCTION_MASK

Method	clearEEPROM

File	ESP32EEPROM.ino
Line	415-422
Description	Erases all EEPROM memory from 0 address to the address passed as argument.
Inputs	- EEPRsize (int): the size of the EEPROM intended to erase.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	NONE

- ***RTC:***

Class	**RTC**
File	RTC.h
Line	72-87
Description	Abstract class to englobe the normal behaviour of each Real Time Clock (RTC)
Attributes	- Datetime (DateTime): date and time stored within the RTC in a special format. Is a protected attribute.
Methods	- setDateTime - (virtual) getDatetime (int) - (virtual) getDatetime (DateTime)
Abstract	YES

Method	setDatetime
File	RTC.ino
Line	41-52
Description	Sets the date and time of the RTC regarding the UNIX time passed as argument and the Time Zone also passed as argument
Inputs	- unixTime (int): the time in UNIX format (number of seconds since 00:00:00 1 Jan 1970). - GMT (float): Global time zone in float format
Output	NONE
Dependent Methods	NONE

Dependent Attributes	- Datetime
Traces	- RTC_LOG_MASK - SETDATETIME_FUNCTION_MASK

- ***DS3231:***

Class	**DS3231**
File	DS3231.h
Line	57-146
Description	Class to define the normal behaviour of the RTC DS3231.
Attributes	- Timezone (int): The global zone where the user is located
Methods	- checkConnection - getDateTime - getDateTimeStr - castTimeZoneToInt - setDateTime
Abstract	NO

Method	checkConnection
File	DS3231.ino
Line	43-75
Description	Check the physical connection between the RTC and the main microcontroller, by I2C connection.
Inputs	NONE
Output	(int): - DS3231_I2C_CONNECTED: The main microcontroller has detected the I2C slave for the RTC. - DS3231_I2C_DISCONNECTED: The main microcontroller has not detected the I2C slave for the RCT.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- DS3231_LOG_MASK - CHECKCONNECTION_FUNCTION_MASK

Method	setDatetime
File	DS3231.ino

Line	292-320
Description	Establish a new date and time with the unix time passed as argument
Inputs	- UNIXtime (int): the new time in UNIX format
Output	(int): - SETDATETIME_VALID_UNIXTIME: The time passed as argument mixed with the global time zone is valid. - SETDATETIME_INVALID_UNIXTIME: The time passed as argument is invalid (minor than 0 or bigger than the maximum possible Integer.
Dependent Methods	NONE
Dependent Attributes	- datetime
Traces	- SETDATETIME_FUNCTION_MASK - SETDATETIME_TRACE1 - SETDATETIME_TRACE2

Method	getDatetime
File	DS3231.ino
Line	220-287
Description	Gets the variable passed as argument to the main microcontroller.
Inputs	- Var: The variable intended to recover (1-Year; 2-Month; 3-Day; 4-Hour; 5-Minute; 6-Second).
Output	(int): The current year, month, day, hour, minute or second stored within the RTC.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- DS3231_LOG_MASK - GETDATETIME_FUNCTION_MASK

Method	getDatetimeStr
File	DS3231.ino
Line	80-87
Description	Gets the date and time in a String format instead of an object format.
Inputs	- Str (char *): The String to store the date and time in a character format to return.

	- sep (char): the character that the user prefer to use as a separator between date and time.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	NONE

Method	castTimeZoneToInt
File	DS3231.ino
Line	121-199
Description	Converts the time zone passed as string to an Integer. If the String is not a valid timezone, it returns a number minor than 0
Inputs	- StrTZ (char *): The String Time Zone intended to convert into an integer format
Output	(int): - CASTTIMEZONETOINT_VALID_TIMEZONE: The String passed as argument is a valid time zone. - CASTTIMEZONETOINT_INVALID_TIMEZONE: The String passed as argument is not a valid time zone
Dependent Methods	NONE
Dependent Attributes	- timezone
Traces	- CASTTIMEZONETOINT_FUNCTION_MASK - CASTTIMEZONETOINT_TRACE1 - CASTTIMEZONETOINT_TRACE2 - CASTTIMEZONETOINT_TRACE3 - CASTTIMEZONETOINT_TRACE4 - CASTTIMEZONETOINT_TRACE5

- ***BUTTON:***

Class	**Button**
File	Button.h
Line	72-82
Description	Class to define the functioning of the button object.
Attributes	NONE
Methods	- edgeDetection

	- pushedLimitButton
Abstract	NO

Method	edgeDetection
File	Button.ino
Line	62-91
Description	Returns whether either rising or falling edge has been detected or not.
Inputs	NONE
Output	(int): - NO_EDGE: There has not been a change of state of the button. - RISING_EDGE: There has been a rising edge detected. - FALLING_EDGE: There has been a falling edge detected.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- BUTTON_LOG_MASK - EDGEDETECTION_FUNCTION_MASK

Method	pushedLimitButton
File	Button.ino
Line	62-91
Description	Evaluates whether the button has been pushed more time than the time passed as arguments in ms or not
Inputs	- timeLimit_ms (int): The time intended to evaluate, in milliseconds, whether the button has been pushed that time
Output	(int): - NOT_EXCEEDED_LIMIT: The button has been pushed less time than the one passed as argument. - EXCEEDED_LIMIT: The button has been pushed longer time than the one passed as argument.
Dependent Methods	- edgeDetection
Dependent Attributes	NONE
Traces	- BUTTON_LOG_MASK

	- PUSHEDLIMITBUTTON_FUNCTION_MASK

- ***RGBLED:***

Class	**RGBLED**
File	RGBLED.h
Line	389-403
Description	Class to define the RGB colours and its functioning
Attributes	- Brightness (int): The brightness of the RGB with an intensity from 0 to 256. - rgbcolor (struct RGBColor): The intensity of each byte for red, green and blue.
Methods	- begins - setColor - blinkingLed
Abstract	NO

Method	begins
File	RGBLED.ino
Line	40-52
Description	Sets up the PWM modules to start the colouring of the LED.
Inputs	NONE
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- RGBLED_LOG_MASK - SETINTBRIGHTNESS_FUNCTION_MASK

Method	setColor
File	RGBLED.ino
Line	70-166
Description	Switf the colour of the LED to the one passed as argument. If other unknown colour is passed as argument, the RGB LED is switched off.
Inputs	- Color (int): The color intended to swift. (0-None; 1-Red; 2-Orange; 3-Yellow; 4-Green; 5-Blue; 6-Indigo).

Output	NONE
Dependent Methods	NONE
Dependent Attributes	- rgbcolor
Traces	- RGBLED_LOG_MASK - SETINTBRIGTHNESS_FUNCTION_MASK

Method	blinkingLed
File	RGBLED.ino
Line	185-219
Description	Makes the LED blink the number of times passed as argument, with the period also passed as argument.
Inputs	- Ncycles (int): The number of blinks needed. - Msxcycle (int): The time per each cycle, in milliseconds.
Output	NONE
Dependent Methods	- setColor
Dependent Attributes	- rgbcolor
Traces	- RGBLED_LOG_MASK - BLINKINGLED_FUNCTION_MASK

- ***ACCGYR:***

Class	**AccGyr**
File	AccGyr.h
Line	86-96
Description	Abstract class related to general use of the accelerometers.
Attributes	- acc (Struct Acc): Struct consisting of the 3 Axis accelerometers. - gyr (Struct Gyr): Struct consist of the 3 Axis gyroscopes.
Methods	- (abstract) getMeasurements (int)
Abstract	YES

- ***AGMPU:***

Class	**AGMPU**

File	AGMPU.h
Line	45-55
Description	Abstract class related to general use of the family of MPU accelerometers, which includes especially MPU6050 and MPU 9250.
Attributes	- Sda (int): Pin related to the Sda I2C communication. - Scl (int): Pin related to the Scl I2C communication.
Methods	- (abstract) getMeasurements (int)
Abstract	YES

- ***AGMPU6050:***

Class	**AGMPU6050**
File	AGMPU6050.h
Line	45-55
Description	Class related to the main behaviour of the accelerometer MPU6050.
Attributes	NONE
Methods	- Initialize (void) - checkConnection (int) - getMeasurements (float)
Abstract	NO

Method	initialize
File	AGMPU6050.ino
Line	39-85
Description	Initialize AGMPU6050 accelerometer, by sending I2C slave address established by default. After that, it establishes the range of the accelerometer and gyroscope.
Inputs	NONE
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- INITIALIZE_FUNCTION_MASK - INITIALIZE_TRACE1 - INITIALIZE_TRACE2 - INITIALIZE_TRACE3

Method	checkConnection
File	AGMPU6050.ino
Line	104-124
Description	Checks the connection between the master microcontroller and the slave connected to the accelerometer.
Inputs	NONE
Output	(int): - MPU6050_I2C_CONNECTED: The accelerometer is connected by I2C wires. - MPU6050_I2C_DISCONNECTED: The accelerometer is disconnected by I2C wires.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- CHECKCONNECTION_FUNCTION_MASK - CHECKCONNECTION_TRACE1 - CHECKCONNECTION _TRACE2 - CHECKCONNECTION _TRACE3

Method	getMeasurements
File	AGMPU6050.ino
Line	104-124
Description	Get the corresponding number of measurements, passed by an index as argument. After that it returns an array of measurements, beginning by the index passed.
Inputs	- nMeasurements (int *): A reference to the number of total variables stored within the accelerometer.
Output	(float *): The array of floats as a result of transforming the integers obtained from the registers corresponding to each variable.
Dependent Methods	NONE
Dependent Attributes	- acc - gyr
Traces	- GETMEASUREMENTS_FUNCTION_MASK - GETMEASUREMENTS _TRACE1 - GETMEASUREMENTS _TRACE2 - GETMEASUREMENTS _TRACE3

- ***MICROSDCARD:***

Class	**MicroSDCard**
File	MicroSDCard.h
Line	50-245
Description	Class to describe the MicroSDCard functioning of creating, reading, writing, appending and deleting files and directories within the SD card.
Attributes	NONE
Methods	- Initialize (void) - createDir (int) - listDir (int) - getFileByIndx (int) - deleteDir (int) - readFile (int) - appendFile (int) - appendFile (int) (overloading) - renameFile (int) - checkFile (int) - deleteFile (int)
Abstract	NO

Method	initialize
File	MicroSDCard.ino
Line	45-68
Description	Initialize SPI communication if it was uninitialized and checks whether there is a module of Micro-SD Card embedded to SPI communication, or whether there is a Micro-SD card itself attached to this module or not.
Inputs	NONE
Output	(int): - MICROSD_CARD_MOD_CONNECTED: There is a Micro-SD Card module embedded and connected by SPI to the main microcontroller, and this module contains a readable and writable Micro-SD Card. - MICROSD_CARD_MOD_DISCONNECTED: There is neither a Micro-SD Card module connected by SPI to the main microcontroller, nor a Micro-SD Card attached to the module in case there were a module connected. If the Micro-SD Card has not the correct format (FAT32) or it has not rights of writing, this is also the return value.
Dependent Methods	NONE
Dependent Attributes	NONE

Traces	- INITIALIZE_FUNCTION_MASK - INITIALIZE _TRACE1 - INITIALIZE _TRACE2 - INITIALIZE _TRACE3

Method	createDir
File	MicroSDCard.ino
Line	92-122
Description	Creates a directory with the path passed as argument. If there is no possible to create such path, it returns an error value.
Inputs	- path (const char *): The absolute name of the new directory intended to create.
Output	(int): - CREATE_DIR_OK: The new directory has been created with success. - CREATE_DIR_ERROR: There has been no possibility of creating this new directory, either the previous route has not exist or the directory already exists.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- CREATEDIR_FUNCTION_MASK - CREATEDIR _TRACE1 - CREATEDIR _TRACE2 - CREATEDIR _TRACE3

Method	listDir
File	MicroSDCard.ino
Line	149-222
Description	Lists the number of files and directories stored within a route passed as argument and returns this number by a reference.
Inputs	- path (const char *): The absolute name of the route intended to check. - nFiles (int *): The reference of the number of files intended to return.
Output	(int): - MICROSD_CARD_LIST_DIR_OK: The route has been successfully read and the number of files and directories is valid:

	- MICROSD_CARD_LIST_DIR_ERROR_NOT_DIR: The route has not been identified with a directory, but a file. - MICROSD_CARD_LIST_DIR_ERROR_INVAL_PATH: The route does not exist.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- LISTDIR_FUNCTION_MASK - LISTDIR _TRACE1 - LISTDIR _TRACE2 - LISTDIR _TRACE3

Method	getFileByIndx
File	MicroSDCard.ino
Line	228-296
Description	Returns the name of the file or directory based on the path passed as argument and the indx passed as argument. It also returns by a reference whether this is a file or a directory. If the index is not valid, or the path does not exist, it returns an error value message.
Inputs	- path (const char *): The route intented to search the file by the index. - Indx: The index intended to search. - nameFile (char *): The reference of the name of the file or directory by the index. - isFile (int *): The reference to point whether the file is a directory or not.
Output	(int): - MICROSD_GET_FILE_INDX_OK_DIR: The path and the index passed as argument are both correct and the file found is a directory. - MICROSD_GET_FILE_INDX_OK_FILE: The path and the index passed as argument are both correct and the file found is not a directory but a normal file. - MICROSD_GET_FILE_ERROR_INVAL_INDX: The index passed as argument is out or range (minor than 0 or bigger than the number of files within this path). - MICROSD_GET_FILE_ERROR_INVAL_PATH: The path passed as argument is not found.
Dependent Methods	NONE
Dependent Attributes	NONE

Traces	- GETFILEBYINDX_FUNCTION_MASK - GETFILEBYINDX _TRACE1 - GETFILEBYINDX _TRACE2 - GETFILEBYINDX _TRACE3 - GETFILEBYINDX _TRACE4 - GETFILEBYINDX _TRACE5 - GETFILEBYINDX _TRACE6

Method	deleteDir
File	MicroSDCard.ino
Line	320-345
Description	Tries to delete a directory passed as an argument. If the directory does not exist or the directory exists but is not empty, the function returns an error value.
Inputs	- path (const char *): The directory intended to delete.
Output	(int): - MICROSD_CARD_DELETE_DIR_OK: The directory has been deleted with success. - MICROSD_CARD_DELETE_DIR_ERROR: The directory could not be deleted, either because the route does not exist, or the directory is not empty.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- DELETEDIR_FUNCTION_MASK - DELETEDIR _TRACE1 - DELETEDIR _TRACE2 - DELETEDIR _TRACE3

Method	readFile
File	MicroSDCard.ino
Line	375-465
Description	Tries to read a file from an absolute path passed as argument, from the lseek passed as argument within the file, and store them within the buff reference also passed as argument.
Inputs	- path (const char *): The file intended to read from. - Buff (char *): The buffer intended to store the information read from the file. - Lseek: (int): Value from where the file point must start to read.

Output	(int): - MICROSD_CARD_READ_FILE_OK_END: The buffer has reached the end of the file before the buffer is full. - MICROSD_CARD_READ_FILE_OK_NOT_END: The buffer has been filled before the file pointer reaches the end. - MICROSD_CARD_READ_FILE_ERROR_LSEEK_OV: The file pointer is bigger than the size of the file. - MICROSD_CARD_READ_FILE_ERROR_INVAL_PATH: The file does not exist or is located in another route.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- READFILE_FUNCTION_MASK - READFILE _TRACE1 - READFILE _TRACE2 - READFILE _TRACE3 - READFILE _TRACE4 - READFILE _TRACE5

Method	appendFile
File	MicroSDCard.ino
Line	492-540
Description	Appends a buffer passed as argument into a file also passes as argument by its absolute route. If the path does not exist or is an invalid type of file, it returns an error value.
Inputs	- path (const char *): The path of the file intended to append. - Buff (char *): The buffer intended to write within the file.
Output	(int): - MICROSD_CARD_APPEND_FILE_OK: The buffer has been written to the file with success. - MICROSD_CARD_APPEND_FILE_ERROR_INVAL_PATH: The path does not exist or is not a valid file. - MICROSD_CARD_READ_FILE_ERROR_BUFF_OVFL: The length of the buffer is bigger than the maximum allowed (4096).
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- APPENDFILE_FUNCTION_MASK - APPENDFILE _TRACE1

	- APPENDFILE _TRACE2 - APPENDFILE _TRACE3 - APPENDFILE _TRACE4

Method	appendFile (overloaded)
File	MicroSDCard.ino
Line	546-575
Description	Appends a buffer passed as argument into a file also passes as argument by its absolute route, but this time only the first length bytes, with length also a variable passed as argument. If the path does not exist or is an invalid type of file, it returns an error value.
Inputs	- path (const char *): The path of the file intended to append. - Buff (char *): The buffer intended to write within the file. - Length (int): The total number of bytes intended to write.
Output	(int): - MICROSD_CARD_APPEND_FILE_OK: The buffer has been written to the file with success. - MICROSD_CARD_APPEND_FILE_ERROR_INVAL_PATH: The path does not exist or is not a valid file. - MICROSD_CARD_READ_FILE_ERROR_BUFF_OVFL: The length of the buffer is bigger than the maximum allowed (4096).
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- APPENDFILE_FUNCTION_MASK_OVERLOADED - APPENDFILE _TRACE1_OVERLOADED - APPENDFILE _TRACE2_OVERLOADED - APPENDFILE _TRACE3_OVERLOADED - APPENDFILE _TRACE4_OVERLOADED

Method	appendFile (overloaded)
File	MicroSDCard.ino
Line	546-575
Description	Appends a buffer passed as argument into a file also passes as argument by its absolute route, but this time only the first length bytes, with length also a variable passed as argument. If the path does not exist or is an invalid type of file, it returns an error value.

Inputs	- path (const char *): The path of the file intended to append. - Buff (char *): The buffer intended to write within the file. - Length (int): The total number of bytes intended to write.
Output	(int): - MICROSD_CARD_APPEND_FILE_OK: The buffer has been written to the file with success. - MICROSD_CARD_APPEND_FILE_ERROR_INVAL_PATH: The path does not exist or is not a valid file. - MICROSD_CARD_READ_FILE_ERROR_BUFF_OVFL: The length of the buffer is bigger than the maximum allowed (4096).
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- APPENDFILE_FUNCTION_MASK_OVERLOADED - APPENDFILE _TRACE1_OVERLOADED - APPENDFILE _TRACE2_OVERLOADED - APPENDFILE _TRACE3_OVERLOADED - APPENDFILE _TRACE4_OVERLOADED

Method	renameFile
File	MicroSDCard.ino
Line	600-626
Description	Tries to rename the file passed as argument by the oldname (absolute path) to the newname also passed as argument.
Inputs	- oldName (char *): The old name of the file in absolute path. - newname (char *): The new name of the file in absolute path.
Output	(int): - MICROSD_CARD_RENAME_FILE_OK: The name of the file has been changed with success. - MICROSD_CARD_RENAME_FILE_ERROR: The name of the file could not be changed, either because the file does not exist or the route for the new name is not valid.
Dependent Methods	NONE
Dependent Attributes	NONE

Traces	- RENAMEFILE_FUNCTION_MASK - RENAMEFILE _TRACE1 - RENAMEFILE _TRACE2 - RENAMEFILE _TRACE3

Method	checkFile
File	MicroSDCard.ino
Line	650-679
Description	Checks whether the file passed as argument in absolute path exists or not.
Inputs	- path (char *): The absolute path of the file intended to check if exists.
Output	(int): - MICROSD_CARD_FOUND_FILE: The file exists in the route passed as argument. - MICROSD_CARD_NOT_FOUND_FILE: The file does not exist or is in another path.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- CHECKFILE_FUNCTION_MASK - CHECKFILE _TRACE1 - CHECKFILE _TRACE2 - CHECKFILE _TRACE3

Method	deleteFile
File	MicroSDCard.ino
Line	703-729
Description	Tries to delete a file passed as argument in absolute path.
Inputs	- path (char *): The absolute path of the file intended to delete.
Output	(int): - MICROSD_CARD_DELETE_FILE_OK: The file has been deleted with success. - MICROSD_CARD_DELETE_FILE_ERROR: The file was not able to be deleted, either because it does not exist or another reason.
Dependent Methods	NONE
Dependent Attributes	NONE

Traces	- DELETEFILE_FUNCTION_MASK - DELETEFILE _TRACE1 - DELETEFILE _TRACE2 - DELETEFILE _TRACE3

- ***WEARABLE:***

Class	**Wearable**
File	Wearable.h
Line	257-822
Description	Main class responsible to the general use of the device. It encompasses all classes before.
Attributes	- timeBegComp (int): time stamp for the beginning, in µs, to rest the delay of the time that the functions employ and make a more precise rate of gathering data. - timeComp (int): time stamp for the end, in µs, to rest the delay of the time that the functions employ and make a more precise rate of gathering data. - Wakeup_reason (esp_sleep_wakeup_reason): variable that stores the reason for the wakeup of the ESP32. - ESP32SSID (char *): variable corresponding to the network SSID recovered from the EEPROM object. - ESP32Password (char *): variable corresponding to the network password recovered from the EEPROM object. - ESP32BrokerIP (char *): variable corresponding to the Broker Static IP recovered from the EEPROM object. - ESP32PatUser (char *): variable corresponding to the User email or nick to login to the remote server. - ESP32PatPass (char *): variable corresponding to the User password to login to the remote server. - ESP32CurrDir (char *): The current path of the directory, in absolute path. - ESP32CurrCSVFile (char *): The current file where the microcontroller is writing at each moment. - buffSD (char *): The buffer of the data collected by the accelerometer, converted into an String that is passable to CSV format. - ESP32BinUpdate (char *): Name of the binary file to flash the device automatically within the future (not implemented yet). - sizeUpdate (int): the size of the binary file to flash the device automatically (not implemented yet). - timeActive (int): the time, in ms, stored within the EEPROM, that the Bluetooth mode remains active until it deactivates itself. - button (Button): The button object. - Ds3231 (DS3231): The RTC module. - rgbled (RGBLED): The RGB LED object. - Mpu6050 (AGMPU6050): The accelerometer module. - SDcard (MicroSDCard): The micro-SD card module.

	- Eeprom (ESP32EEPROM): The internal EEPROM of the microcontroller. - offsetAcc (struct Acc): The offset of the acceleration measurements taken at the beginning. - offsetGyr (struct Gyr): The offset of the angular acceleration measurements taken at the beginning. - Ax, ay, az, gx, gy, gz (float *): References of the variables to compensate the change of coordinate axis. - stampBegin (int): The first measurement of the milliseconds to compensate the offset. - nWiFiConn (int): The number of WiFi connections established since the device was powered up.
Methods	- isValidASCII (int) - setup (void) - loop (int) - chargingBattery (void) - notChargingBattery (void) - regularMode (void) - collectData (void) - storeData (void) - offsetTransform (void) - coordinateAxisChange (void) - settingMode (void) - transferingMode (void) - delayingAction (void) - checkHardwareConnections (int) - recoverEEPROMMemoryData (int) - storeDataIntoEEPROMMemory (int) - autocalibration (int) - msgUpdateToJSON (void) - castJSONtoAttributes (int) - castAttributestoJSON (int) - castJSONtoDatetime (int) - checkOTAUpdate (int) - recvUpdateFromBroker (int) - WiFiConnect (int) - WiFiDisconnect (int) - WiFiCallback (void) - castJSONIPVertoString (int) - sendMessageToBroker - recvMessageFromBroker - analyzeMsgUpdFromBroker - analyzeMsfUpdPartitionFromBroker - receiveBluetoothMsg (int) - receiveBluetoothAsyncMsg (int) - sendBluetoothMsg (int)
Abstract	NO

Method	isValidASCII
File	Wearable.ino

Line	41-59
Description	Check if a character is a valid ASCII character or not.
Inputs	- c (char): The character intended to check.
Output	(int): - ISVALIDASCII_VALID: The character is valid. - ISVALIDASCII_INVALID: The character is invalid.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- WEARABLE_LOG_MASK - ISVALIDASCII_FUNCTION_MASK

Method	setup
File	Wearable.ino
Line	64-326
Description	One of the two only methods that are public, along with the loop method. First it checks the physical and logical connections with the other components such the accelerometer, RTC and Micro-SD Card. Then, it checks whether there is the crucial information about the WiFi connection or not. If there are not, it automatically passes to the setting Mode and waits for the relevant data by Bluetooth, with no limit of time. If there are, it goes to the creation (if it does not exist) of the directory and file of the date and the first calibration of the accelerometer.
Inputs	NONE
Output	NONE
Dependent Methods	- checkHardwareConnections - recoverEEPROMMemoryData - receiveBluetoothMsg - castJSONtoAttributes - storeDataIntoEEPROMMemory - castJSONtoDatetime - autoCalibration
Dependent Attributes	- timeActive - nWiFiConn - rgbled → begins →setColor →blinkingLed - eeprom → initialize - rtcDs3231 → getDatetime - ESP32CurrDir - ESP32CurrCSVFile

	- SDCard → createDir →appendFile - stampBegin - timeBegComp
Traces	- SETUP_FUNCTION_MASK - SETUP_TRACE1 - SETUP_TRACE2 - SETUP_TRACE3 - SETUP_TRACE4 - SETUP_TRACE5 - SETUP_TRACE6 - SETUP_TRACE7 - SETUP_TRACE8 - SETUP_TRACE9 - SETUP_TRACE10 - SETUP_TRACE11 - SETUP_TRACE12 - SETUP_TRACE13 - SETUP_TRACE14 - SETUP_TRACE15 - SETUP_TRACE16 - SETUP_TRACE17 - SETUP_TRACE18 - SETUP_TRACE19 - SETUP_TRACE20 - SETUP_TRACE21

Method	loop
File	Wearable.ino
Line	332-366
Description	One of the two only methods that are public, along with the setup method. It checks whether the plug is connected to the device. If it is, activates the protocol of the charging battery. Otherwise, it activates the not charging battery mode.
Inputs	NONE
Output	NONE
Dependent Methods	- chargingBattery - delayingAction - notChargingBattery
Dependent Attributes	NONE
Traces	- LOOP_FUNCTION_MASK - LOOP_TRACE1 - LOOP_TRACE2 - LOOP_TRACE3

Method	chargingBattery
File	Wearable.ino
Line	370-380
Description	Read the value of the pin attached to the voltage divider of the battery and prints the value.
Inputs	NONE
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- WEARABLE_LOG_MASK - CHARGINGBATTERY_FUNCTION_MASK

Method	notChargingBattery
File	Wearable.ino
Line	384-459
Description	Checks whether the button has been pushed or not. If the button has been pushed, checks whether the time limit has been overpassed or not. If it has not been overpassed, the device is in setting (Bluetooth) mode. Otherwise, is in transferring (Wi-Fi) mode. If the button has not been pushed, it checks whether there has been a change of hour or not. If the hour is different from the previous one, it goes to transferring mode, otherwise, it is in regular mode (collecting data from the accelerometer).
Inputs	NONE
Output	NONE
Dependent Methods	- regularMode - settingMode - transferringMode
Dependent Attributes	- rtcDs3231: →getDateTime - button: → pushedLimitButton
Traces	- NOTCHARGINGBATTERY_FUNCTION_MASK - NOTCHARGINGBATTERY_TRACE1 - NOTCHARGINGBATTERY_TRACE2 - NOTCHARGINGBATTERY_TRACE3 - NOTCHARGINGBATTERY_TRACE4 - NOTCHARGINGBATTERY_TRACE5 - NOTCHARGINGBATTERY_TRACE6 - NOTCHARGINGBATTERY_TRACE7 - NOTCHARGINGBATTERY_TRACE8 - NOTCHARGINGBATTERY_TRACE9

Method	regularMode
File	Wearable.ino
Line	465-519
Description	Checks constantly the date. If it has not change from the last measurement, the device collects data from the accelerometer each second and store it into the SD card once the second has changed. If the date has changed, it creates a new CSV file with the name of the new date. If the month is also different, it creates a new directory with the new month.
Inputs	NONE
Output	NONE
Dependent Methods	- collectData - storeData
Dependent Attributes	- rtcDs3231: →getDateTime - ESP32CurrDir - ESP32CurrCSVFile
Traces	- REGULARMODE_FUNCTION_MASK - REGULARMODE_TRACE1 - REGULARMODE _TRACE2 - REGULARMODE _TRACE3 - REGULARMODE _TRACE4 - REGULARMODE _TRACE5

Method	collectData
File	Wearable.ino
Line	524-574
Description	Takes each possible variable from the accelerometer and makes the necessary offset transformation, change of axis and matrix rotation. After this, it stores this information, in a String format, within the main buffer passed as argument.
Inputs	- Buff (char *): The buffer to store data in a CSV format to save after within the SD card.
Output	NONE
Dependent Methods	- offsetTransform - coordinateAxisChange
Dependent Attributes	- rtcDs3231: →getDateTime - mpu6050: →getMeasurements
Traces	- COLLECTDATA_FUNCTION_MASK - COLLECTDATA _TRACE1 - COLLECTDATA _TRACE2 - COLLECTDATA _TRACE3

Method	storeData
File	Wearable.ino
Line	579-593
Description	Stores all data within the main buffer, into the file within the Micro-SD.
Inputs	- Buff (char *): The buffer to store data in a CSV format to save after within the SD card.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	- ESP32CurrCSVFile - SDcard: → appendFile
Traces	- STOREDATA_FUNCTION_MASK - STOREDATA_TRACE1

Method	offsetTransform
File	Wearable.ino
Line	598-622
Description	Subtract the data collected by the accelerometer, the offset measurement taken by the first calibration.
Inputs	- Ax, Ay, Az, Gx, Gy, Gz (float *): references to the measurements taken by the accelerometer and transformed into a float measurement, to subtract the offset measurement.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	- offsetAcc: → x → y → z - offsetGyr: → x → y → z
Traces	- OFFSETTRANSFORM_FUNCTION_MASK - OFFSETTRANSFORM_TRACE1 - OFFSETTRANSFORM_TRACE2
Method	coordinateAxisChange
File	Wearable.ino
Line	628-687

Description	Change the coordinate axis based on the position of the accelerometer within the PCB. By doing this, we can equal the axis of the Wearable with the Axis of the people.
Inputs	- Ax, Ay, Az, Gx, Gy, Gz (float *): references to the measurements taken by the accelerometer and transformed into a float measurement, to subtract the offset measurement.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	- offsetAcc: ➔ x ➔ y ➔ z - offsetGyr: ➔ x ➔ y ➔ z
Traces	- COORDINATEAXISCHANGE_FUNCTION_MASK - COORDINATEAXISCHANGE _TRACE1 - COORDINATEAXISCHANGE _TRACE2

Method	settingMode
File	Wearable.ino
Line	693-811
Description	Describes the normal behaviour of the device when is in Bluetooth (setting) mode. It sets the timeActive to 0. When another Bluetooth device is connected to the Wearable, it sends the data stored in the EEPROM to that device. After that it waits for a valid message from the device. If a message arrives, it evaluates if it is valid. If it is not, the Wearable discards the message. If it is so, it stores the new information to the EEPROM and resets the time to 0 again. If the Bluetooth has been active more than the time established, or the button is pressed again, it deactivates the Bluetooth and returns to the regular or normal mode.
Inputs	NONE
Output	NONE
Dependent Methods	- recoverEEPROMMemoryData - castAttributestoJSON - castAttributestoJSON - receiveBthAsyncMessage - castJSONtoAttributes - storeDataIntoEEPROMMemory

Dependent Attributes	- rgbled: ➔ setColor ➔ blinkingLed - eeprom: ➔ clearEEPROM - timeActive
Traces	- SETTINGMODE_FUNCTION_MASK - SETTINGMODE _TRACE1 - SETTINGMODE _TRACE2 - SETTINGMODE _TRACE3 - SETTINGMODE _TRACE4 - SETTINGMODE _TRACE5 - SETTINGMODE _TRACE6 - SETTINGMODE _TRACE7 - SETTINGMODE _TRACE8 - SETTINGMODE _TRACE9 - SETTINGMODE _TRACE10

Method	settingMode
File	Wearable.ino
Line	816-849
Description	Describes the normal behaviour of the device when is in Wi-Fi (transferring) mode. It first evaluates the status of the Broker. If the broker is available, it loops each file of the SD-Card and begin transferring the data from the SD to the broker, where the route of the server is located. When all the files have been transferred, the device returns to the regular or normal mode.
Inputs	NONE
Output	NONE
Dependent Methods	- WiFiConnect - WiFiDisconnect - sendMessageToBroker - receiveMessageToBroker
Dependent Attributes	NONE
Traces	- TRANSFERRINGMODE_FUNCTION_MASK - TRANSFERRINGMODE_TRACE1 - TRANSFERRINGMODE_TRACE2 - TRANSFERRINGMODE_TRACE3 - TRANSFERRINGMODE_TRACE4 - TRANSFERRINGMODE_TRACE5 - TRANSFERRINGMODE_TRACE6 - TRANSFERRINGMODE_TRACE7 - TRANSFERRINGMODE_TRACE8

Method	delayingAction

File	Wearable.ino
Line	867-885
Description	Sleeps or delay the device, to achieve a regular baud rate of gathering data. To manage this, it subtracts the normal baud rate, with the time in µs, employed to the execution from the last cycle.
Inputs	- t_us (int): The time intended to delay, including all internal process to sleep and save energy.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	- timeComp - timeBegComp - wakeup_reason
Traces	NONE

Method	checkHardwareConnections
File	Wearable.ino
Line	910-987
Description	Checks the connections with the accelerometer, the RTC and Micro-SD Card Module. If there is some failure in one of them, it returns an error value.
Inputs	NONE
Output	(int): - HARDWARE_OK: The connections with the modules has been with success. - HARDWARE_SD_CARD_ERROR_CONNECTION: The SD Card Module has not been detected. - HARDWARE_RTC_ERROR_CONNECTION: The RTC module has not been detected. - HARDWARE_IMU_ERROR_CONNECTION: The accelerometer has not been detected. - HARDWARE_UNKNOWN_ERROR: There has been an unknown error.
Dependent Methods	NONE
Dependent Attributes	- mpu6050: ➔ checkConnection ➔ initialize - rtcDs3231: ➔ checkConnection - SDcard: ➔ initialize
Traces	- WEARABLE_LOG_MASK

	- CHECKHARDWARECONNECTION_FUNCTION_MASK

Method	recoverEEPROMMemoryData
File	Wearable.ino
Line	1011-1229
Description	Recovers all relevant information from the EEPROM memory. If there is some of the relevant data, the function returns a value indicating which value is missing.
Inputs	NONE
Output	(int): - WEAR_EEPROM_DATA_OK: All relevant data has been recovered with success. - WEAR_EEPROM_DATA_MISSING_NET_SSID: The network SSID is missing. - WEAR_EEPROM_DATA_MISSING_NET_PASS: The network password is missing. - WEAR_EEPROM_DATA_MISSING_BROKER_IP: The IP of the broker is missing. - WEAR_EEPROM_DATA_MISSING_PAT_USER: The name of the user is missing. - WEAR_EEPROM_DATA_MISSING_PAT_PASS: The password of the user is missing.
Dependent Methods	NONE
Dependent Attributes	- eeprom: ➔ getStrEeprom - ESP32SSID - ESP32Pass - ESP32BrokerIP - ESP32PatUser - ESP32PatPass
Traces	- WEARABLE_LOG_MASK - RECOVEREEPROMMEMORYDATA_FUNCTION_MASK

Method	storeDataIntoEEPROMMemory
File	Wearable.ino
Line	1011-1229
Description	Tries to store the local information regarding to the relevant data. If some of the data has been failed to store within the EEPROM memory.
Inputs	NONE
Output	(int): - STORE_DATA_EEPROM_OK: All the relevant data has been stored with success.

	- STORE_DATA_EEPROM_ERROR_TIME_ACTIVE: The time active has been failed to be stored. - STORE_DATA_EEPROM_ERROR_MQTT_PASSWORD: The patient password has been failed to be stored. - STORE_DATA_EEPROM_ERROR_MQTT_USER: The patient user has been failed to be stored. - STORE_DATA_EEPROM_ERROR_SERVER_IP: The broker IP has failed to be stored. - STORE_DATA_EEPROM_ERROR_NET_PASSWORD: The password of the network has been failed to be stored. - STORE_DATA_EEPROM_ERROR_NET_SSID: The SSID has been failed to be stored.
Dependent Methods	NONE
Dependent Attributes	- eeprom: ➔ setStrEeprom ➔ setIntEeprom - ESP32SSID - ESP32Pass - ESP32BrokerIP - ESP32PatUser - ESP32PatPass - timeActive
Traces	- WEARABLE_LOG_MASK - STOREDATAINTOEEPROMMEMORY_FUNCTION_MASK

Method	storeDataIntoEEPROMMemory
File	Wearable.ino
Line	1320-1368
Description	Makes a calibration making average first and standard deviation then, with several samples establish by the manufacturer (in this case 200). After this, these measurements are stored in the offset variables.
Inputs	NONE
Output	NONE
Dependent Methods	NONE
Dependent Attributes	- offsetAcc: ➔ x ➔ y ➔ z - offsetGyr: ➔ x ➔ y ➔ z
Traces	- AUTOCALIBRATION_FUNCTION_MASK - AUTOCALIBRATION_TRACE1 - AUTOCALIBRATION_TRACE2

	- AUTOCALIBRATION_TRACE3

Method	msgUpdateToJSON
File	Wearable.ino
Line	1390-1406
Description	Creates a JSON object based on the valid message for the broker for updating in a remote way.
Inputs	- msg (char *): The reference of the char to store the JSON casted to String.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- MSGUPDATETOJSON_FUNCTION_MASK - MSGUPDATETOJSON _TRACE1

Method	castJSONtoAttributes
File	Wearable.ino
Line	1431-1577
Description	Takes a JSON message and tries to convert each key into a relevant data. After this, it stores this relevant information into the EEPROM, and returns a digit based on the mask with each bit corresponding to each relevant data. If there is a failure in storing these data, it returns an error message.
Inputs	- msg (char *): The message intended to cast into JSON and the key values to store.
Output	(int): - dataCast: The mask corresponding to the variables that has been stored with success, until value 64. For instance, 45 value, which corresponds to 101101 mask, indicates than the keys 1, 3, 4 and 6 are valid, but not keys 2 and 5. - CASTJSONSTATUS_INCORRECT_TYPE_KEY: One of the keys are not correct. - CASTJSONSTATUS_MISSING_KEY: One indispensable key is missing: class, data, timeActive or timestamp. - CASTJSONSTATUS_SYNTAX_ERROR: The message passed as variable is not castable to JSON format.
Dependent Methods	NONE

Dependent Attributes	- Eeprom: ➔ setIntEeprom ➔ setStrEeprom
Traces	- CASTJSONTOATTRIBUTE_FUNCTION_MASK - CASTJSONTOATTRIBUTE _TRACE1 - CASTJSONTOATTRIBUTE _TRACE2 - CASTJSONTOATTRIBUTE _TRACE3 - CASTJSONTOATTRIBUTE _TRACE4 - CASTJSONTOATTRIBUTE _TRACE5 - CASTJSONTOATTRIBUTE _TRACE6 - CASTJSONTOATTRIBUTE _TRACE7 - CASTJSONTOATTRIBUTE _TRACE8 - CASTJSONTOATTRIBUTE _TRACE9 - CASTJSONTOATTRIBUTE _TRACE10 - CASTJSONTOATTRIBUTE _TRACE11 - CASTJSONTOATTRIBUTE _TRACE12 - CASTJSONTOATTRIBUTE _TRACE13

Method	castAttributestoJSON
File	Wearable.ino
Line	1582-1632
Description	Takes the relevant information from the Wearable (SSID, password, IP, User, password, timeActive), and converts them into a String that is castable to JSON, int order to send them by Wi-Fi or Bluetooth.
Inputs	- Msgsize (int): The length of the message in order not to exceed a maximum. - msg (char *): The message intended to store relevant data into JSON format. - Inval (int): Some of the relevant data within the EEPROM is not valid or missing.
Output	(int): - CASTATTRTOJSON_OK: The message has been parsed with success. - CASTATTRTOJSON_ERROR_DATA_INCOMP: Some of the relevant data from the EEPROM are incomplete. - CASTATTRTOJSON_ERROR_INVAL_LENGTH: The length is invalid (minor than 0 or bigger than the maximum).
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- CASTATTRIBUTESTOJSON_FUNCTION_MASK - CASTATTRIBUTESTOJSON_TRACE1 - CASTATTRIBUTESTOJSON_TRACE2 - CASTATTRIBUTESTOJSON_TRACE3

Method	castJSONtoDatetime
File	Wearable.ino
Line	1638-1701
Description	Converts a message castable into JSON format, into the time and time zone. If there is a failure in casting the variables before, it returns an error value.
Inputs	- msg (const char *): The message intended, in JSON format, to extract time and time zone.
Output	(int): - CASTJSONDATETIME_STATUS_OK: Success in extracting UNIX timestamp and time zone from the JSON message. - CASTJSONDATETIME_STATUS_INCORRECT_TYPE_KEY : The type of data from the timestamp or the time zone is incorrect. - CASTJSONDATETIME_STATUS_MISSING_KEY: One of the keys before is missing. - CASTJSONDATETIME_STATUS_SYNTAX_ERROR: The message passed as argument is not a JSON valid format.
Dependent Methods	NONE
Dependent Attributes	- rtcDs3231: ➔ setDateTime ➔ castTimeZoneToInt
Traces	- CASTJSONTODATETIME_FUNCTION_MASK - CASTJSONTODATETIME _TRACE1 - CASTJSONTODATETIME _TRACE2 - CASTJSONTODATETIME _TRACE3 - CASTJSONTODATETIME _TRACE4 - CASTJSONTODATETIME _TRACE5 - CASTJSONTODATETIME _TRACE6 - CASTJSONTODATETIME _TRACE7

Method	checkOTAUpdate (not implemented yet)
File	Wearable.ino
Line	1735-1801
Description	Checks whether there is an available upgrade version within the broker or not, by asking directly to the broker with a message.
Inputs	NONE
Output	(int):

	- CHECK_OTA_UPDATE_AVBLE_UPDATE_BIN: There is a new upgrade version available in the broker. - CHECK_OTA_UPDATE_NO_AVBLE_UPDATE_BIN: There is no available upgraded version in the broker. - CHECK_OTA_UPDATE_ERROR_MESSAGE: The message returned from the broker is not valid. - CHECK_OTA_UPDATE_NO_MESSAGE: There has been no message from the broker after a timeout period. - CHECK_OTA_UPDATE_BROKER_DISCONNECTED: The broker is unreachable.
Dependent Methods	- WiFiConnect - analyzeMsgUpdateFromBroker
Dependent Attributes	- ESP32BinUpdate - SizeUpdate
Traces	- CHECKOTAUPDATE_FUNCTION_MASK - CHECKOTAUPDATE _TRACE1 - CHECKOTAUPDATE _TRACE2 - CHECKOTAUPDATE _TRACE3 - CHECKOTAUPDATE _TRACE4 - CHECKOTAUPDATE _TRACE5 - CHECKOTAUPDATE _TRACE6 - CHECKOTAUPDATE _TRACE7

Method	recvUpdateFromBroker
File	Wearable.ino
Line	1812-2034
Description	Tries to recover an upgraded version of the firmware for the Wearable for flashing internally.
Inputs	NONE
Output	(int): - WEAR_RECEIVE_UPD_FROM_BROKER_OK: The file has been received with success. - WEAR_RECEIVE_UPD_FROM_BROKER_ERROR_INV_SIZE: The real size of the file is incorrect based on the expected size. - WEAR_RECEIVE_UPD_FROM_BROKER_ERROR_INV_PART_SIZE: The size of one partition is incorrect. - WEAR_RECEIVE_UPD_FROM_BROKER_ERROR_INV_N_PART: The file has been fractioned and damage. - WEAR_RECEIVE_UPD_FROM_BROKER_ERROR_CARD_DISMOUNT: The micro-SD card has been unattached. - WEAR_RECEIVE_UPD_FROM_BROKER_ERROR_BROKER_DISCONN: The Wearable has lost the connection with the broker.

Dependent Methods	- sendMessageToBroker - recvMessageFromBroker - analyzeMsgUpdPartitionFromBroker
Dependent Attributes	- SDcard: ➔ Initialize ➔ createDir ➔ listDir ➔ deleteDir ➔ getFileByIndx ➔ checkFile ➔ deleteFile - ESP32BinUpdate
Traces	- RECVUPDATEFROMBROKER_FUNCTION_MASK - RECVUPDATEFROMBROKER _TRACE1 - RECVUPDATEFROMBROKER _TRACE2 - RECVUPDATEFROMBROKER _TRACE3 - RECVUPDATEFROMBROKER _TRACE4 - RECVUPDATEFROMBROKER _TRACE5 - RECVUPDATEFROMBROKER _TRACE6 - RECVUPDATEFROMBROKER _TRACE7 - RECVUPDATEFROMBROKER _TRACE8 - RECVUPDATEFROMBROKER _TRACE9 - RECVUPDATEFROMBROKER _TRACE10 - RECVUPDATEFROMBROKER _TRACE11 - RECVUPDATEFROMBROKER _TRACE12 - RECVUPDATEFROMBROKER _TRACE13 - RECVUPDATEFROMBROKER _TRACE14 - RECVUPDATEFROMBROKER _TRACE15 - RECVUPDATEFROMBROKER _TRACE16

Method	WiFiConnect
File	Wearable.ino
Line	2073-2258
Description	Tries to connect, by Wi-Fi, with the broker. First connects with the home network, then tries to connect with a ping command to the broker IP, then tries to enter with the MQTT and FTP ports. If some of these steps fails, it returns an error value.
Inputs	NONE
Output	(int): - WEAR_WIFI_CONNECT_OK: The connection with the broker has been established with success. - WEAR_WIFI_CONNECT_ERR_FTP_PORT: The FTP port is unreachable. - WEAR_WIFI_CONNECT_ERR_MQTT_PORT: The MQTT port is unreachable. - WEAR_WIFI_CONNECT_ERR_BROKER_IP: The broker IP has not been found.

	- WEAR_WIFI_CONNECT_ERR_UNAV_NETWORK: The network password stored in the EEPROM is incorrect. The connection with the net has been refused. - WEAR_WIFI_CONNECT_ERR_INVAL_NETWORK: The network
Dependent Methods	NONE
Dependent Attributes	- ESP32SSID - ESP32Pass - nWiFiConn - ESP32BrokerIP
Traces	- WIFICONNECT_FUNCTION_MASK - WIFICONNECT _TRACE1 - WIFICONNECT _TRACE2 - WIFICONNECT _TRACE3 - WIFICONNECT _TRACE4 - WIFICONNECT _TRACE5 - WIFICONNECT _TRACE6 - WIFICONNECT _TRACE7 - WIFICONNECT _TRACE8 - WIFICONNECT _TRACE9 - WIFICONNECT _TRACE10 - WIFICONNECT _TRACE11 - WIFICONNECT _TRACE12 - WIFICONNECT _TRACE13 - WIFICONNECT _TRACE14

Method	WiFiDisconnect
File	Wearable.ino
Line	2264-2305
Description	Closes MQTT and FTP ports and disables Wi-Fi.
Inputs	NONE
Output	(int): 0
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- WIFIDISCONNECT_FUNCTION_MASK - WIFIDISCONNECT _TRACE1 - WIFIDISCONNECT _TRACE2 - WIFIDISCONNECT _TRACE3 - WIFIDISCONNECT _TRACE4
Method	WiFiCallback
File	Wearable.ino

Line	2348-2375
Description	Callback activated in a background after a Wi-Fi event.
Inputs	- Topic (char *): The kind of event that has made the Wi-Fi callback trigger. - Message (byte *): The messaged array arrived in byte format. - Length (unsigned int): The length of the message arrived.
Output	NONE
Dependent Methods	NONE
Dependent Attributes	MQTTResponse (static) MQTTStatus (static)
Traces	- WIFICALLBACK_FUNCTION_MASK - WIFICALLBACK_TRACE1

Method	CastJSONIPVertoString
File	Wearable.ino
Line	2381-2439
Description	Takes a JSON message passed as argument and tries to parse the IP and the version of the new upgraded version of firmware. If some of the keys are missing, it returns an error value.
Inputs	- JSONstr (char *): The message intended to parse into JSON. - JSONstrLength (char *): The length of the message.
Output	(int): - JSON_CAST_IP_VER_OK: The IP and the version is casted successfully. - JSON_CAST_IP_VER_STR_LENGTH_ERROR: The message is bigger than the maximum established by the manufacturer. - JSON_CAST_IP_VER_STR_OVERFLOW_ERROR: There has been an overflow error.
Dependent Methods	NONE
Dependent Attributes	- SDcard: ➔ checkFile
Traces	- CASTJSONIPVERTOSTRING_FUNCTION_MASK - CASTJSONIPVERTOSTRING _TRACE1 - CASTJSONIPVERTOSTRING _TRACE2 - CASTJSONIPVERTOSTRING _TRACE3 - CASTJSONIPVERTOSTRING _TRACE4 - CASTJSONIPVERTOSTRING _TRACE5

Method	sendMessageToBroker
File	Wearable.ino
Line	2471-2504
Description	Sends a message passed as argument to the broker, within the MQTT topic also passed as argument. If there is an error during the transmission, the function returns an error value.
Inputs	- msg (char *): The message intended to send to the broker. - Topic const (char *): The topic intended to send.
Output	(int): - SEND_MSG_WIFI_MQTT_OK: The message has been sent with success. - SEND_MSG_WIFI_MQTT_UNAVBLE_PORT: The MQTT port is not available. - SEND_MSG_WIFI_MQTT_BUFFER_OVERFLOW: The buffer has overpassed the maximum allowed.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- SENDMESSAGETOBROKER_FUNCTION_MASK - SENDMESSAGETOBROKER _TRACE1 - SENDMESSAGETOBROKER _TRACE2 - SENDMESSAGETOBROKER _TRACE3

Method	recvMessageFromBroker
File	Wearable.ino
Line	2539-2602
Description	Waits for a message in a threshold passed as argument. If the message is not sent within the threshold, it returns an error value. If the threshold is negative, it waits indefinitely until a message arrives.
Inputs	- msg (char *): The reference intended to store the message that arrives. - Topic (const char *): The topic from where the message expected. - timeThres_ms (int): The threshold within the message is expected.
Output	(int): - RECV_MSG_WIFI_MQTT_OK: The message is received with success. - RECV_MSG_WIFI_MQTT_ERROR_TIMEOUT_REACHED: The message is not received within threshold.

	- RECV_MSG_WIFI_MQTT_ERROR_UNAVBLE_PORT: The MQTT port is not available. - RECV_MSG_WIFI_MQTT_ERROR_UNAVBLE_NETWORK : The network is unavailable. - RECV_MSG_WIFI_MQTT_ERROR_UNINIT_ATTR: One or more attributes needed are uninitialized.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- RECVMESSAGEFROMBROKER_FUNCTION_MASK - RECVMESSAGEFROMBROKER_TRACE1 - RECVMESSAGEFROMBROKER_TRACE2 - RECVMESSAGEFROMBROKER_TRACE3 - RECVMESSAGEFROMBROKER_TRACE4

Method	analyzeMsgUpdFromBroker (not implemented yet)
File	Wearable.ino
Line	2638-2785
Description	Analyze the message arrived whether it is referent to an updated version of the firmware or not. If the message is in JSON format, and the JSON class key is an update, it returns a reference of the updated bin file and its size. Otherwise, it returns an error value.
Inputs	- msg (const char *): The JSON intended to evaluate in a String format. - nameUpdate (const char *): The name of the expected bin file. - sizeUpdate (int): The reference to the length or the size of the updated bin file.
Output	(int): - ANALYZE_MSG_UPDATE_BROKER_UPDATE_AVAILABLE: The message is correct, and an update is available in the broker. - ANALYZE_MSG_UPDATE_BROKER_NO_UPDATE_AVAILABLE: The message is correct, and there is no update available in the broker. - ANALYZE_MSG_UPDATE_BROKER_ERROR_INVAL_VALUE: The message is correct but some keys or some values are not valid. - ANALYZE_MSG_UPDATE_BROKER_ERROR_MISSING_KEY: The message is correct, but some keys are missing. - ANALYZE_MSG_UPDATE_BROKER_ERROR_SYNTAX: The message is not correct because there is not a JSON format. - ANALYZE_MSG_UPDATE_BROKER_ERROR_POSSIBLE_OVERFLOW: The message has overpassed the limits of the length.

Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- MSGUPDATEFROMJSON_FUNCTION_MASK - MSGUPDATEFROMJSON_TRACE1 - MSGUPDATEFROMJSON_TRACE2 - MSGUPDATEFROMJSON_TRACE3 - MSGUPDATEFROMJSON_TRACE4 - MSGUPDATEFROMJSON_TRACE5 - MSGUPDATEFROMJSON_TRACE6 - MSGUPDATEFROMJSON_TRACE7 - MSGUPDATEFROMJSON_TRACE8 - MSGUPDATEFROMJSON_TRACE9 - MSGUPDATEFROMJSON_TRACE10 - MSGUPDATEFROMJSON_TRACE11 - MSGUPDATEFROMJSON_TRACE12

Method	analyzeMsgUpdPartitionFromBroker (not implemented yet)
File	Wearable.ino
Line	2820-2964
Description	Analyze the message passed as argument. If the message is in JSON format and the JSON is formed in the specific format, return as a reference whether the partition of the updated bin file is available or not. If it is available, it also returns the size of the partition.
Inputs	- msg (const char *): The JSON message intended to assess in String format. - sizePartition (int *): The reference to the length or the size of the updated partition file.
Output	(int): - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_PART_AVAILABLE: The message is valid and there is a partition available in the broker. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_NO_PART_AVAILABLE: The message is valid and there is not partition available in the broker. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_INVAL_VALUE: The message is a valid JSON but some keys or some values are not valid. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_MISSING_KEY: The message is a valid JSON, but some keys are missing. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_SYNTAX: The message is not a valid JSON.

	- WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_POSSIBLE_OVERFLOW: The length of the message has overpassed the maximum allowed.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- MSGUPDATEPARTITIONFROMJSON_FUNCTION_MASK - MSGUPDATEPARTITIONFROMJSON _TRACE1 - MSGUPDATEPARTITIONFROMJSON _TRACE2 - MSGUPDATEPARTITIONFROMJSON _TRACE3 - MSGUPDATEPARTITIONFROMJSON _TRACE4 - MSGUPDATEPARTITIONFROMJSON _TRACE5 - MSGUPDATEPARTITIONFROMJSON _TRACE6 - MSGUPDATEPARTITIONFROMJSON _TRACE7 - MSGUPDATEPARTITIONFROMJSON _TRACE8 - MSGUPDATEPARTITIONFROMJSON _TRACE9 - MSGUPDATEPARTITIONFROMJSON _TRACE10 - MSGUPDATEPARTITIONFROMJSON _TRACE11 - MSGUPDATEPARTITIONFROMJSON _TRACE12

Method	analyzeMsgUpdPartitionFromBroker (not implemented yet)
File	Wearable.ino
Line	2820-2964
Description	Analyze the message passed as argument. If the message is in JSON format and the JSON is formed in the specific format, return as a reference whether the partition of the updated bin file is available or not. If it is available, it also returns the size of the partition.
Inputs	- msg (const char *): The JSON message intended to assess in String format. - sizePartition (int *): The reference to the length or the size of the updated partition file.
Output	(int): - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_PART_AVAILABLE: The message is valid and there is a partition available in the broker. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_NO_PART_AVAILABLE: The message is valid and there is not partition available in the broker. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_INVAL_VALUE: The message is a valid JSON but some keys or some values are not valid.

	- WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_MISSING_KEY: The message is a valid JSON, but some keys are missing. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_SYNTAX: The message is not a valid JSON. - WEAR_ANALYZE_MSG_UPDATE_PART_BROKER_ERROR_POSSIBLE_OVERFLOW: The length of the message has overpassed the maximum allowed.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- MSGUPDATEPARTITIONFROMJSON_FUNCTION_MASK - MSGUPDATEPARTITIONFROMJSON _TRACE1 - MSGUPDATEPARTITIONFROMJSON _TRACE2 - MSGUPDATEPARTITIONFROMJSON _TRACE3 - MSGUPDATEPARTITIONFROMJSON _TRACE4 - MSGUPDATEPARTITIONFROMJSON _TRACE5 - MSGUPDATEPARTITIONFROMJSON _TRACE6 - MSGUPDATEPARTITIONFROMJSON _TRACE7 - MSGUPDATEPARTITIONFROMJSON _TRACE8 - MSGUPDATEPARTITIONFROMJSON _TRACE9 - MSGUPDATEPARTITIONFROMJSON _TRACE10 - MSGUPDATEPARTITIONFROMJSON _TRACE11 - MSGUPDATEPARTITIONFROMJSON _TRACE12

Method	receiveBluetoothMsg
File	Wearable.ino
Line	3042-3080
Description	Waits for a message from the Bluetooth Module. If a message arrives, it returns a valid value if the buffer has not overpassed the maximum length, otherwise it returns an error message
Inputs	- msgSize (int): the size of the message passed as argument. - msg (const char *): The JSON message intended to assess in String format.
Output	(int): - RECEIVE_MSG_BLUETOOTH_OK: The message is correctly saved. - RECEIVE_MSG_BLUETOOTH_ERROR_BUFFER_OVERFLOW: The buffer is empty before the length is cover.
Dependent Methods	NONE

Dependent Attributes	NONE
Traces	- RECEIVEBLUETOOTHMSG_FUNCTION_MASK - RECEIVEBLUETOOTHMSG _TRACE1 - RECEIVEBLUETOOTHMSG _TRACE2

Method	receiveBluetoothAsyncMsg
File	Wearable.ino
Line	3085-3140
Description	Checks whether there is a message available in the Bluetooth module or not. If there is, it returns a value of message available. Otherwise, it returns a value that indicates there is no message available.
Inputs	- msgSize (int): the size of the message passed as argument. - msg (const char *): The JSON message intended to assess in String format.
Output	(int): - RECEIVE_MSG_BLUETOOTH_ASYNC_AVAILABLE_MESSAGE: There is a message available in the Bluetooth module. - RECEIVE_MSG_BLUETOOTH_ASYNC_NO_AVAILABLE_MESSAGE: There is no message available in the Bluetooth module. - RECEIVE_MSG_BLUETOOTH_ASYNC_ERROR_BUFF_OVERFLOW: The message available in the Bluetooth module is bigger than the maximum. In this case the message is discarded.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- RECEIVEBLUETOOTHASYNCMSG_FUNCTION_MASK - RECEIVEBLUETOOTHASYNCMSG _TRACE1 - RECEIVEBLUETOOTHASYNCMSG _TRACE2 - RECEIVEBLUETOOTHASYNCMSG _TRACE3 - RECEIVEBLUETOOTHASYNCMSG _TRACE4

Method	receiveBluetoothAsyncMsg
File	Wearable.ino

Line	3085-3140
Description	Checks whether there is a message available in the Bluetooth module or not. If there is, it returns a value of message available. Otherwise, it returns a value that indicates there is no message available.
Inputs	- msgSize (int): the size of the message passed as argument. - msg (const char *): The JSON message intended to assess in String format.
Output	(int): - RECEIVE_MSG_BLUETOOTH_ASYNC_AVAILABLE_MESSAGE: There is a message available in the Bluetooth module. - RECEIVE_MSG_BLUETOOTH_ASYNC_NO_AVAILABLE_MESSAGE: There is no message available in the Bluetooth module. - RECEIVE_MSG_BLUETOOTH_ASYNC_ERROR_BUFF_OVERFLOW: The message available in the Bluetooth module is bigger than the maximum. In this case the message is discarded.
Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- RECEIVEBLUETOOTHASYNCMSG_FUNCTION_MASK - RECEIVEBLUETOOTHASYNCMSG _TRACE1 - RECEIVEBLUETOOTHASYNCMSG _TRACE2 - RECEIVEBLUETOOTHASYNCMSG _TRACE3 - RECEIVEBLUETOOTHASYNCMSG _TRACE4

Method	sendBluetoothMsg
File	Wearable.ino
Line	3163-3183
Description	Tries to send a message by Bluetooth.
Inputs	- msgSize (int): the size of the message passed as argument. - msg (const char *): The JSON message intended to assess in String format.
Output	(int): - SENDBLUETOOTHMSG_OK: The message has been sent with success. - SENDBLUETOOTHMSG_BUFF_OVERFLOW: The message is bigger than the maximum length or size allowed

Dependent Methods	NONE
Dependent Attributes	NONE
Traces	- SENDBLUETOOTHMSG_FUNCTION_MASK - SENDBLUETOOTHMSG_TRACE1

6 Testing

6.1 Unit Testing

For the unitary testing, the procedure is using each function and test it trying to cover each possible output and maximising the coverage. Each function is added to a script and executed individually, pointing the difference testcases, methods, attributes, and arguments at each process. The theoretical outcome will be compared with the real outcome and, if both outcomes match and the coverage is covered 100%, the function is passed the test. The statements are the separation between one command to the processor (excluding traces and comments). Here are two examples of unitary testing for two simple functions. There are more examples in the Annex below. If the readers wants a complete detail of the test, this link leads to the remote repository [38]:

- ***BUTTON:***
  - ➔ ***edgeDetection***

Method	**edgeDetection**
script	Button_edgeDetection.ino
Testbench	Button_edgeDetection.xlsx
Testcases	4
Statements	10

Testcase	**1**
Definition	No actions
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 0
Expected Outcome	NO_EDGES (0)
Real Outcome	NO_EDGES (0)
Expected Traces	1, 2, 3, 9, 10
Real Traces	1, 2, 3, 9, 10

Testcase	**2**
Definition	No actions
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 1
Expected Outcome	NO_EDGES (0)
Real Outcome	NO_EDGES (0)
Expected Traces	1, 2, 3, 9, 10
Real Traces	1, 2, 3, 9, 10

Testcase	**3**
Definition	The button is pressed
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 0 (1 after pressed)
Expected Outcome	RISING_EDGE (1)
Real Outcome	RISING_EDGE (1)
Expected Traces	1, 2, 3, 4, 5, 6
Real Traces	1, 2, 3, 4, 5, 6

Testcase	**4**
Definition	The button is released
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 1 (0 after pressed)
Expected Outcome	FALLING_EDGE (-1)
Real Outcome	FALLING_EDGE (-1)
Expected Traces	1, 2, 3, 4, 5, 7, 8
Real Traces	1, 2, 3, 4, 5, 7, 8

Coverage	**10**
(%)	100

→ ***pushedLimitButton***

Method	**pushedLimitButton**
script	Button_pushedLimitButton.ino
Testbench	Button_ pushedLimitButton.xlsx
Testcases	4
Statements	16

Testcase	**1**
Definition	Invalid negative integer as argument
Arguments	timeLimit_ms → -1
Attributes	digitalRead(GPIO_PUSH_BUTTON) → *Irrelevant*
Expected Outcome	INVALID_INTEGER (-2)
Real Outcome	INVALID_INTEGER (-2)
Expected Traces	1, 2
Real Traces	1, 2

Testcase	**2**
Definition	Button not pressed when function is called
Arguments	timeLimit_ms → *Irrelevant*
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 0
Expected Outcome	PREVIOUS_STATE_LOW (-1)
Real Outcome	PREVIOUS_STATE_LOW (-1)
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Testcase	**3**
Definition	Time limit is 5 seconds and button is released after 3 seconds pressed.
Arguments	timeLimit_ms → 5000
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 1
Expected Outcome	NOT_EXCEEDED_LIMIT (0)
Real Outcome	NOT_EXCEEDED_LIMIT (0)
Expected Traces	1, 3, 5, 6, 7, loop (8, 9, 11), 12, 13, 16
Real Traces	1, 3, 5, 6, 7, loop (8, 9, 11), 12, 13, 16

Testcase	**4**
Definition	Time limit is 5 seconds and button is pressed more than 5 seconds.
Arguments	timeLimit_ms → 5000
Attributes	digitalRead(GPIO_PUSH_BUTTON) → 1
Expected Outcome	EXCEEDED_LIMIT (1)
Real Outcome	EXCEEDED_LIMIT (1)
Expected Traces	1, 3, 5, 6, 7, loop (8, 9, 11), 10, 12, 14, 15, 16
Real Traces	1, 3, 5, 6, 7, loop (8, 9, 11), 10, 12, 14, 15, 16

Coverage	**10**
(%)	100

Next, there is a summary of the total of Testcases per each method, most of them remains in the remote repository and some of them are summarize in the Annex:

- ***BATTERY:***

Method	Testcases	Coverage (%)
Initialize	1	100
batteryVoltage	2	100

- ***DS3231:***

Method	Testcases	Coverage (%)
checkConnections	1	100
getDatetime	8	100
getDatetimeStr	1	100
setDatetime	2	100
castTimeZonetoInt	40	100

- ***RGBLED:***

Method	Testcases	Coverage (%)
begins	1	100
setColor	8	100
blinkingLed	1	100

- ***MICROSDCARD:***

Method	Testcases	Coverage (%)
Initialize	2	100
createDir	2	100
listDir	4	100
getFileByIndx	4	100
deleteDir	3	100
readFile	4	100
appendFile	4	100
appendFile (overloading)	4	100
renameFile	3	100
checkFile	2	100
deleteFile	2	100

- ***ESP32EEPROM:***

Method	Testcases	Coverage (%)
Initialize	2	100
setIntEEPROM	6	100
getIntEEPROM	6	100
setStrEEPROM	5	100
getStrEEPROM	7	100
isValidASCII	3	100

- ***AGMPU6050:***

Method	Testcases	Coverage (%)
Initialize	1	100
checkConnections	2	100
getMeasurements	7	100

- ***WEARABLE:***

Method	Testcases	Coverage (%)
isValidASCII	2	100
setup	12	100
loop	2	100
chargingBattery	1	100
notChargingBattery	6	100
regularMode	4	100
collectData	2	100
storeData	1	100
offsetTransform	1	100
coordinateAxisChange	9 (1 per each 9 possibilities of manufacturing)	100
deleteFile	2	100
settingMode	10	100
transferringMode	4	100

delayingAction	2	100
checkHardwareConnections	5	100
recoverEEPROMMemoryData	6	100
storeDataIntoEEPROMMemory	8	100
autoCalibration	1	100
msgUpdateToJSON	1	100
castJSONToAttributes	20	100
castAttributesToJSON	8	100
castJSONtoDatetime	6	100
checkOTAUpdate	5	100
recvUpdateFromBrk	18	100
WiFiConnect	10	100
WiFiDisconnect	3	100
wifiCallback	4	100
castJSONIPVerToString	6	100
sendMessageToBroker	4	100
recvMessageFromBroker	3	100
analyzeMsgUpdFromBroker	16	100
analyzeMsgUpdPartitionFromBroker	14	100
receiveBluetoothMsg	2	100
receiveBluetoothAsyncMsg	3	100
sendBluetoothMsg	2	100

6.2 Functional Testing

This part is essential and crucial to check the requirements, especially if the Functional Requirements are fulfilled. Because normally this is a different task that performs other team members different from the developer (normally a tester), the functioning of this part is giving a trail to the tester, so they can follow a bunch of steps to check whether the device behaves as it should. The result will be remarked in bold, and it must be the same as the theoretical.

- ***FR1: This Functional Requirement regards about the hardware connections, the configuration of the storing modules (the EEPROM and the SD card) and calibration of the device.***

(FR1.1, FR1.2, FR1.3 requirements fulfilled)

Step1	Disconnect the accelerometer from the PCB

Step2	Power on the device
Expected Result	(FR1.4) The LED remains red because it does not detect the accelerometer
Check	OK
Step3	Power off the device
Step4	Connect the accelerometer to the PCB
Step5	Disconnect the RTC
Step6	Power on the device
Expected Result	(FR1.5) The LED remains red because it does not detect the RTC
Check	OK
Step7	Power off the device
Step8	Connect the RTC to the PCB
Step9	Detach the Micro-SD Card from the board.
Step10	Power on the device
Expected Result	(FR1.6) The LED remains red because it does not detect the SD-card
Check	OK
Step11	Power off the device
Step12	Attach the SD to the PCB
Step13	Power on the device
Expected Result	(FR1.7, FR1.8, FR1.9) The LED turns green because it starts gathering measurements from the accelerometer
Check	OK

- ***FR2-FR3: These Functional Requirements regards about when the device is being charged and where it disconnects.***

Step1	Plug the device with the charger
Expected Result	(FR2.1) The LED starts blinking indicating that the device is being charged
Check	OK
Step2	Waits until the charge is completely loaded
Expected Result	(FR3.1, FR3.2, FR3.3, FR3.4, FR3.5) The LED stops blinking and turns into green.
Check	OK
Step3	Unplug the device
Expected Result	The LED switches off.
Check	OK

- ***FR4: This Functional Requirement regards about when the device has the button pressed, so we can evaluate whether it changes of status manually or not.***

Step1	Power on the device and waits until the LED is in green
Step2	Push the button less than 3 seconds
Expected Result	(FR4.1-FR4.2) The device must turn from Regular Mode (gathering and collecting data from the accelerometer) to Setting Mode (Bluetooth activated). The LED turns into blue
Check	OK
Step3	Push the button again and return to Regular Mode
Step4	One in the Regular Mode, push the button more than 3 seconds this time
Expected Result	(FR4.1-FR4.2) The device must turn from Regular Mode to Transferring Mode (Wi-Fi activated). The LED turns into indigo colour.
Check	OK
Step5	Waits until all the files have been transmitted and return to the Regular Mode

- ***FR5: This Functional Requirement regards about the Regular Mode, when there is no charger plugged to the device, and checks whether each second updates the CSV file, and if there is a change of file once the day is over.***

(FR5.2, FR5.3, FR5.4, FR5.5 requirements fulfilled)

Step1	Power on the device and waits until the LED is in green
Step2	Waits until there is a change of date
Expected Result	(FR5.1) The device creates a new CSV file within the SD Card, and all the collected data from now on is stored in this new file.
Check	OK
Step3	Waits until a different second, and check after this the SD-Card is written
Expected Result	(FR5.1-FR5.6) The device must write on the SD-Card each second
Check	OK

- ***FR6: This Functional Requirement regards about the Setting Mode (Bluetooth) and if the device behaves as it should when a smartphone or tablet is connected, and if the device stores the information within the EEPROM or discards it depending on whether is correct or not.***

Step1	Power on the device and waits until the LED is in green
Step2	Push button to swift from Regular to Setting Mode
Step3	Connect a smartphone or tablet via Bluetooth
Expected Result	(FR6.1) The LED blinks twice and the device sends the stored relevant information in JSON format.
Check	OK
Step4	Send a false information from the smartphone or tablet to the Wearable
Expected Result	(FR6.3-FR6.4) The device discards this information.
Check	OK
Step5	Send a correct information from the smartphone or tablet to the Wearable
Expected Result	(FR6.5-FR6.6) The device stores the relevant information into the EEPROM.
Check	OK
Step6	Waits until the timeout passes (in this case 1 minute)
Expected Result	(FR6.2) The device returns to the regular mode gathering data from the accelerometer.
Check	OK

- ***FR7: This Functional Requirement Regards about the Transferring Mode (Wi-Fi) and its behaviour when the time to transfer comes into it, ensuring that all files are sent correctly to the Broker.***

(FR7.1 requirements fulfilled)

Step1	Power on the device and waits until the LED is in green
Step2	Turn off the Router of the home Network
Step3	Push the button more than 3 seconds to swift from Regular Mode to Transferring Mode
Expected Result	(FR7.2) The LED first turns into an indigo colour, after this turns into a red colour and blinks once, because it does not find the home network.
Check	OK
Step4	Return to the Regular Mode, by waiting 10 seconds
Step5	Turn on the Router of the home Network
Step6	Turn off the broker (Raspberry Pi)
Step7	Push the button more than 3 seconds to swift from Regular Mode to Transferring Mode
Expected Result	(FR7.3) The LED first turns into an indigo colour, after this turns into a red colour and blinks twice, because it does not find the broker IP.

Check	OK
Step8	Return to the Regular Mode, by waiting 10 seconds
Step9	Turn on the Raspberry Pi
Step10	Disable MQTT and FTP ports from the Broker
Step11	Push the button more than 3 seconds to swift from Regular Mode to Transferring Mode
Expected Result	(FR7.4) The LED first turns into an indigo colour, after this turns into a red colour and blinks four times, because it does not find the port of the broker (1883).
Check	OK
Step12	Return to the Regular Mode, by waiting 10 seconds
Step13	Enable the port 1883 by booting the Mosquitto process
Step14	Push the button more than 3 seconds to swift from Regular Mode to Transferring Mode
Expected Result	(FR7.5, FR7.6, FR7.7) The LED first turns into an indigo colour, after this turns into a red colour, after this a yellow colour and finally again into an indigo colour to show that the device is transferring the files from the SD to the Raspberry Pi.
Check	OK
Step15	(FR7.8) After transmitting the files, the device enters in a sleep mode and LED turns off. To return to the regular mode, it has to push the button again.

7 Deployment

7.1 Retirement homes

The validation of the system, especially regarding the wearable function and usability and reliability, will be performed in several retirement homes, as well as in individual homes.

The professionals will ask the patients to perform several tasks, such as walk straight 2.4 and 6 meters twice in a row and perform some activity daily lifetime. In the meantime, several professionals will take notes about the activity, how much time they take to complete, and if the patients have some incapacities that difficult the activity.

To record the data with the possibility of watching directly, the Wearable is flashed in a special way, that will help us to plot in the screen with the Serial Plotter, a tool from the Arduino IDE. The marks shows when the patient start to do the activity, and when he finishes. Besides this, the file is also stored within the SD card.

Here is the following data from the Retirement Homes:

Place	**Address**	**Patients**
Centro de Día Jose Manuel Bringas [39]	Camino Viejo de Villaverde, 26, 28041, Madrid	215 219 249 302 388 477 661 821
Centro de Día Activa Edades [40]	Doctor Cornago, 1, 28223, Pozuelo de Alarcon	63 140 216 340 576 643
Hospital Getafe [41]	Carretera Madrid – Toledo, km. 12.5, 28905, Getafe	223 996

To monitor measurements for each user, we will ask them to turn the device on, and whenever the Arduino Serial Plotter is ready and able, we will ask them to start walking, carefully taking one mark when the exercise starts, and when the

exercise ends, to flag each time, to make it easier the researchers to analyze the signal.

Here is an example of a random plot track, remarking the flags, and the steps of the patients:

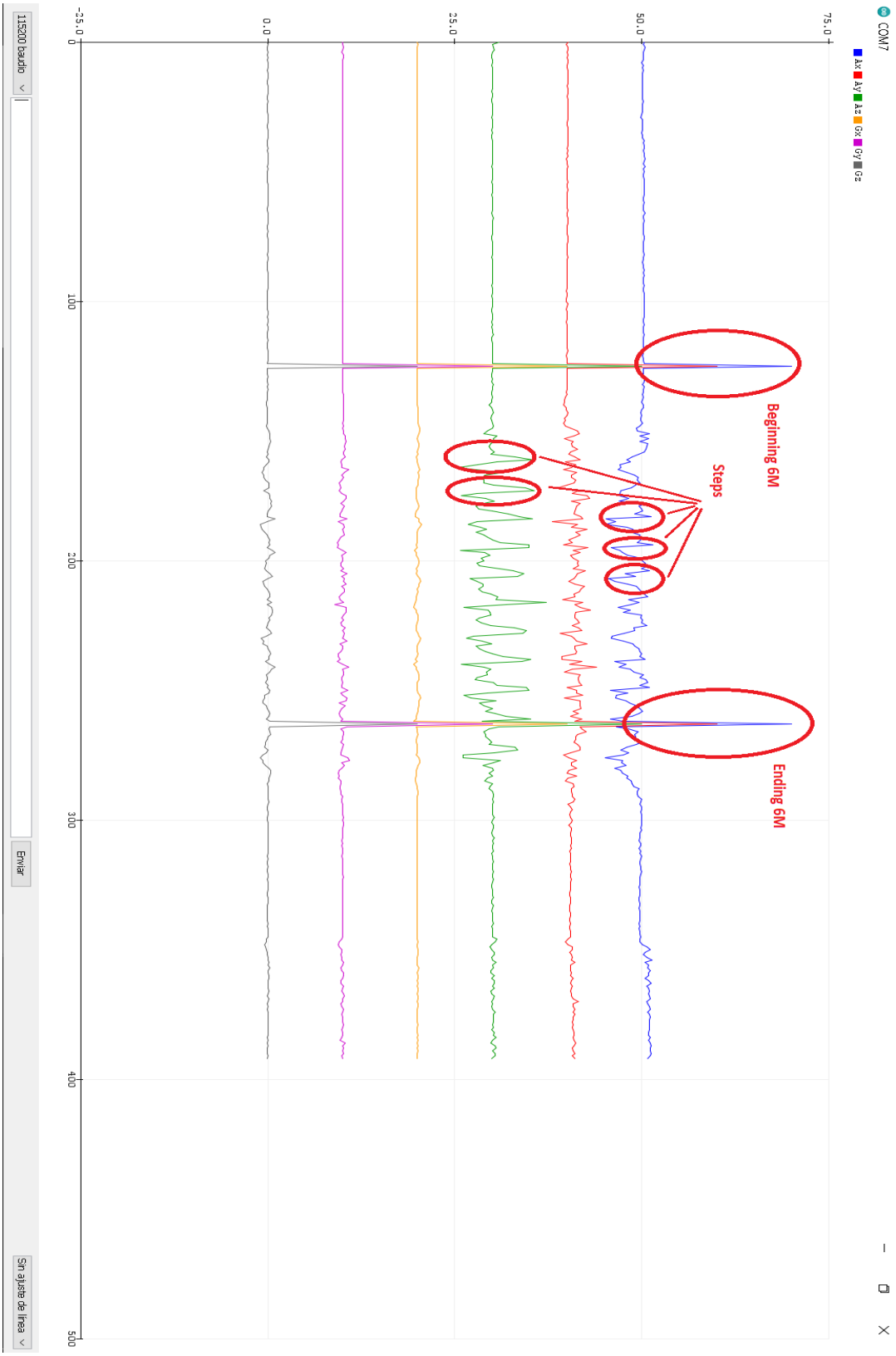


Figure 54 Sample of Direct Signal from Plotter

The times collected for each patient are the following [42]:

Patient	**Times**
63 (hecho)	2M4 1st Round → 2,4 s 2M4 2nd Round → 3 s 6M 1st Round → 6,4 s 6M 2nd Round → 6,2 s
140 (hecho)	2M4 1st Round → 4,2 s 2M4 2nd Round → 4 s 6M 1st Round → 7,4 s 6M 2nd Round → 8,4 s
215 (Hecho, luego quitar)	2M4 1st Round → 2,3 s 2M4 2nd Round → 2,3 s 6M 1st Round → 6,5 s 6M 2nd Round → 6,2 s
216 (Hecho, luego quitar)	2M4 1st Round → 5,2 s 2M4 2nd Round → 6,1 s 6M 1st Round → 11,5 s 6M 2nd Round → 11,9 s
219	2M4 1st Round → 2,4 s 2M4 2nd Round → 3 s 6M 1st Round → 6,4 s 6M 2nd Round → 6,2 s
223 (hecho)	2M4 1st Round → 5 s 2M4 2nd Round → 4,4 s 6M 1st Round → 9,8 s 6M 2nd Round → 10,3 s
249 (hecho)	2M4 1st Round → 4,9 s 2M4 2nd Round → 5 s 6M 1st Round → 11,3 s 6M 2nd Round → 11,8 s
302 (hecho)	2M4 1st Round → 2,6 s 2M4 2nd Round → 2,8 s 6M 1st Round → 6,6 s 6M 2nd Round → 6,5 s

340	2M4 1st Round → 2,4 s 2M4 2nd Round → 3 s 6M 1st Round → 6,4 s 6M 2nd Round → 6,2 s
388 (hecho)	2M4 1st Round → 2,6 s 2M4 2nd Round → 2,5 s 6M 1st Round → 6,7 s 6M 2nd Round → 7 s
477 (hecho)	2M4 1st Round → 2,4 s 2M4 2nd Round → 2,6s 6M 1st Round → 6,7 s 6M 2nd Round → 6,8 s
576 (Hecho)	2M4 1st Round → 2,7 s 2M4 2nd Round → 2,9 s 6M 1st Round → 7,7 s 6M 2nd Round → 7,2 s
643 (hecho	Error tracking parameters
661 (hecho	2M4 1st Round → 5 s 2M4 2nd Round → 4,2 s 6M 1st Round → 6,5 s 6M 2nd Round → 6,7 s
821	Not clear results
996 (hecho)	2M4 1st Round → 4,66 s 2M4 2nd Round → 4,51 s 6M 1st Round → 10,3 s 6M 2nd Round → 10,73 s

The details for each graphic are gathered in the Annex.

7.2 Individual Homes

In this case, the system will be installed as described in the ActiveUP Project. The patient will receive each one an equipment consisting on, besides one Wearable, a Gaitspeed and Chairstand device, as long as a Broker, in order to connect with each other.

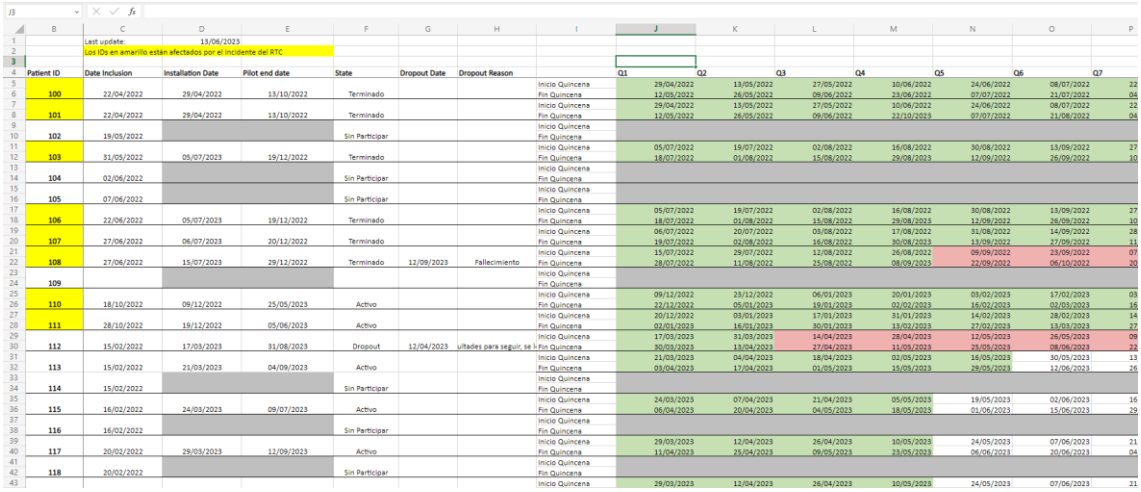
In case the patient has no internet connection, they will also receive a router with a SIM card attached to it with 4GB of data.

The patient is requested to do some exercises every 2 weeks regarding the Gaitspeed and Chairstand devices, to compare the exercises with the collected data taken by the Wearable device. The main idea is that the users wear the device during all day and take it off by the time to go to sleep. However, as some patients have complained that the Wearable is sometimes uncomfortable, the minimum requirement to take measurements is 1 or 2 hours per day, more is optional. The ideal situation is that they wear the device during the time they are more active.

The document where all data are collected is called Adherence Document and is stored in the cloud. This document is divided into three parts separated by tabs:

- One for the dates of the pilot.
- One for the measurements taken by the Wearable, the Gaitspeed and the Chair Stand devices.
- The last one to check the communication with the user and detail the possible failures and disadvantages they have faced.

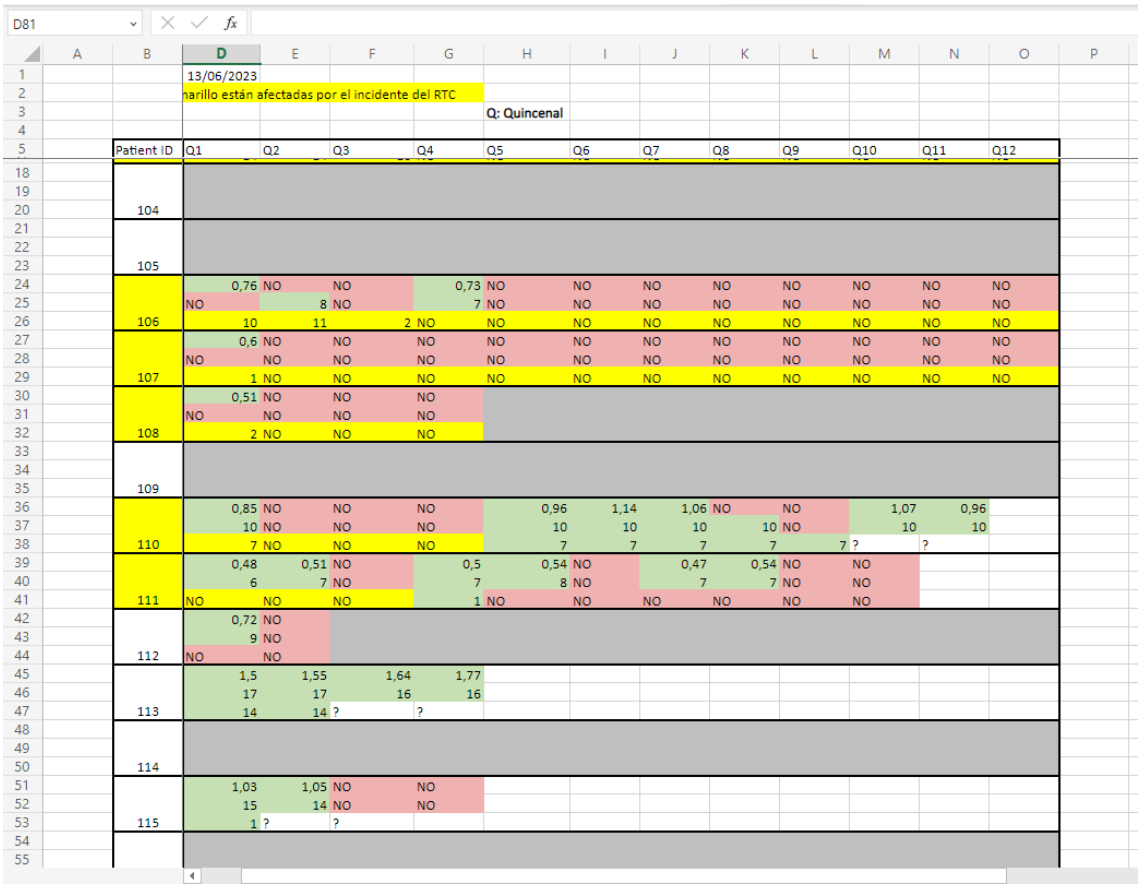
The tab of the dates marks the inclusion date, the installation date, the end date, and the data of dropout, if it would, for each user. It also separates the two weeks for each measurement taken by the Gaitspeed and Chairstand:



	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Last update:	13/06/2023												
2		Los IDs en amarillo están afectados por el incidente del RTC													
3															
4	Patient ID	Date Inclusion	Installation Date	Pilot end date	State	Dropout Date	Dropout Reason		Q1	Q2	Q3	Q4	Q5	Q6	Q7
5								Inicio Quincena	29/04/2022	13/05/2022	27/05/2022	10/06/2022	24/06/2022	08/07/2022	22
6	100	22/04/2022	29/04/2022	13/10/2022	Terminado			Fin Quincena	12/05/2022	26/05/2022	09/06/2022	23/06/2022	07/07/2022	21/07/2022	04
7								Inicio Quincena	29/04/2022	13/05/2022	27/05/2022	10/06/2022	24/06/2022	08/07/2022	22
8	101	22/04/2022	29/04/2022	13/10/2022	Terminado			Fin Quincena	12/05/2022	26/05/2022	09/06/2022	23/10/2023	07/07/2022	31/08/2022	04
9								Inicio Quincena							
10	102	19/05/2022			Sin Participar			Fin Quincena							
11								Inicio Quincena	05/07/2022	19/07/2022	02/08/2022	16/08/2022	30/08/2022	13/09/2022	27
12	103	31/05/2022	05/07/2023	19/12/2022	Terminado			Fin Quincena	18/07/2022	01/08/2022	15/08/2022	29/08/2023	12/09/2022	26/09/2022	10
13								Inicio Quincena							
14	104	02/06/2022			Sin Participar			Fin Quincena							
15								Inicio Quincena							
16	105	07/06/2022			Sin Participar			Fin Quincena							
17								Inicio Quincena	05/07/2022	19/07/2022	02/08/2022	16/08/2022	30/08/2022	13/09/2022	27
18	106	22/06/2022	05/07/2023	19/12/2022	Terminado			Fin Quincena	18/07/2022	01/08/2022	15/08/2022	29/08/2023	12/09/2022	26/09/2022	10
19								Inicio Quincena	06/07/2022	20/07/2022	03/08/2022	17/08/2022	31/08/2022	14/09/2022	28
20	107	27/06/2022	06/07/2023	20/12/2022	Terminado			Fin Quincena	19/07/2022	02/08/2022	16/08/2022	30/08/2023	13/09/2022	27/09/2022	11
21								Inicio Quincena	15/07/2022	29/07/2022	12/08/2022	26/08/2022	09/09/2022	23/09/2022	07
22	108	27/06/2022	15/07/2023	29/12/2022	Terminado	12/09/2023	Fallecimiento	Fin Quincena	28/07/2022	11/08/2022	25/08/2022	08/09/2023	22/09/2022	06/10/2022	20
23								Inicio Quincena							
24	109							Fin Quincena							
25								Inicio Quincena	09/12/2022	23/12/2022	06/01/2023	20/01/2023	03/02/2023	17/02/2023	03
26	110	18/10/2022	09/12/2022	25/05/2023	Activo			Fin Quincena	22/12/2022	05/01/2023	19/01/2023	02/02/2023	16/02/2023	02/03/2023	16
27								Inicio Quincena	20/12/2022	03/01/2023	17/01/2023	31/01/2023	14/02/2023	28/02/2023	14
28	111	28/10/2022	19/12/2022	05/06/2023	Activo			Fin Quincena	02/01/2023	16/01/2023	30/01/2023	13/02/2023	27/02/2023	13/03/2023	27
29								Inicio Quincena	17/03/2023	31/03/2023	14/04/2023	28/04/2023	12/05/2023	26/05/2023	09
30	112	15/02/2022	17/03/2023	31/08/2023	Dropout	12/04/2023	ultades para seguir, se l	Fin Quincena	30/03/2023	13/04/2023	27/04/2023	11/05/2023	25/05/2023	08/06/2023	22
31								Inicio Quincena	21/03/2023	04/04/2023	18/04/2023	02/05/2023	16/05/2023	30/05/2023	13
32	113	15/02/2022	21/03/2023	04/09/2023	Activo			Fin Quincena	03/04/2023	17/04/2023	01/05/2023	15/05/2023	29/05/2023	12/06/2023	26
33								Inicio Quincena							
34	114	15/02/2022			Sin Participar			Fin Quincena							
35								Inicio Quincena	24/03/2023	07/04/2023	21/04/2023	05/05/2023	19/05/2023	02/06/2023	16
36	115	16/02/2022	24/03/2023	09/07/2023	Activo			Fin Quincena	06/04/2023	20/04/2023	04/05/2023	18/05/2023	01/06/2023	15/06/2023	29
37								Inicio Quincena							
38	116	16/02/2022			Sin Participar			Fin Quincena							
39								Inicio Quincena	29/03/2023	12/04/2023	26/04/2023	10/05/2023	24/05/2023	07/06/2023	21
40	117	20/02/2022	29/03/2023	12/09/2023	Activo			Fin Quincena	11/04/2023	25/04/2023	09/05/2023	23/05/2023	06/06/2023	20/06/2023	04
41								Inicio Quincena							
42	118	20/02/2022			Sin Participar			Fin Quincena							
43								Inicio Quincena	29/03/2023	12/04/2023	26/04/2023	10/05/2023	24/05/2023	07/06/2023	21

Figure 55 Tab of Dates and Users

The tab of the measurements taken marks the measurements taken by the devices. It includes one column per 2 weeks, making a total of 12 columns (6 months). Each column has a line for the gait speed measurement, one for the chair stand measurement, and one for the days that user has been wearing the device. Because sometimes the Gaitspeed and Chairstand devices did not work well, sometimes the measurements are taking raw.



	A	B	D	E	F	G	H	I	J	K	L	M	N	O	P
1			13/06/2023												
2			narillo están afectadas por el incidente del RTC												
3							Q: Quincenal								
4															
5		Patient ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	
18															
19															
20		104													
21															
22															
23		105													
24			0,76	NO	NO	0,73	NO	NO	NO	NO	NO	NO	NO	NO	
25			NO	8	NO	7	NO	NO	NO	NO	NO	NO	NO	NO	
26		106	10	11		2 NO	NO	NO	NO	NO	NO	NO	NO	NO	
27			0,6	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	
28			NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	
29		107	1	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	
30			0,51	NO	NO	NO									
31			NO	NO	NO	NO									
32		108	2	NO	NO	NO									
33															
34															
35		109													
36			0,85	NO	NO	NO	0,96	1,14	1,06	NO	NO	1,07	0,96		
37			10	NO	NO	NO	10	10	10	10	NO	10	10		
38		110	7	NO	NO	NO	7	7	7	7	7	?	?		
39			0,48	0,51	NO	0,5	0,54	NO	0,47	0,54	NO	NO			
40			6	7	NO	7	8	NO	7	7	NO	NO			
41		111	NO	NO	NO	1	NO	NO	NO	NO	NO	NO			
42			0,72	NO											
43			9	NO											
44		112	NO	NO											
45			1,5	1,55	1,64	1,77									
46			17	17	16	16									
47		113	14	14	?	?									
48															
49															
50		114													
51			1,03	1,05	NO	NO									
52			15	14	NO	NO									
53		115	1	?	?										
54															
55															

Figure 56 Tab of Exercises for each User

The final tab is made to maintain personal contact with each user. Unlike the tab of the dates and the measurements, this tab has one column per week, making a total of 24 columns (6 months). Each column has two internal columns, one for pointing the type of communication (online or face-to-face) and the other for the detailed information gathered from the patient.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
11	106	No		No		No		No		No		No		No		No
12	107	No		No		No		No		No		No		No		No
13	108	No		No		No		Online	Fallecimiento (me entere varios meses más tarde)							
14	109															
15	110	Face-to-Face	Instalación y fallo de la marcha y las levantadas.	Online	Dolor del estómago. Deja el Wearable por unas semanas para ver como evoluciona	Online	Va a estar unos meses en su pueblo	No		No		Face-to-Face	Recogida de datos y ejercicios (no ha sido capaz de hacerlos sola)	Online	No logra hacer el ejercicio de la marcha	Face-to-Face
16	111	Face-to-Face	Instalación y primeros ejercicios de vivifrail	No		Online	Posible agotamiento datos	Online	Posible agotamiento datos	Face-to-Face	Hacer ejercicios vivifrail. Dice que le molesta el Wearable	No		Online	Está en su pueblo	Online
17	112	Face-to-Face	Instalación y primeros ejercicios de vivifrail	Face-to-Face	Entrega tarjeta SIM	Online	Salida del piloto (dice que no quiere continuar porque tiene parkinson y no se siente con fuerzas)									
18	113	Face-to-Face	Instalación y primeros ejercicios de vivifrail	Face-to-Face	Recogida de datos y ejercicios vivifrail	No		Online	Se los está poniendo todos los días según él	Online	Pendiente de llamar para recoger datos	No		No		No
19	114															
20	115	Face-to-Face	Instalación y primeros ejercicios de vivifrail	Online	(Ha hablado Walter)	Online	(Ha hablado Walter)	Face-to-Face	Pruebas vivifrail y recogida de datos. Se queja mucho del Wearable porque le molesta	Online	Pendiente de llamar para recoger datos	No		No		No
21	116															
22	117	Face-to-Face	Instalación y primeros ejercicios de vivifrail	No		Online	No ha podido hacer los ejercicios porque no le aparecían como actividades pendientes	Face-to-Face	Pruebas vivifrail y recogida de datos. Se pone a menudo el Wearable mal y se queja de que se le nota	Online	Pendiente de llamar para recoger datos	No		No		No
23	118															
24	119	Face-to-Face	Instalación y primeros ejercicios de vivifrail	No		No		No		Online	Pendiente de llamar para recoger datos. Ver además por qué no hace los ejercicios	No		No		No
25	120	Face-to-Face	Instalación y primeros ejercicios de vivifrail	Online	Se queja de que no le aparecen los ejercicios en la tablet	No		No		Online	Pendiente de llamar para recoger datos. Ver además por qué no hace los ejercicios	No		No		No

Figure 57 Tab of Detailed Communication between Patient and Caretaker

Here there is the link to access to the Adherence Document: [43]

The data collected since the beginning of the project are the following:

Patient	**Data (MB)**
100	490
101	848
103	9720
106	1750
107	35,9
110	694
112	1
113	1310
115	903
117	137
123	217
Total	16,2 GB

To access this data, we can access by Filezilla (or another server board), with the following data, by SFTP:

- Public IP of the server (138.4.131.3)
- Port: 16957
- User: ageinglab
- Rsa key

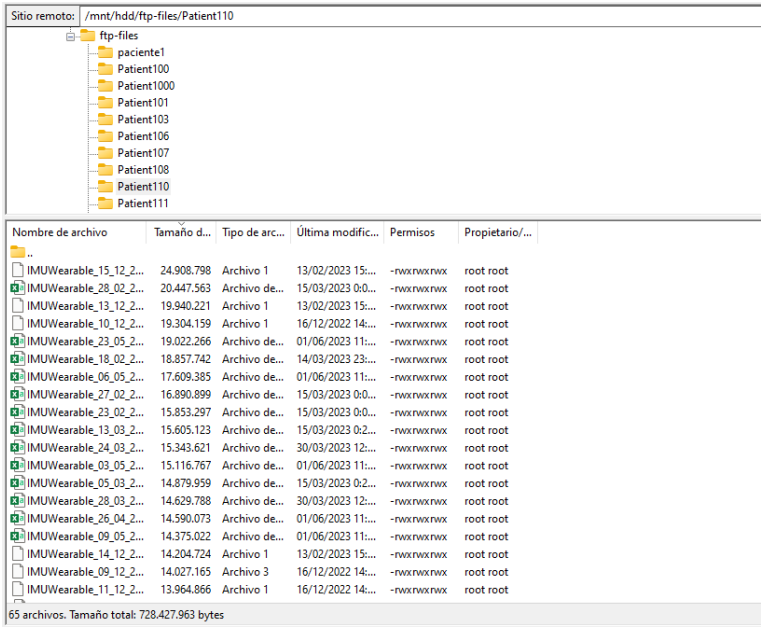


Figure 58 Example: Remote Data of Patient 110

8 Conclusion

To analyze the results of the wearable, we surveyed the patients of the pilot, to ask them about the usability and functionality of the wearable. The survey consists of a simple questionnaire that ask about the satisfaction with the final device, if they have had some inconvenient using it, if the device has caused them some discomfort, if they recommend other people to use it, etc. This questionnaire is anonymous, and it will only be used with research purposes. The link to the complete document is here [44]

The questionnaire has the following structure:

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**					
2.	**El sistema me resultó innecesariamente complejo.**					
3.	**Creo que el sistema es bastante fácil de utilizar.**					
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**					
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**					
6.	**Opino que hubo demasiada inconsistencia en el sistema.**					
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**					
8.	**Me sentí algo incómodo al utilizar este sistema.**					
9.	**Me sentí muy seguro al utilizar este sistema.**					
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**					

The questionnaire has been surveyed to 6 different patients, though it will be made in the future for more, to make more complete research. The results are the following:

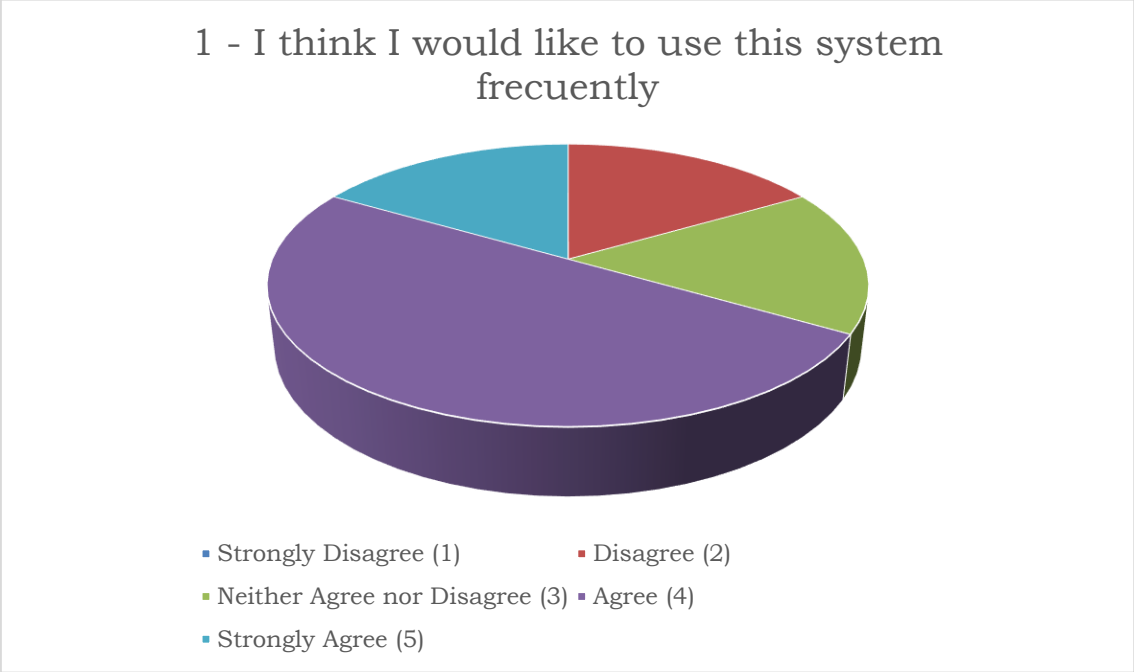


Figure 59 Question 1 Summary

The first question reflects that a strong portion of user would like to use the system, as most of them have a positive answer about this question.

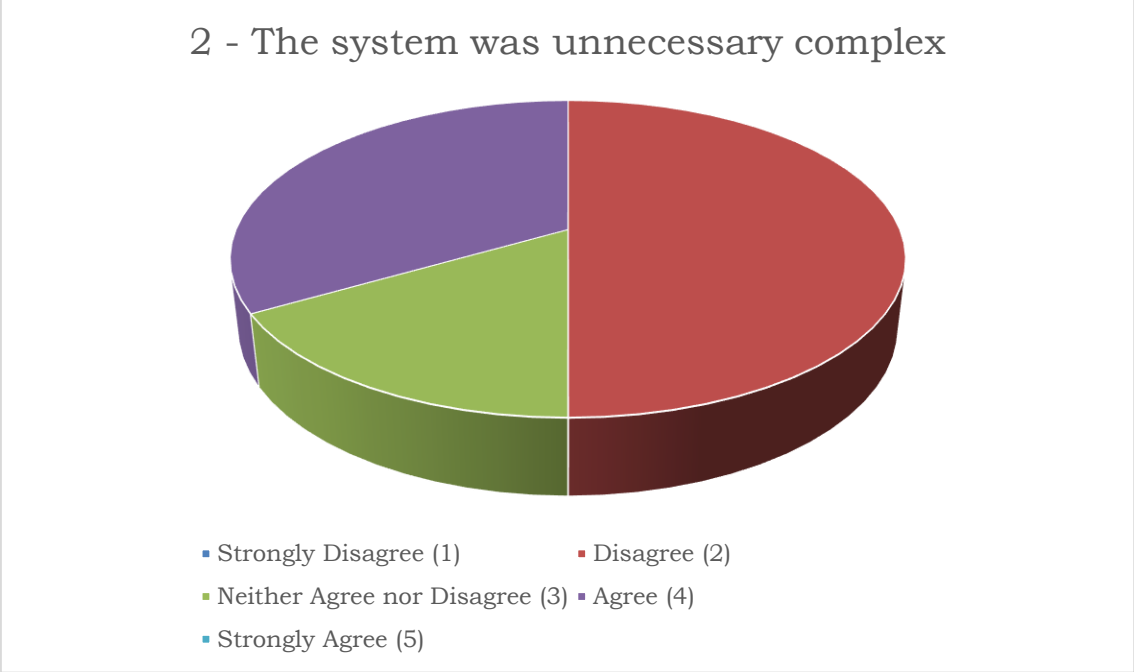


Figure 60 Question 2 Summary

The second question reflects that the system was quite simple, and easy to use for old people, as regarding the Wearable, for the user the only task to do is just wear it and turn the switch on.

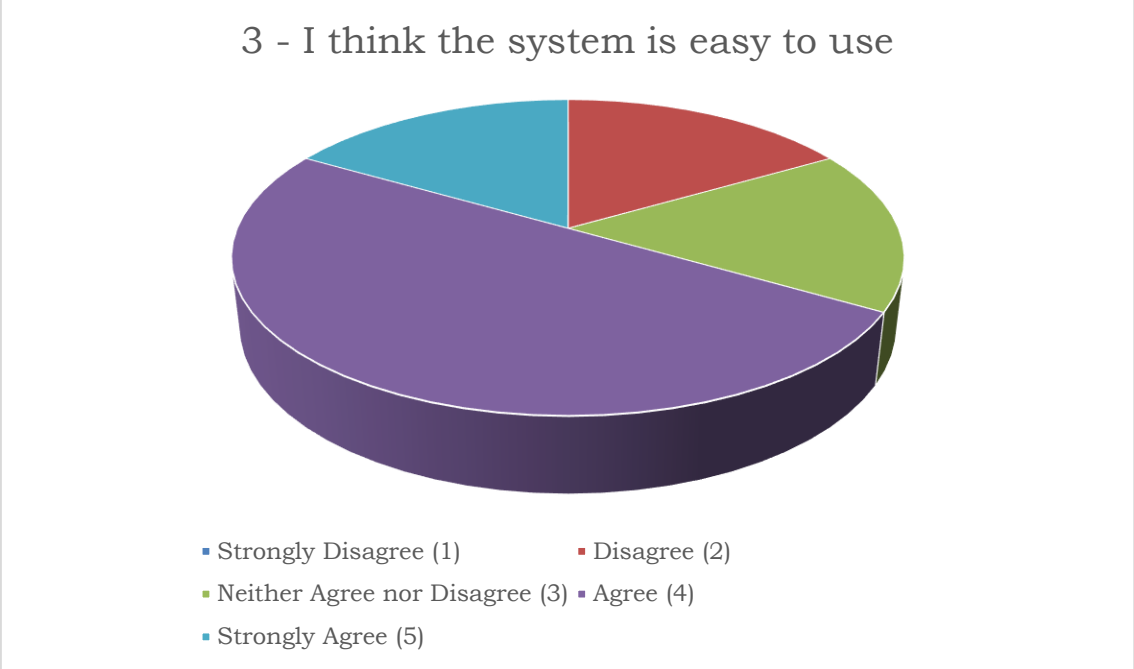


Figure 61 Question 3 Summary

The third question is exactly the opposite of the second, so is obvious that the users think the system is quite easy to use, as the graphics shows in the picture 65.

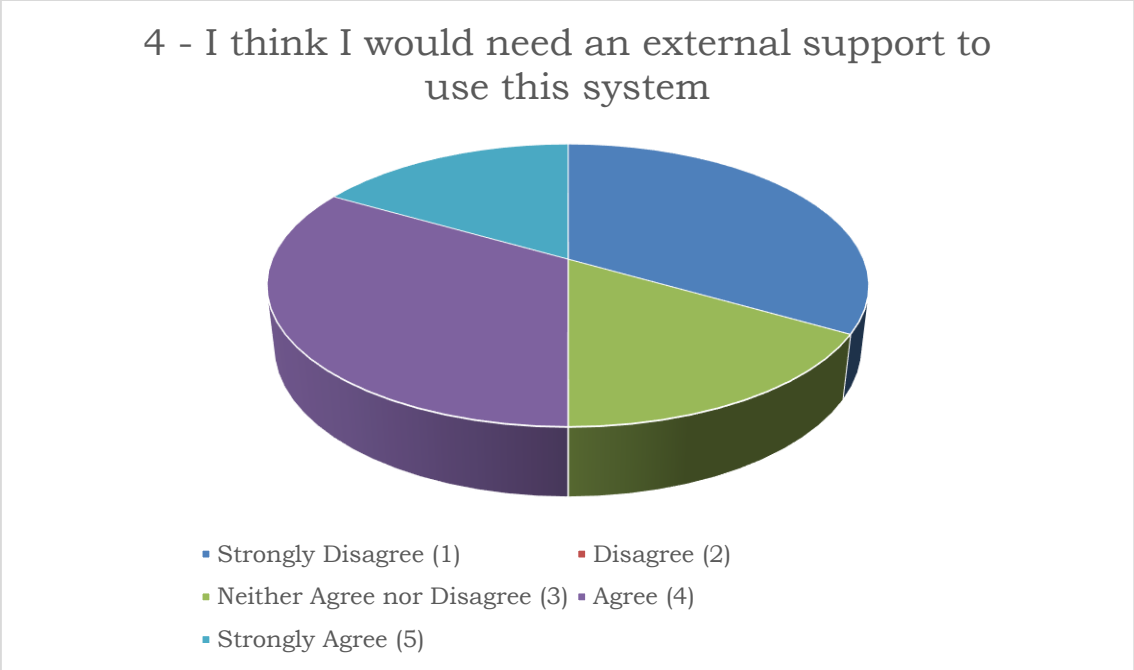


Figure 62 Question 4 Summary

The fourth question is regarding the speed of how much they would take to learn and the obviousness and simplicity of the system, so there is a few differences among users this time, as some of them think that they will learn without support, while others think that they will need despite the ease of the system.

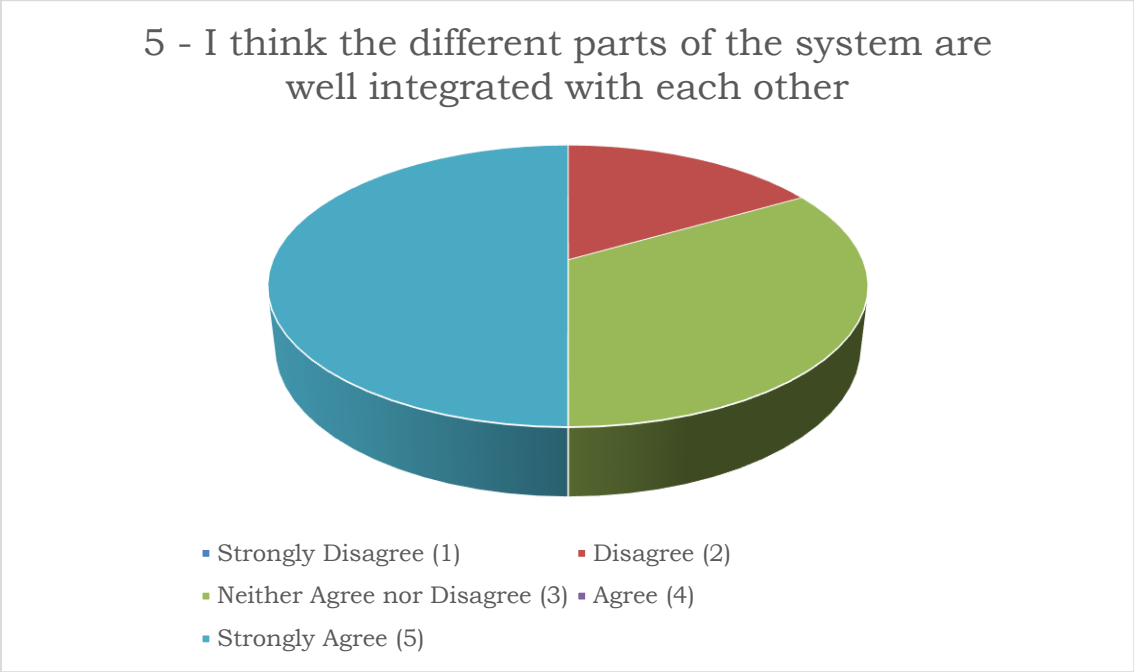


Figure 63 Question 5 Summary

The fifth question is regarding the communication with the components of the system, such as the sensors, the broker and the smartphone or tablet, showing the robustness of the system and the reliability of the one.

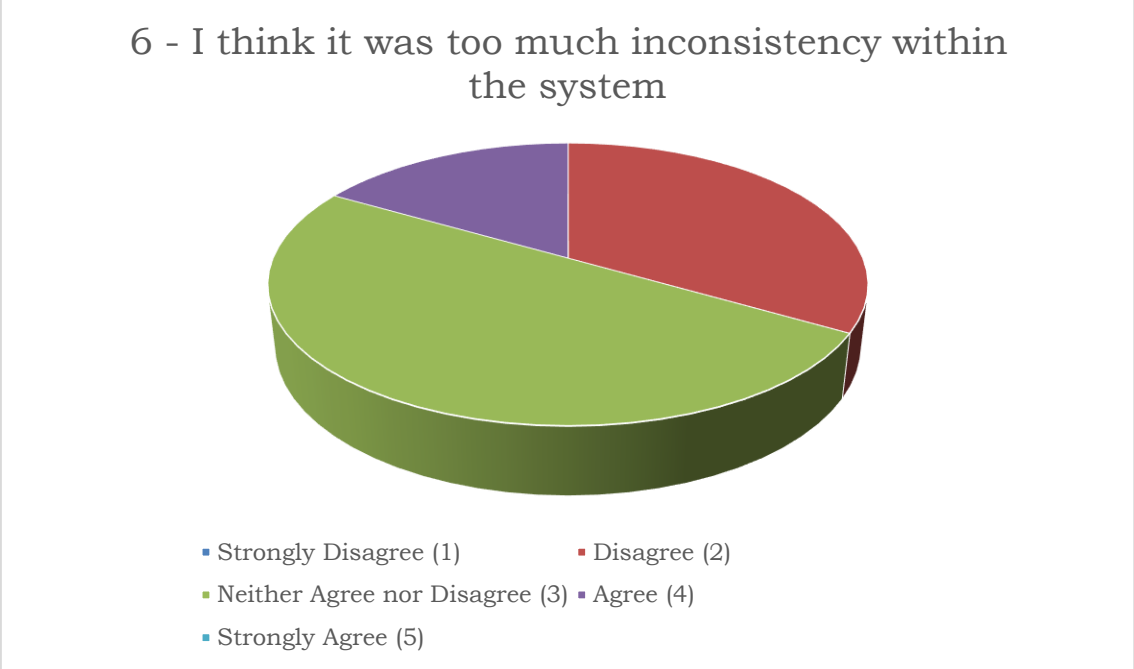


Figure 64 Question 6 Summary

The sixth question regards about the consistency of the system and the opinion of the users shows that their opinions about this topic are well balanced: there are no radical positions and most of the user think in a middle term.

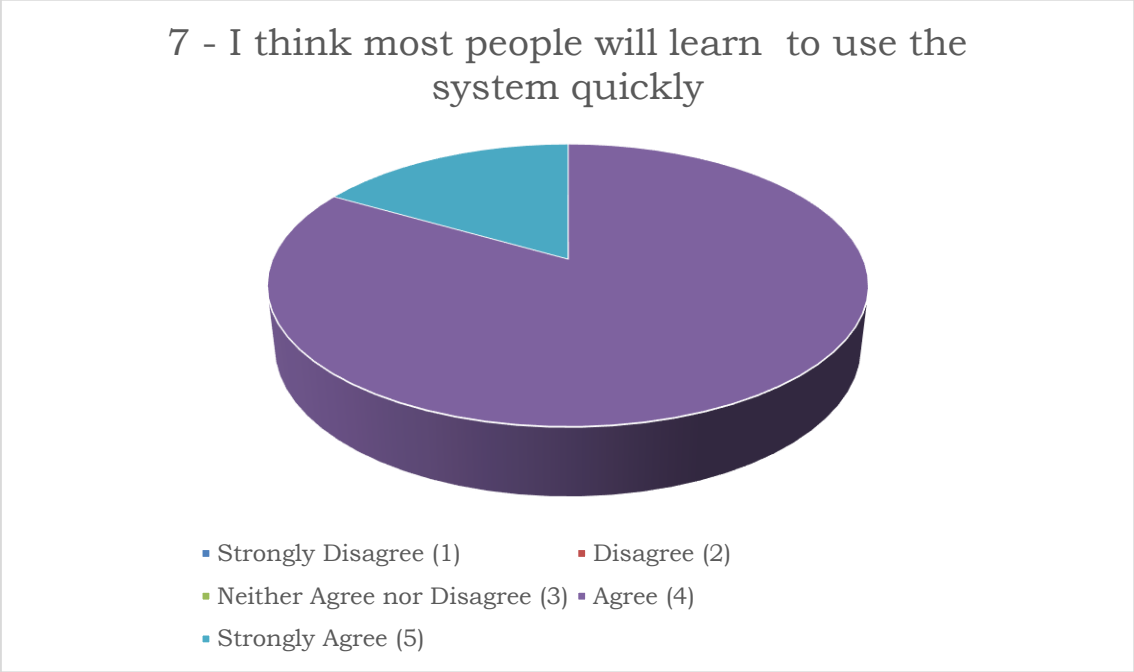


Figure 65 Question 7 Summary

The seventh question reflects how the users think the system is quite simple to use by an external people that has no knowledges about the system, showing a positive opinion about it.

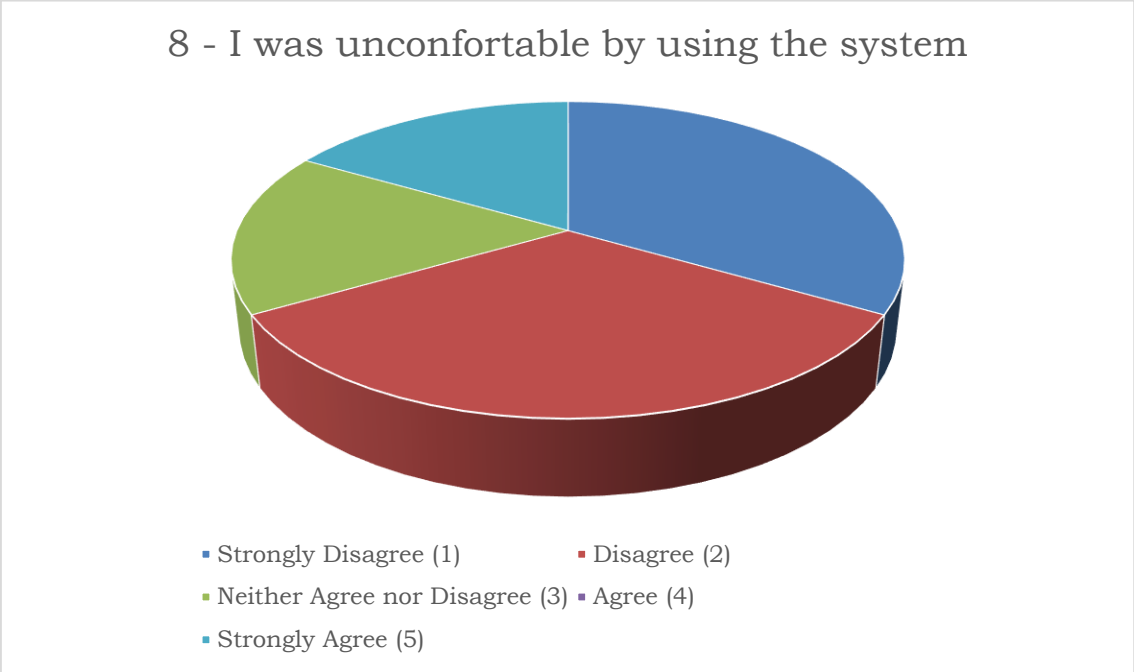


Figure 66 Question 8 Summary

The eight question shows disparity of opinions about it. Most users have complained about the discomfort the wearable has sometimes produced them. The extreme value of this graphic shows this controversial opinion about the topic.

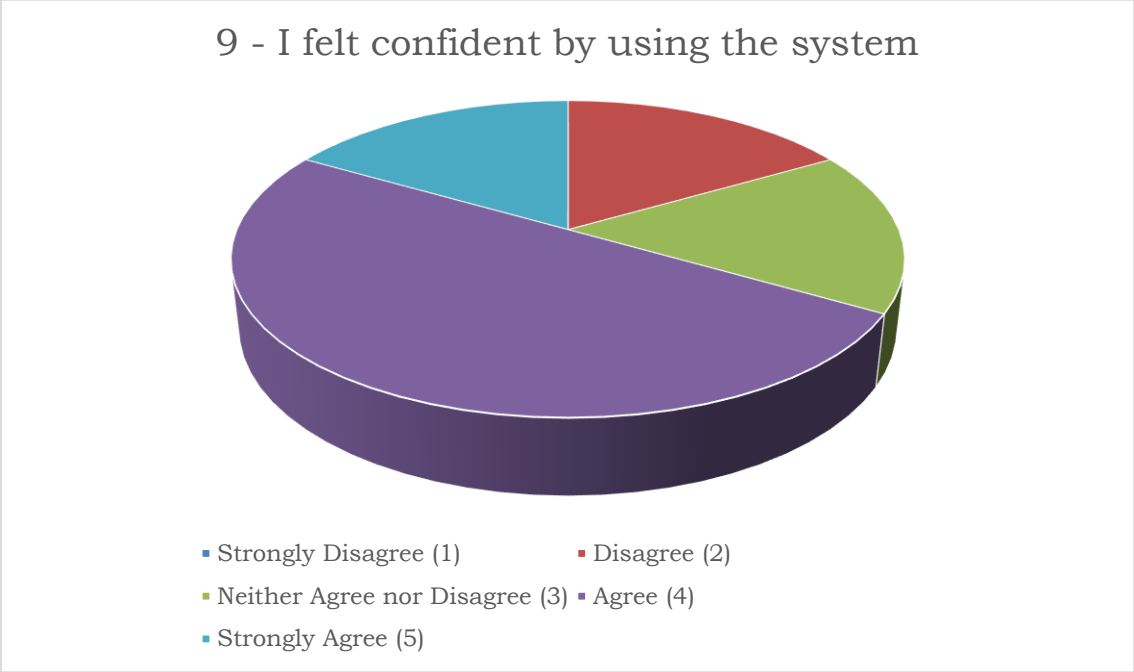


Figure 67 Question 9 Summary

The ninth question reflects that, in general, the users have no concerns of the security of the system. Only one of them has a negative opinion of this question, the rest have either neutral or positive opinion. The tenth question reflects the difficulty of old people to know about the new technologies, as only one of the survey respondents has negative question about it.

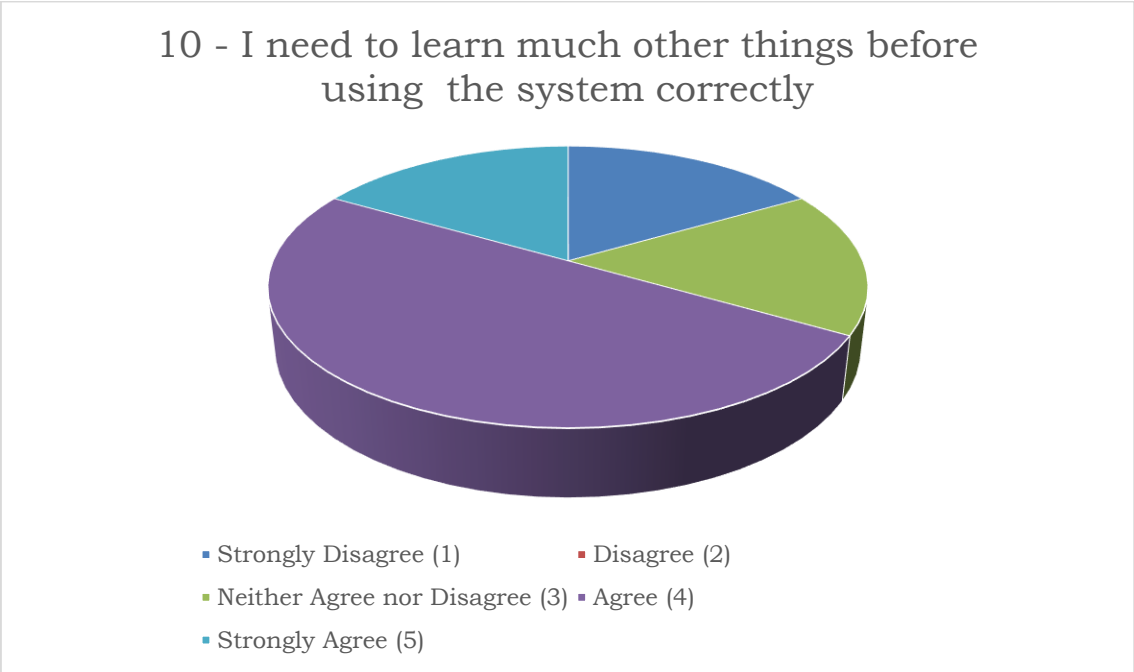


Figure 68 Question 10 Summary

9 Future Work

There are several possible upgrades for the system to make it simpler and improve for the developers, the professionals and caretakers and the users themselves. Some of them regards about programming and technical issues, some of them about the calculation of the algorithm of speed, and some of them about the ubiquity, usability, comfortability, and heuristics.

PROGRAMMING

→ Improve the code, by erasing some functions that are redundant, and some of them that are not necessary, to make the flash program smaller. The current sketch is quite huge, and it has several issues by the time to boot into the flash memory of the program, because of the size of the flash itself. This could be solved partially by burning bootloader program (a program is allocated within the flash, and helps the executable to be load with a simple USB cable, and partitioning the flash memory increasing the size of the reserved memory for the APP (see Figure 73), but there are still better ways to reduce the size of the sketch [45]:

- Avoiding String Objects. Instead trying to cast to other basic type of data, such as char, binary or byte.
- Besides reducing the number of traces, avoid storing them within memory, instead storing them in the sketch storage.
- Declaring local variables instead of global variables.

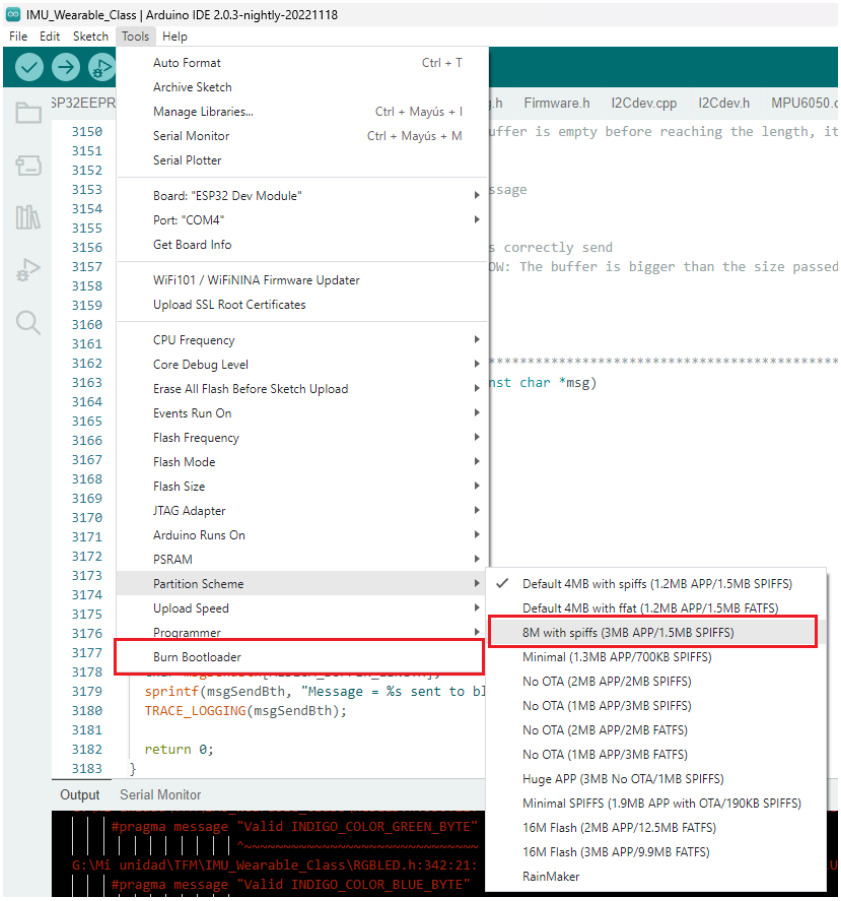


Figure 69 Flash Partition and Bootloader

→ Creating new functionalities that helps the new possible developers to program in the future.

For instance, the main way to trace the code as developer is to plug an USB cable and check the Serial Port Connection to the computer, so that the device can print the output in the screen. This has some advantages, such as the possibility to see where the execution of the process is during all the time, but also has some inconveniences. Because the computer must be connected to the device by a USB cable all the time, some test with patients is not feasible, due to the fact the programmer must follow the user wherever he goes.

A solution to solve this problem could be to print the output of the traces in a Bluetooth device or even within the SD-Card as an especial file to log the execution of the device. By doing this, the limitations of the USB cable are eliminated. This is made partially in the test done in the retirement homes, by merging the COM ports of computers with Bluetooth features.

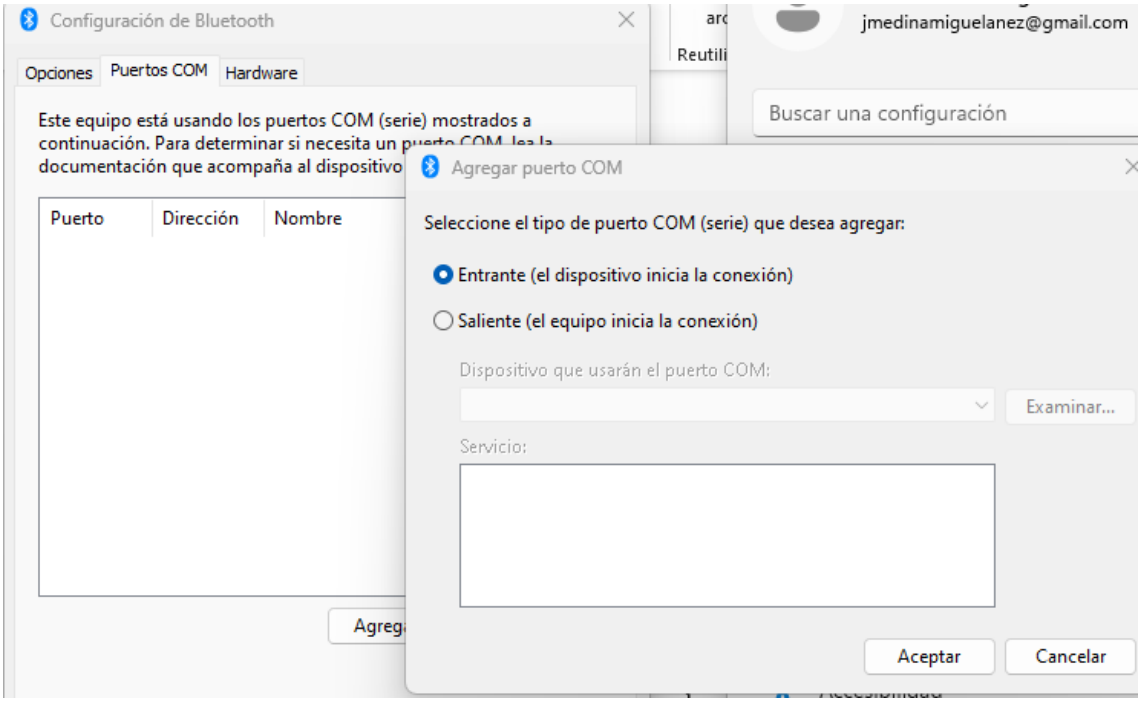


Figure 70 Bluetooth with COM Ports

→ Exploring new possibles of uploading the sketch into the flash memory, such as changing the interface of the Source Code, using other different from Arduino, such as esp-idf (the official for ESP8266 and ESP32), using Micropython, an especial version of Python for microcontrollers arm as instead of C and C++, and even using external tools instead of interfaces, such as scripts or makefile. This is much more complex than using an interface, but it has the advantages of providing much more flexibility by the time to manufacture the product. For instance, if the device has suffered a change in the hardware connections (there has been a change of the pins, or even the main board is different), a normal code would need to be change, because the GPIO are different. But using a makefile is a different story because it can use the GPIOs inputs and outputs as labels that can be added to the makefile, making the compilation much easier for an external operator who has no technical knowledge of programming.

ALGORITHM

Other possible implementation for the future, is using different algorithms that can predict the speed of the user, by processing the data collected by the device, and stored within the CSV files.

There are three different algorithms developed, detailed in different articles, that can predict speed: Czech, Mueller and Urbanek [46]. In summary, the three of them takes the speed by the bauds obtained with the Fourier Transformed, the Butterworth filters and the Dominant Frequency.

Besides, we are in the process of training a new model based on the relation of standard deviation and speed. After collecting 200 samples of 4 different people, we trained a logarithmic model with each standard deviation of the accelerations and with this result:

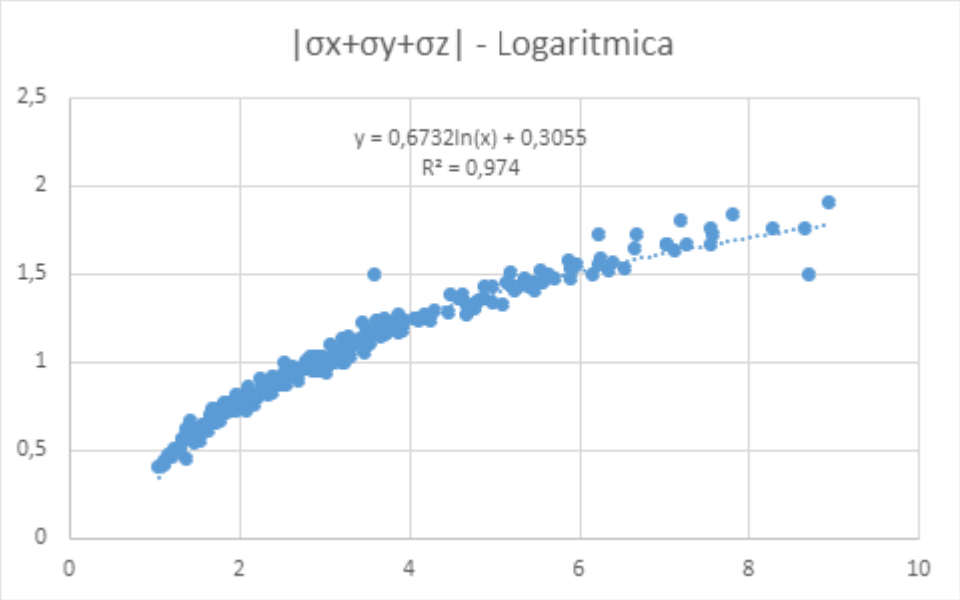


Figure 71 Regression Model for Speed

The relation between the standard deviation a real speed is clearly logarithmic, as X shows the standard deviation for the module of the acceleration, and Y shows the real speed taking by dividing 6 m of distance by the time spent during the walking. The results are very optimistic and there have been applied with some patients, with a maximum error of 10-15%

USABILITY

Other topic that concerns about the wearable is the usability and comfortability. As said before, some patients have complained about the wearable being quite uncomfortable. It is being studied ways to make it more comfortable, diminishing its size or even using instead a necklace or a wristlet instead, which makes the user more predisposed to wear it.

10 References

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4098658/

[2] https://www.elsevier.es/es-revista-revista-espanola-geriatria-gerontologia-124-articulo-aplicacion-escala-fragilidad-edmonton-poblacion-S0211139X17300926#:~:text=Generalmente%20se%20ha%20definido%20fragilidad,estimada%20por%20fuerza%20de%20prehensi%C3%B3n).

[3] https://www.agespace.org/tech/elderly-home-sensors

[4] https://fortune.com/well/2023/03/23/can-sit-stand-test-predict-how-long-you-will-live/

[5] https://es.wikipedia.org/wiki/Internet_de_las_cosas

[6] https://aws.amazon.com/es/what-is/iot/

[7] https://www.paessler.com/es/it-explained/mqtt

[8] https://www.arduino.cc/

[9] https://es.wikipedia.org/wiki/Arduino

[10] https://es.wikipedia.org/wiki/Arduino_IDE

[11] https://docs.arduino.cc/tutorials/

[12] https://es.wikipedia.org/wiki/Bluetooth#:~:text=Bluetooth%20es%20una%20especificaci%C3%B3n%20industrial,ISM%20de%20los%202.4%20GHz.

[13] https://es.wikipedia.org/wiki/Wifi

[14] https://en.wikipedia.org/wiki/802.11_Frame_Types

[15] https://vanhunteradams.com/Protocols/UART/UART.html

[16] https://www.circuitbasics.com/basics-uart-communication/

[17] https://es.wikipedia.org/wiki/I%C2%B2C

[18] https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[19] https://es.wikipedia.org/wiki/Serial_Peripheral_Interface

[20] https://panamahitek.com/como-funciona-el-protocolo-spi/

[21] https://docs.google.com/document/d/1hml7hNAeGzL6e-8TkViS6213NMUWm9BW/edit

[22] https://docs.google.com/document/d/1cdtDU72c8BXaNHvTT5ukk723iUMmr8OX/edit

[23] https://docs.google.com/document/d/1c2Vhme3fWxGmNU7a_mtNIW-Qv0Ho1a55/edit#heading=h.3znysh7

[24] https://activeupprod.mgbiomed.es:444/docs

[25] https://es.wikipedia.org/wiki/ESP32

[26] https://www.dfrobot.com/product-1590.html

[27] https://wiki.dfrobot.com/FireBeetle_ESP32_IOT_Microcontroller(V3.0)__Supports_Wi-Fi_&_Bluetooth__SKU__DFR0478

[28] https://diyi0t.com/reduce-the-esp32-power-consumption/?utm_content=cmp-true

[29] https://drive.google.com/drive/u/1/folders/1KXJZfNCScW7UKeZ11LzvohhTOpgsblyn

[30] https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html

[31] https://drive.google.com/drive/u/1/folders/1KXJZfNCScW7UKeZ11LzvohhTOpgsblyn

[32] https://www.luisllamas.es/reloj-y-calendario-en-arduino-con-los-rtc-ds1307-y-ds3231/

[33] https://www.cqrobot.com/index.php?route=product/product&product_id=1596

[34] https://www.utmel.com/components/tp4056-standalone-linear-li-lon-battery-charger-datasheet-schematics-and-current?id=689

[35] https://drive.google.com/drive/u/0/folders/1P7Y2pmDrze5z7QX5Mci2c1zOIoZhGwuW

[36] https://drive.google.com/drive/u/0/folders/1T2nr5V6wVbuAAza900i3ejGW06Uuarxa

[37] https://drive.google.com/drive/u/0/folders/1HDaIFnMga8AzvUPuTRiK6xpE_tXoTrOr

[38] https://drive.google.com/drive/u/0/folders/1QIB8BTpFa4fEYe8Ui9ipwO2XaSSbS74k

[39] https://www.madrid.es/portales/munimadrid/es/Inicio/Mayores/Direcciones-y-telefonos/Centro-de-Dia-Municipal-Jose-Manuel-Bringas-/?vgnextfmt=default&vgnextoid=d3b7d4985261c010VgnVCM1000000b205a0aRCRD&vgnextchannel=ac6931d3b28fe410VgnVCM1000000b205a0aRCRD

[40] https://activaedades.com/

[41] https://www.comunidad.madrid/hospital/getafe/

[42] https://drive.google.com/drive/u/1/folders/1wq17U4t4vMGTIFy0eGgANY_yvhKEvOmv

[43] https://docs.google.com/spreadsheets/d/1FYRrMTnxr0B-02n5PgaQd46m4EzogSFn/edit#gid=1572369872

[44] https://drive.google.com/drive/u/1/folders/1b1dze6CVR53kpUoZSqJlFGfKwqgElW72

[45] https://support.arduino.cc/hc/en-us/articles/360013825179

[46] https://drive.google.com/drive/u/2/folders/1Ba-yec7sRmQe35Ul5L2KX65hflNdAPRf

Annex

A Testing

- ***DS3231:***

  → castTimeZonetoInt

Method	**castTimeZonetoInt**
script	DS3231_castTimeZonetoInt.ino
Testbench	DS3231_castTimeZonetoInt.xlsx
Testcases	40
Statements	22

Testcase	**1**
Definition	Invalid length of the String
Arguments	strTZ → “abcd”
Expected Attributes	Timezone → *Irrelevant*
Real Attributes	Timezone → *Irrelevant*
Expected Outcome	CASTTIMEZONETOINT_INVALID_TIMEZONE (-1)
Real Outcome	CASTTIMEZONETOINT_INVALID_TIMEZONE (-1)
Expected Traces	1, 21, 22
Real Traces	1, 21, 22

Testcase	**2**
Definition	Invalid format of String
Arguments	strTZ → “abcde”
Expected Attributes	Timezone → *Irrelevant*
Real Attributes	Timezone → *Irrelevant*
Expected Outcome	CASTTIMEZONETOINT_INVALID_TIMEZONE (-1)
Real Outcome	CASTTIMEZONETOINT_INVALID_TIMEZONE (-1)
Expected Traces	1, 2, 19, 20
Real Traces	1, 2, 19, 20

Testcase	**3**
Definition	UTC = -12
Arguments	strTZ → TIMEZONE_MINUS12 (“-1200”)
Expected Attributes	Timezone → -43200

Real Attributes	Timezone → -43200
Expected Outcome	CASTTIMEZONETOINT_VALID_TIMEZONE (0)
Real Outcome	CASTTIMEZONETOINT_VALID_TIMEZONE (0)
Expected Traces	1, 2 (1 Condition), 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
Real Traces	1, 2 (1 Condition), 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Testcase	**4**
Definition	UTC = -11
Arguments	strTZ → TIMEZONE_MINUS11("-1100")
Expected Attributes	Timezone → -39600
Real Attributes	Timezone → -39600
Expected Outcome	CASTTIMEZONETOINT_VALID_TIMEZONE (0)
Real Outcome	CASTTIMEZONETOINT_VALID_TIMEZONE (0)
Expected Traces	1, 2 (2 Condition), 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
Real Traces	1, 2 (2 Condition), 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Coverage	**10**
(%)	100

→ checkConnections

Method	**checkConnections**
script	Button_checkConnections.ino
Testbench	Button_checkConnections.xlsx
Testcases	3
Statements	12

Testcase	**1**
Definition	The device has no connection with RTC
Expected Outcome	DS3231_I2C_DISCONNECTED (-1)
Real Outcome	DS3231_I2C_DISCONNECTED (-1)
Expected Traces	1, 2, 10, 11, 12
Real Traces	1, 2, 10, 11, 12
Testcase	**2**

Definition	The device has lost power supply
Expected Outcome	DS3231_I2C_CONNECTED (0)
Real Outcome	DS3231_I2C_CONNECTED (0)
Expected Traces	1, 2, 3, 4, 5, 6, 9
Real Traces	1, 2, 3, 4, 5, 6, 9

Testcase	**3**
Definition	The RTC is in perfect condition
Expected Outcome	DS3231_I2C_CONNECTED (0)
Real Outcome	DS3231_I2C_CONNECTED (0)
Expected Traces	1, 2, 3, 4, 5, 6, 9
Real Traces	1, 2, 3, 4, 5, 6, 9

Coverage	**10**
(%)	100

- ***ESP32EEPROM:***
  - ➔ ***getIntEEPROM***

Method	**getIntEEPROM**
script	ESP32EEPROM_getIntEEPROM.ino
Testbench	ESP32EEPROM_getIntEEPROM.xlsx
Testcases	6
Statements	10

Testcase	**1**
Definition	The address is minor than 0
Arguments	addr → -1 val → *irrelevant*
Value Stored	*Irrelevant*
Expected Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Expected Traces	1, 2, 3
Real Traces	1, 2, 3

Testcase	**2**
Definition	The address is bigger than the maximum size minus 1.

Arguments	addr → 512 val → *irrelevant*
Value Stored	*Irrelevant*
Expected Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Expected Traces	1, 2, 3
Real Traces	1, 2, 3

Testcase	**3**
Definition	The integer occupies 1 byte of memory.
Arguments	addr → 16 val → 167
Value Stored	1 Byte → 0xa7 (167) 2 Byte → 0x00 (0) 3 Byte → 0x00 (0) 4 Byte → 0x00 (0)
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 4, loop (5, 6), 7, 8, 9, 10
Real Traces	1, 4, loop (5, 6), 7, 8, 9, 10

Testcase	**4**
Definition	The integer occupies 2 bytes of memory.
Arguments	addr → 48 val → 23456
Value Stored	1 Byte → 0xa0 (160) 2 Byte → 0x5b (91) 3 Byte → 0x00 (0) 4 Byte → 0x00 (0)
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 4, loop (5, 6), 7, 8, 9, 10
Real Traces	1, 4, loop (5, 6), 7, 8, 9, 10

Testcase	**5**
Definition	The integer occupies 3 bytes of memory.

Arguments	addr → 80 val → 12346789
Value Stored	1 Byte → 0xa5 (167) 2 Byte → 0x65 (101) 3 Byte → 0xbc (188) 4 Byte → 0x00 (0)
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 4, loop (5, 6), 7, 8, 9, 10
Real Traces	1, 4, loop (5, 6), 7, 8, 9, 10

Testcase	**6**
Definition	The integer occupies 4 bytes of memory.
Arguments	addr → 128 val → 1567890123
Value Stored	1 Byte → 0xcb (203) 2 Byte → 0x1a (26) 3 Byte → 0x74 (116) 4 Byte → 0x5d (93)
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 4, loop (5, 6), 7, 8, 9, 10
Real Traces	1, 4, loop (5, 6), 7, 8, 9, 10

Coverage	**10**
(%)	100

➔ ***setIntEEPROM***

Method	**setIntEEPROM**
script	ESP32EEPROM_setIntEEPROM.ino
Testbench	ESP32EEPROM_setIntEEPROM.xlsx
Testcases	6
Statements	8

Testcase	**1**
Definition	The address is minor than 0

Arguments	addr → -1 val → *irrelevant*
Value Stored	*Irrelevant*
Expected Outcome	EEPROM_INT_WRITE_INVALID_ADDR (-1)
Real Outcome	EEPROM_INT_WRITE_INVALID_ADDR (-1)
Expected Traces	1, 2 (1 Condition)
Real Traces	1, 2 (1 Condition)

Testcase	**2**
Definition	The address is bigger than the maximum size minus 1.
Arguments	addr → 512 val → *irrelevant*
Value Stored	*Irrelevant*
Expected Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (-1)
Expected Traces	1, 2 (2 Condition)
Real Traces	1, 2 (2 Condition)

Testcase	**3**
Definition	The integer occupies 1 byte of memory.
Arguments	addr → 16 *val → (Expected) 34
Value Stored	1 Byte → 0x22 2 Byte → 0x00 3 Byte → 0x00 4 Byte → 0x00
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 3, loop (4, 5), 6, 7, 8
Real Traces	1, 3, loop (4, 5), 6, 7, 8

Testcase	**4**
Definition	The integer occupies 2 bytes of memory.
Arguments	addr → 48 *val → (expected) 12700
Value Stored	1 Byte → 0x9c

	2 Byte → 0x31 3 Byte → 0x00 4 Byte → 0x00
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 3, loop (4, 5), 6, 7, 8
Real Traces	1, 3, loop (4, 5), 6, 7, 8

Testcase	**5**
Definition	The integer occupies 3 bytes of memory.
Arguments	addr → 80 val → 12345678
Value Stored	1 Byte → 0x4e 2 Byte → 0x61 3 Byte → 0xbc 4 Byte → 0x00
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 3, loop (4, 5), 6, 7, 8
Real Traces	1, 3, loop (4, 5), 6, 7, 8

Testcase	**6**
Definition	The integer occupies 4 bytes of memory.
Arguments	addr → 128 val → 1567890123
Value Stored	1 Byte → 0xd2 2 Byte → 0x02 3 Byte → 0x96 4 Byte → 0x49
Expected Outcome	EEPROM_INT_READ_VALID_ADDR (0)
Real Outcome	EEPROM_INT_READ_INVALID_ADDR (0)
Expected Traces	1, 4, loop (5, 6), 7, 8, 9, 10
Real Traces	1, 4, loop (5, 6), 7, 8, 9, 10

Coverage	**10**
(%)	100

➔ ***isValidASCII***

Method	**isValidASCII**
script	ESP32EEPROM_isValidASCII.ino
Testbench	ESP32EEPROM_isValidASCII.xlsx
Testcases	3
Statements	5

Testcase	**1**
Definition	Readable ASCII Character
Arguments	'c' (99)
Expected Outcome	READABLE_ASCII_CHAR (1)
Real Outcome	READABLE_ASCII_CHAR (1)
Expected Traces	1, 2, 3
Real Traces	1, 2, 3

Testcase	**2**
Definition	Non-Readable ASCII Character (minor than 32).
Arguments	10
Expected Outcome	NOT_READABLE_ASCII_CHAR (0)
Real Outcome	NOT_READABLE_ASCII_CHAR (0)
Expected Traces	1, 2, 4, 5
Real Traces	1, 2, 4, 5

Testcase	**3**
Definition	Readable ASCII Character (bigger than 126)
Arguments	127
Expected Outcome	NOT_READABLE_ASCII_CHAR (0)
Real Outcome	NOT_READABLE_ASCII_CHAR (0)
Expected Traces	1, 2, 4, 5
Real Traces	1, 2, 4, 5

Coverage	**10**
(%)	100

- ***RGBLED:***

  ➔ ***setColor***

Method	**setColor**
script	RGBLED_setColor.ino
Testbench	RGBLED_setColor.xlsx
Testcases	8
Statements	40

Testcase	**1**
Definition	The colour is none
Arguments	Color → RGB_LED_NONE_COLOR
Real Colour	Switched off
Expected Traces	1, 2, 3, 4, 5
Real Traces	1, 2, 3, 4, 5

Testcase	**2**
Definition	The colour is red
Arguments	Color → RGB_LED_RED_COLOR
Real Colour	Red
Expected Traces	1, 6, 7, 8, 9, 10
Real Traces	1, 6, 7, 8, 9, 10

Testcase	**3**
Definition	The colour is orange
Arguments	Color → RGB_LED_ORANGE_COLOR
Real Colour	Orange
Expected Traces	1, 6, 11, 12, 13, 14, 15
Real Traces	1, 6, 11, 12, 13, 14, 15

Testcase	**4**
Definition	The colour is yellow
Arguments	Color → RGB_LED_YELLOW_COLOR
Real Colour	Yellow
Expected Traces	1, 6, 11, 16, 17, 18, 19, 20
Real Traces	1, 6, 11, 16, 17, 18, 19, 20

Testcase	**5**
Definition	The colour is green
Arguments	Color → RGB_LED_GREEN_COLOR
Real Colour	Green
Expected Traces	1, 6, 11, 16, 21, 22, 23, 24, 25
Real Traces	1, 6, 11, 16, 21, 22, 23, 24, 25

Testcase	**6**
Definition	The colour is blue
Arguments	Color → RGB_LED_BLUE_COLOR
Real Colour	Blue
Expected Traces	1, 6, 11, 16, 21, 26, 27, 28, 29, 30
Real Traces	1, 6, 11, 16, 21, 26, 27, 28, 29, 30

Testcase	**7**
Definition	The colour is indigo
Arguments	Color → RGB_LED_INDIGO_COLOR
Real Colour	Indigo
Expected Traces	1, 6, 11, 16, 21, 26, 31, 32, 33, 34, 35
Real Traces	1, 6, 11, 16, 21, 26, 31, 32, 33, 34, 35

Testcase	**8**
Definition	Another colour
Arguments	Color → 7
Real Colour	Switched off
Expected Traces	1, 6, 11, 16, 21, 26, 31, 36, 37, 38, 39, 40
Real Traces	1, 6, 11, 16, 21, 26, 31, 36, 37, 38, 39, 40

Coverage	**10**
(%)	100

- ***MICROSDCARD:***
  → ***createDir***

Method	**createDir**
script	MicroSDCard_createDir.ino
Testbench	MicroSDCard_createDir.xlsx

Testcases	2
Statements	5

Testcase	**1**
Definition	The path already exists or has not been created of another reason
Arguments	path → /MyDirectory
Expected Outcome	MICROSD_CARD_CREATE_DIR_ERROR
Real Outcome	MICROSD_CARD_CREATE_DIR_ERROR
Expected Traces	1, 4, 5
Real Traces	1, 4, 5

Testcase	**2**
Definition	The directory has been successfully created
Arguments	path → /MyDirectory
Expected Outcome	MICROSD_CARD_CREATE_DIR_OK
Real Outcome	MICROSD_CARD_CREATE_DIR_OK
Expected Traces	1, 2, 3
Real Traces	1, 2, 3

Coverage	**10**
(%)	100

→ ***listDir***

Method	**createDir**
script	MicroSDCard_listDir.ino
Testbench	MicroSDCard_listDir.xlsx
Testcases	4
Statements	13

Testcase	**1**
Definition	The directory does not exist
Arguments	path → /MyDirectory
Expected References	nFiles → *Irrelevant*
Real References	nFiles → *Irrelevant*

Expected Outcome	MICROSD_CARD_LIST_DIR_ERROR_INVAL_PATH
Real Outcome	MICROSD_CARD_LIST_DIR_ERROR_INVAL_PATH
Expected Traces	1, 2, 3
Real Traces	1, 2, 3

Testcase	**2**
Definition	The directory is a file
Arguments	path → /MyDirectory
Expected References	nFiles → *Irrelevant*
Real References	nFiles → *Irrelevant*
Expected Outcome	MICROSD_CARD_LIST_DIR_ERROR_NOT_DIR
Real Outcome	MICROSD_CARD_LIST_DIR_ERROR_NOT_DIR
Expected Traces	1, 2, 4, 5
Real Traces	1, 2, 4, 5

Testcase	**3**
Definition	The directory exists and its empty
Arguments	path → /MyDirectory
Expected References	nFiles → 0
Real References	nFiles → 0
Expected Outcome	MICROSD_CARD_LIST_DIR_OK
Real Outcome	MICROSD_CARD_LIST_DIR_OK
Expected Traces	1, 2, 4, 6, 7, 8, 13
Real Traces	1, 2, 4, 6, 7, 8, 13

Testcase	**4**
Definition	The directory exists and its not empty
Arguments	path → /MyDirectory
Expected Reference	nFiles → 4
Real References	nFiles → 4
Expected Outcome	MICROSD_CARD_LIST_DIR_OK
Real Outcome	MICROSD_CARD_LIST_DIR_OK
Expected Traces	1, 2, 4, 6, 7, 8, loop (9, 10, 11, 12), 13
Real Traces	1, 2, 4, 6, 7, 8, loop (9, 10, 11, 12), 13

Coverage	**10**
(%)	100

➔ ***deleteDir***

Method	**deleteDir**
script	MicroSDCard_deleteDir.ino
Testbench	MicroSDCard_deleteDir.xlsx
Testcases	3
Statements	4

Testcase	**1**
Definition	The directory does not exist
Arguments	path → /MyDirectory
Expected Outcome	MICROSD_CARD_DELETE_DIR_ERROR
Real Outcome	MICROSD_CARD_DELETE_DIR_ERROR
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Testcase	**2**
Definition	The directory exists but its not empy
Arguments	path → /MyDirectory
Expected Outcome	MICROSD_CARD_DELETE_DIR_ERROR
Real Outcome	MICROSD_CARD_DELETE_DIR_ERROR
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Testcase	**3**
Definition	The directory exists and it is empty
Arguments	path → /MyDirectory
Expected Outcome	MICROSD_CARD_DELETE_DIR_OK
Real Outcome	MICROSD_CARD_DELETE_DIR_OK
Expected Traces	1, 2
Real Traces	1, 2

Coverage	**4**
(%)	100

➔ ***appendFile***

Method	**appendFile**
script	MicroSDCard_appendFile.ino
Testbench	MicroSDCard_appendFile.xlsx
Testcases	4
Statements	8

Testcase	**1**
Definition	The buffer is too big to append
Arguments	path → /MyDirectory/myText buff → TextTestCase1.txt
Expected Outcome	MICROSD_CARD_APPEND_FILE_ERROR_BUF_OVFL
Real Outcome	MICROSD_CARD_APPEND_FILE_ERROR_BUF_OVFL
Expected Traces	1, 2
Real Traces	1, 2

Testcase	**2**
Definition	The path is not a valid path
Arguments	path → /MyDirectory/myText buff → *Irrelevant*
Expected Outcome	MICROSD_CARD_APPEND_FILE_ERROR_INVAL_PATH
Real Outcome	MICROSD_CARD_APPEND_FILE_ERROR_INVAL_PATH
Expected Traces	1, 3, 4, 6, 7, 8
Real Traces	1, 3, 4, 6, 7, 8

Testcase	**3**
Definition	The file does not exist
Arguments	path → /MyDirectory/myText buff → TextTestCase2.txt
Expected Outcome	MICROSD_CARD_APPEND_FILE_OK
Real Outcome	MICROSD_CARD_APPEND_FILE_OK
Expected Traces	1, 3, 4, 5
Real Traces	1, 3, 4, 5

Testcase	**4**

Definition	The file exists
Arguments	path → /MyDirectory/myText buff → A new message ready for append
Expected Outcome	MICROSD_CARD_DELETE_DIR_OK
Real Outcome	MICROSD_CARD_DELETE_DIR_OK
Expected Traces	1, 3, 4, 5
Real Traces	1, 3, 4, 5

Coverage	**5**
(%)	100

→ ***readFile***

• **Method**	**readFile**
script	MicroSDCard_readFile.ino
Testbench	MicroSDCard_readFile.xlsx
Testcases	4
Statements	26

Testcase	**1**
Definition	The file does not exist
Arguments	path → /MyDirectory/myText.txt (*strlen =3653)* lseek → *Irrelevant*
Expected Outcome	MICROSD_CARD_READ_FILE_ERROR_NOT_FOUND_PATH
Real Outcome	MICROSD_CARD_READ_FILE_ERROR_NOT_FOUND_PATH
Expected Traces	1, 24, 25, 26
Real Traces	1, 24, 25, 26

Testcase	**2**
Definition	The file pointer is beyond the size of the file
Arguments	path → /MyDirectory/myText.txt (*strlen =3653)* lseek → MAXIMUM_BUFFER_LENGTH (4096)
Expected Outcome	MICROSD_CARD_APPEND_FILE_ERROR_LSEEK_OV
Real Outcome	MICROSD_CARD_APPEND_FILE_ERROR_LSEEK_OV
Expected Traces	1, 2, 3, 4, 20, 21, 22, 23
Real Traces	1, 2, 3, 4, 20, 21, 22, 23

Testcase	**3**
Definition	The buffer fills before the end of the file
Arguments	path → /MyDirectory/myText.txt lseek → 0
Expected Outcome	MICROSD_CARD_READ_FILE_OK_NOT_END
Real Outcome	MICROSD_CARD_READ_FILE_OK_NOT_END
Expected Traces	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Real Traces	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Testcase	**4**
Definition	The file is completed before the buffer is full
Arguments	path → /MyDirectory/myText.txt lseek → HUGE_BUFFER_LENGTH (2048)
Expected Outcome	MICROSD_CARD_READ_FILE_OK_END
Real Outcome	MICROSD_CARD_READ_FILE_OK_END
Expected Traces	1, 2, 3, 4, 5, 6, 12, 13, 14, 15, 16, 17, 18, 19
Real Traces	1, 2, 3, 4, 5, 6, 12, 13, 14, 15, 16, 17, 18, 19

Coverage	**5**
(%)	100

→ ***renameFile***

• **Method**	**renameFile**
script	MicroSDCard_renameFile.ino
Testbench	MicroSDCard_renameFile.xlsx
Testcases	3
Statements	4

Testcase	**1**
Definition	The old file does not exist
Arguments	oldName → /myFile.txt newName → /myNewFile.txt
Expected Outcome	MICROSD_CARD_RENAME_FILE_ERROR

Real Outcome	MICROSD_CARD_RENAME_FILE_ERROR
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Testcase	**2**
Definition	The old file does exists and the new one does not
Arguments	oldName → /myFile.txt newName → /myNewFile.txt
Expected Outcome	MICROSD_CARD_RENAME_FILE_OK
Real Outcome	MICROSD_CARD_RENAME_FILE_OK
Expected Traces	1, 2
Real Traces	1, 2

Testcase	**3**
Definition	Both old and new file exists
Arguments	oldName → /myFile.txt newName → /myNewFile.txt
Expected Outcome	MICROSD_CARD_RENAME_FILE_ERROR
Real Outcome	MICROSD_CARD_RENAME_FILE_ERROR
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Coverage	**4**
(%)	100

➔ ***deleteFile***

• **Method**	**deleteFile**
script	MicroSDCard_deleteFile.ino
Testbench	MicroSDCard_deleteFile.xlsx
Testcases	2
Statements	4

Testcase	**1**
Definition	The file to delete does not exist
Arguments	Path → /myFile.txt
Expected Outcome	MICROSD_CARD_DELETE_FILE_ERROR

Real Outcome	MICROSD_CARD_DELETE_FILE_ERROR
Expected Traces	1, 3, 4
Real Traces	1, 3, 4

Testcase	**2**
Definition	The file to delete does exist
Arguments	Path → /myFile.txt
Expected Outcome	MICROSD_CARD_DELETE_FILE_OK
Real Outcome	MICROSD_CARD_DELETE_FILE_OK
Expected Traces	1, 2
Real Traces	1, 2

Coverage	**4**
(%)	100

→ getDatetime

Method	**getDatetime**
script	Button_getDatetime.ino
Testbench	Button_getDatetime.xlsx
Testcases	7
Statements	22

Testcase	**1**
Definition	The chosen variable is year
Arguments	var → VAR_YEAR (1)
Date and Time	2022/09/02--03:09:44
Expected Outcome	2022
Real Outcome	2022
Expected Traces	1, 2, 3, 4
Real Traces	1, 2, 3, 4

Testcase	**2**
Definition	The chosen variable is month
Arguments	var → VAR_MONTH (2)

Date and Time	2022/09/02--03:19:50
Expected Outcome	9
Real Outcome	9
Expected Traces	1, 2, 5, 6, 7
Real Traces	1, 2, 5, 6, 7

Testcase	**3**
Definition	The chosen variable is day
Arguments	var → VAR_DAY (3)
Date and Time	2022/09/02--03:22:29
Expected Outcome	2
Real Outcome	2
Expected Traces	1, 2, 5, 8, 9, 10
Real Traces	1, 2, 5, 8, 9, 10

Testcase	**4**
Definition	The chosen variable is hour
Arguments	var → VAR_HOUR (4)
Date and Time	2022/09/02--03:25:12
Expected Outcome	3
Real Outcome	3
Expected Traces	1, 2, 5, 8, 11, 12, 13
Real Traces	1, 2, 5, 8, 11, 12, 13

Testcase	**5**
Definition	The chosen variable is minute
Arguments	var → VAR_MINUTE (5)
Date and Time	2022/09/02--03:28:09
Expected Outcome	25
Real Outcome	25
Expected Traces	1, 2, 5, 8, 11, 12, 14, 15, 16
Real Traces	1, 2, 5, 8, 11, 12, 14, 15, 16

Testcase	**6**
Definition	The chosen variable is second
Arguments	var → VAR_SECOND (6)

Date and Time	2022/09/02--03:31:19
Expected Outcome	19
Real Outcome	19
Expected Traces	1, 2, 5, 8, 11, 12, 14, 17, 18, 19
Real Traces	1, 2, 5, 8, 11, 12, 14, 17, 18, 19

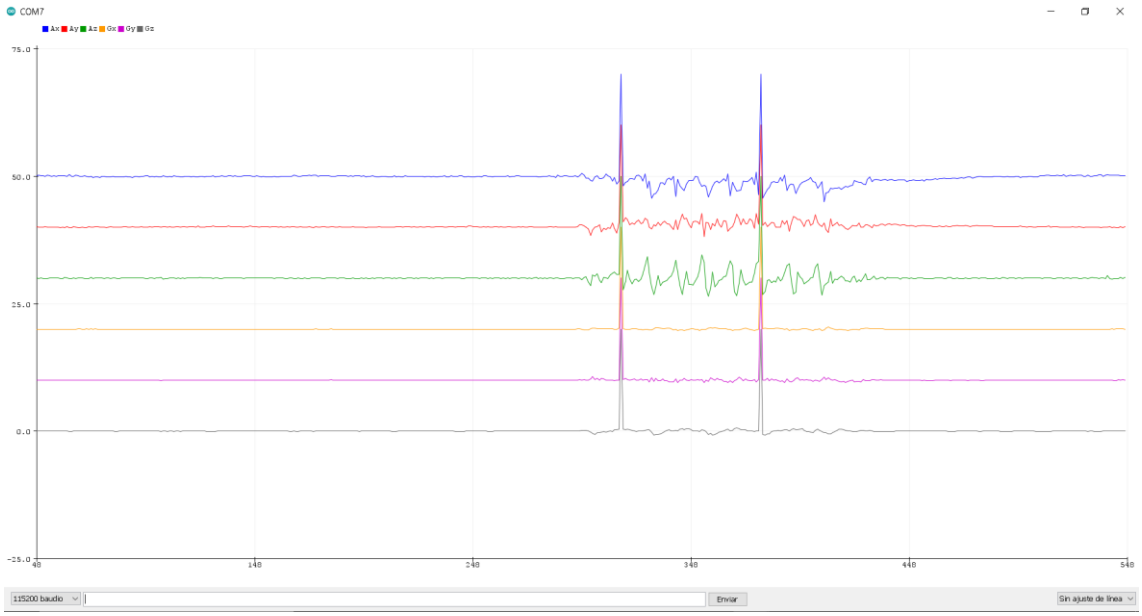
Testcase	**7**
Definition	The chosen variable other different
Arguments	var → 7
Date and Time	2022/09/02--03:31:19
Expected Outcome	19
Real Outcome	19
Expected Traces	1, 20, 21, 22
Real Traces	1, 20, 21, 22

Coverage	**10**
(%)	100

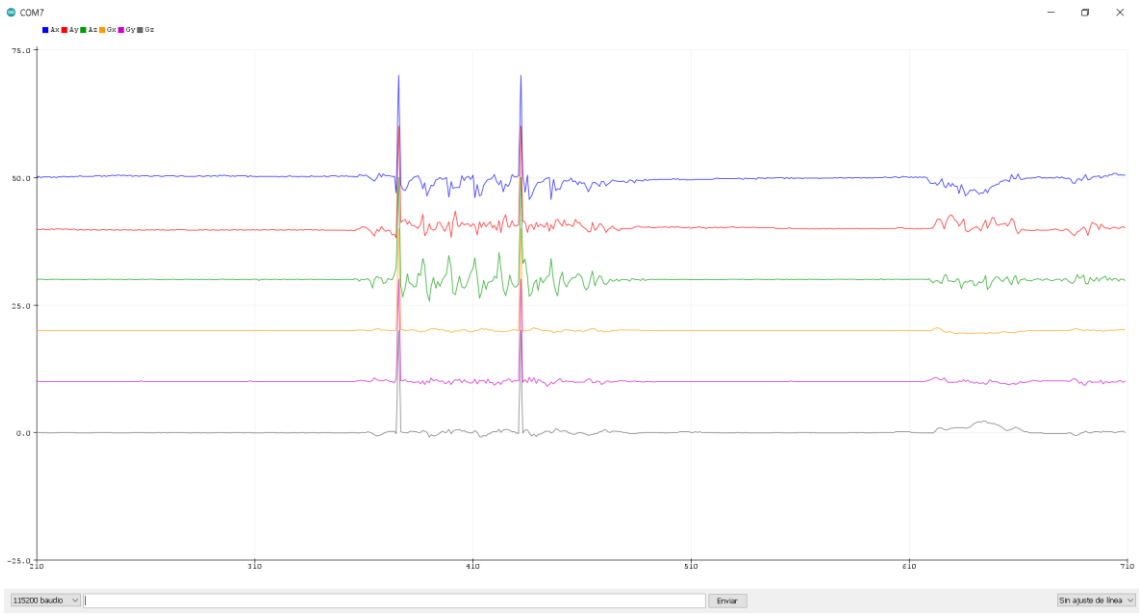
B Plotter

- **Patient 63:**

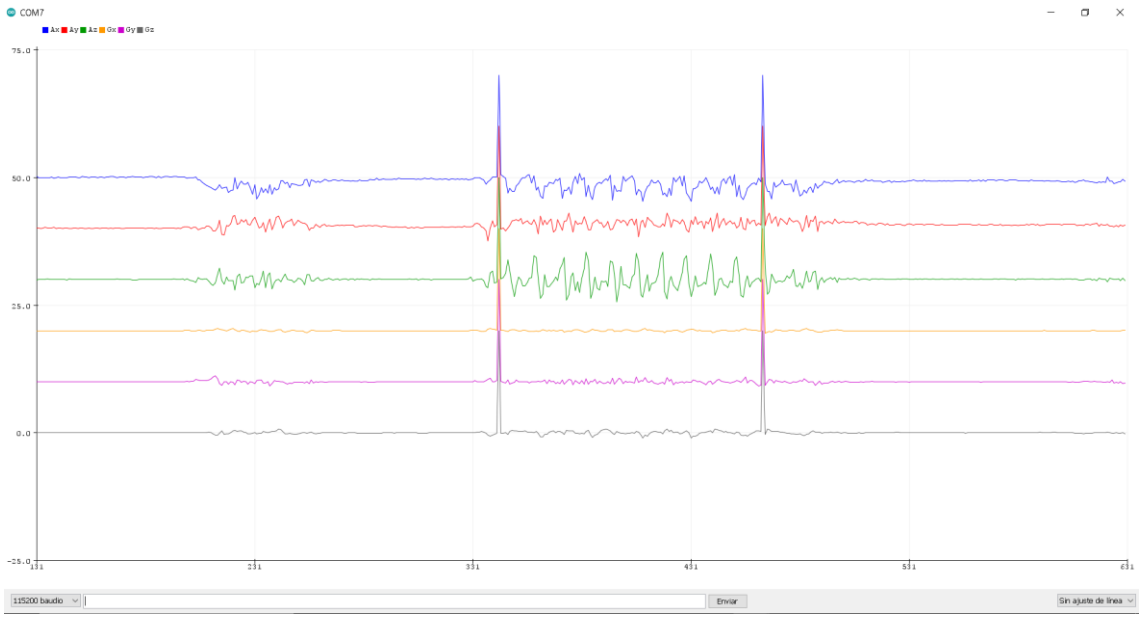
➔ 2M4 1st Round



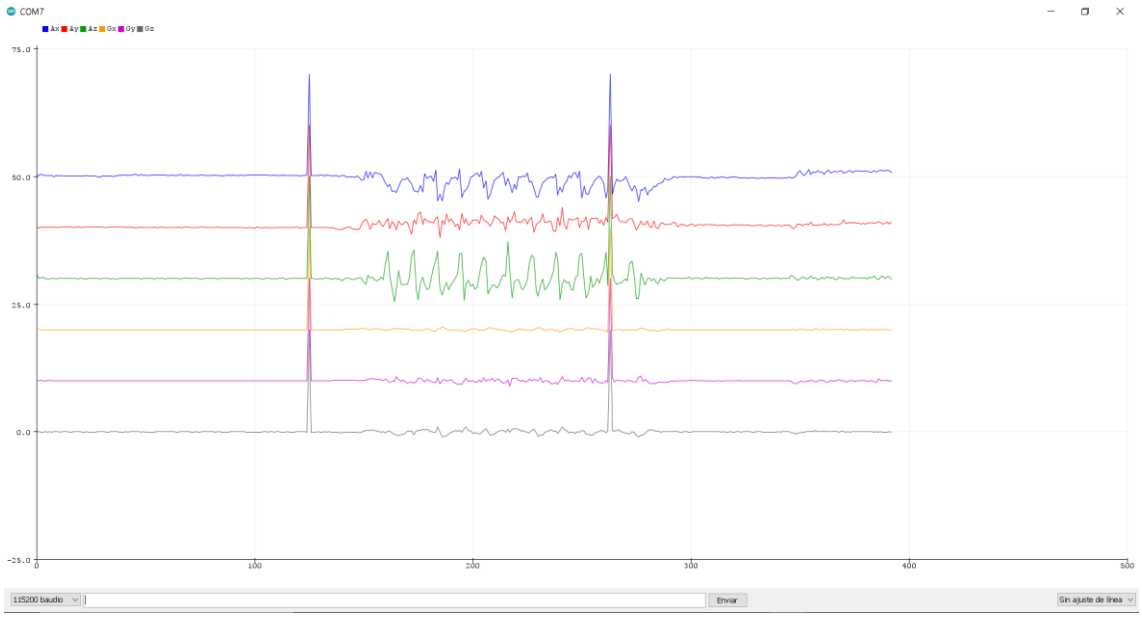
➔ 2M4 2nd Round:



➔ 6M 1st Round

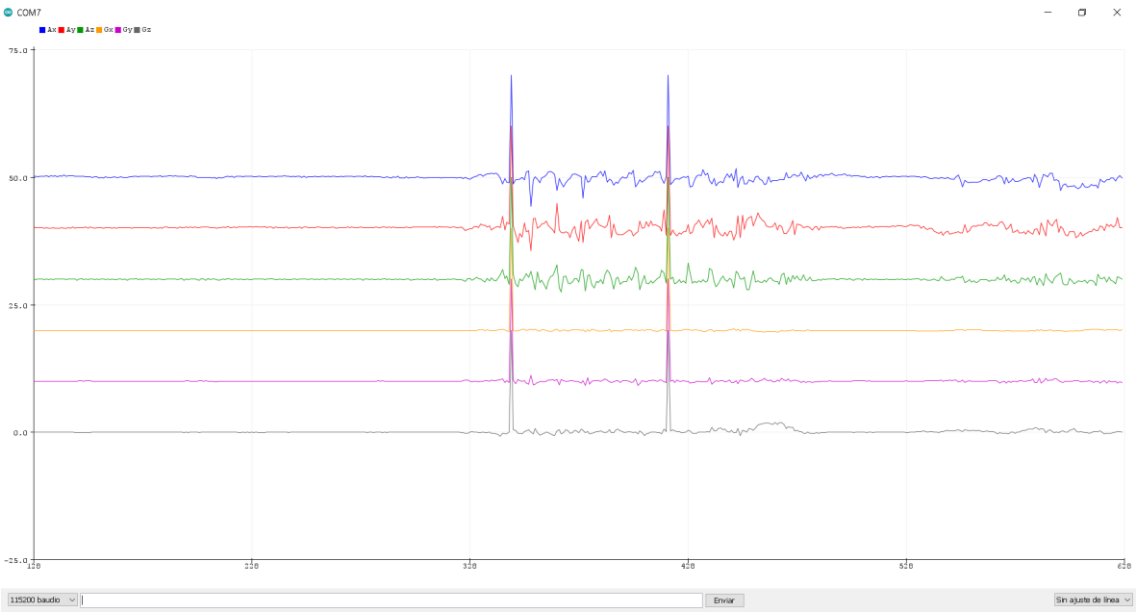


➔ 6M 2nd Round

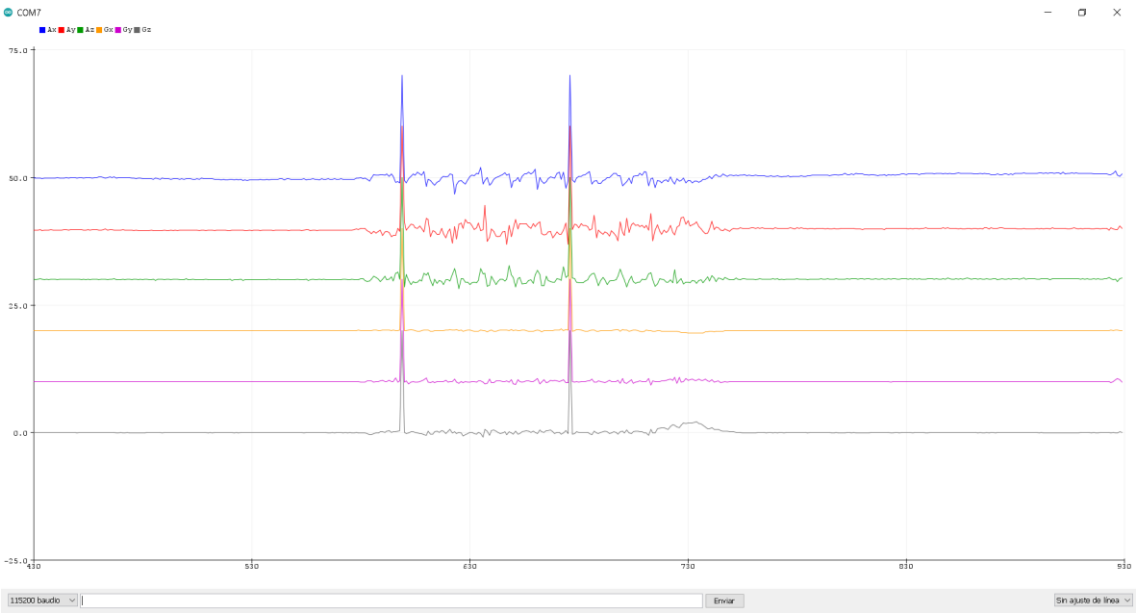


- **Patient 140:**

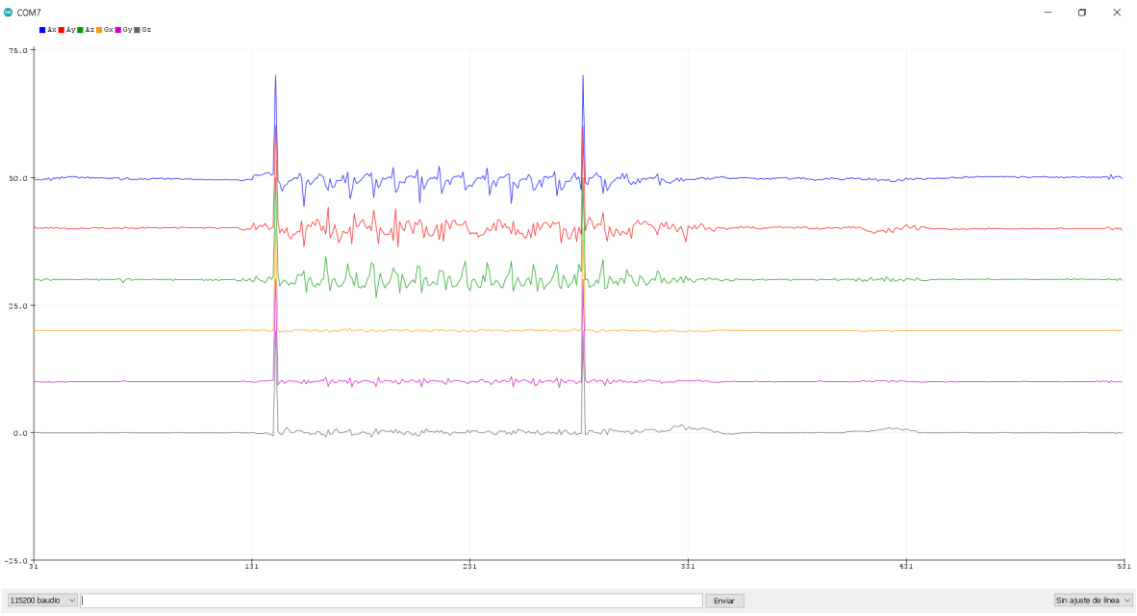
➔ 2M4 1st Round



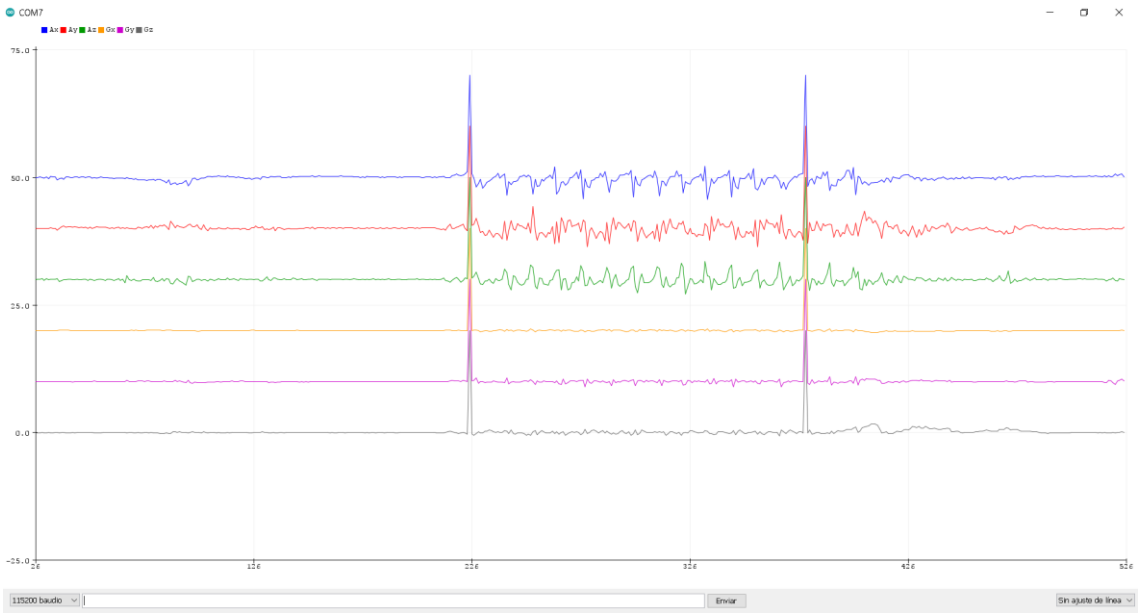
➔ 2M4 2nd Round:



➔ 6M 1st Round

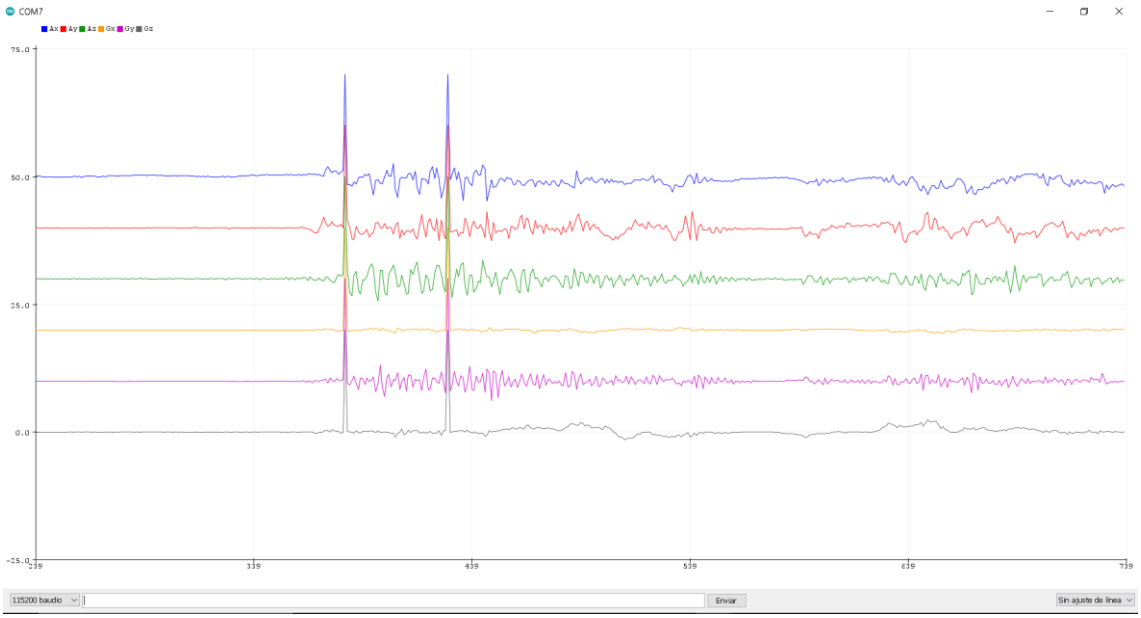


➔ 6M 2nd Round

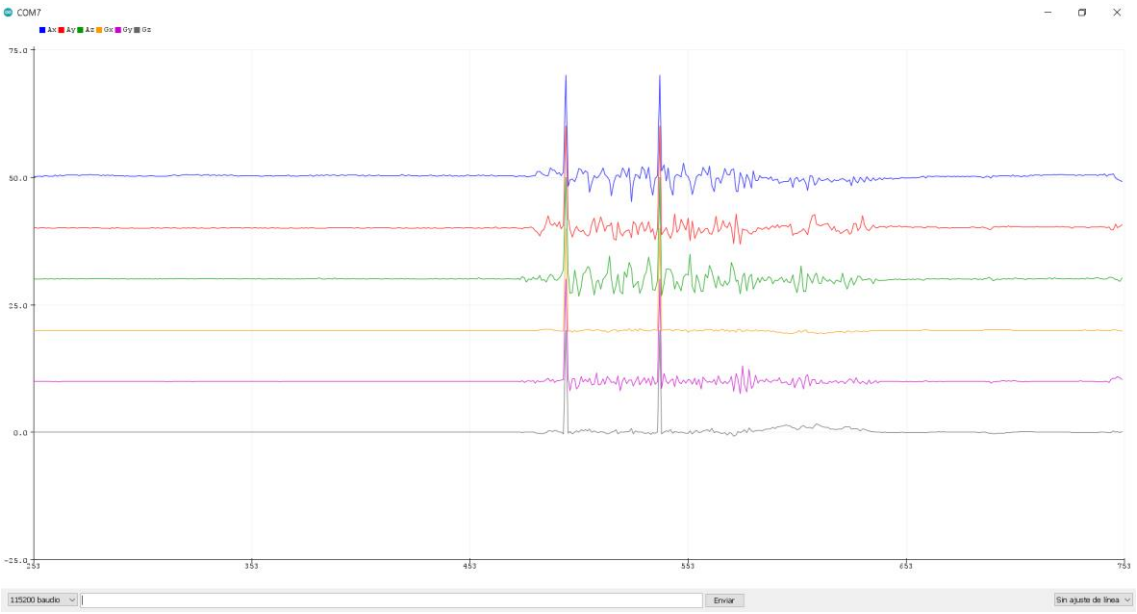


- **Patient 215:**

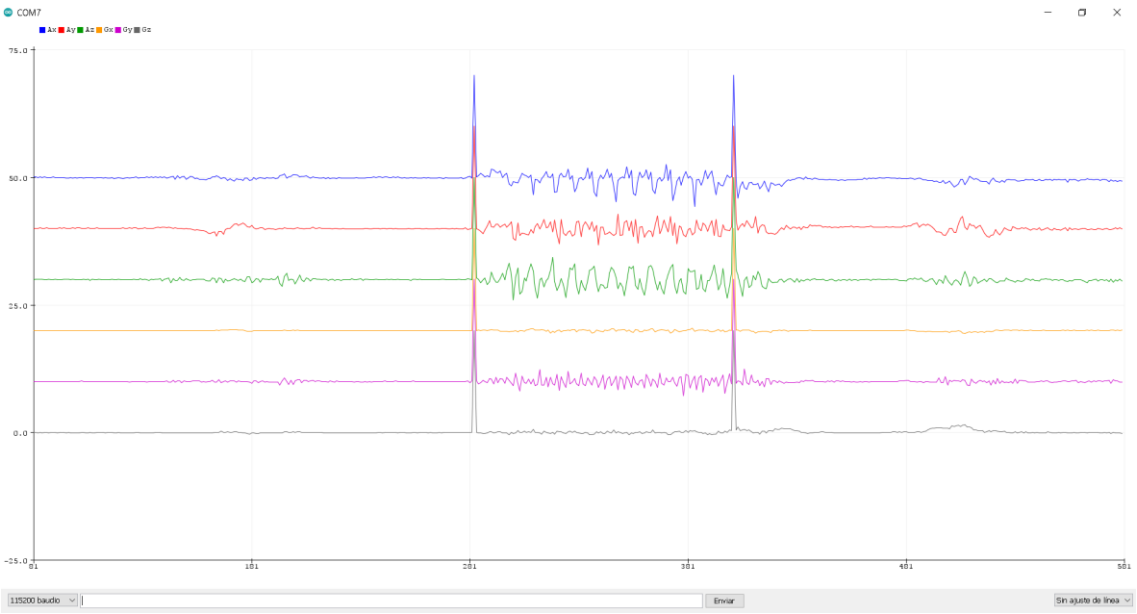
➔ 2M4 1st Round



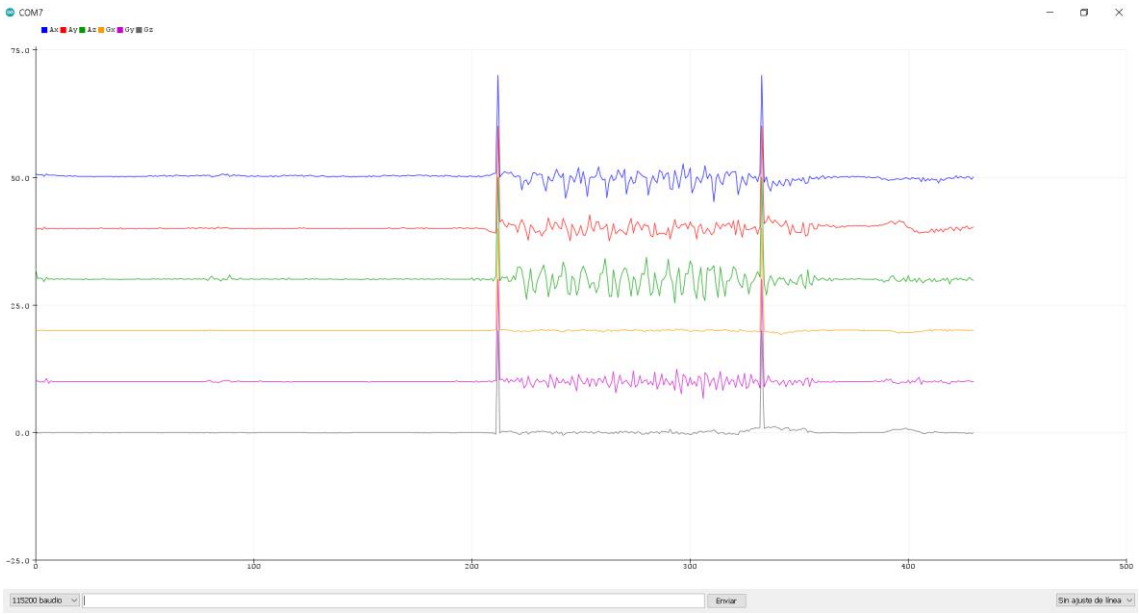
➔ 2M4 2nd Round:



➔ 6M 1st Round

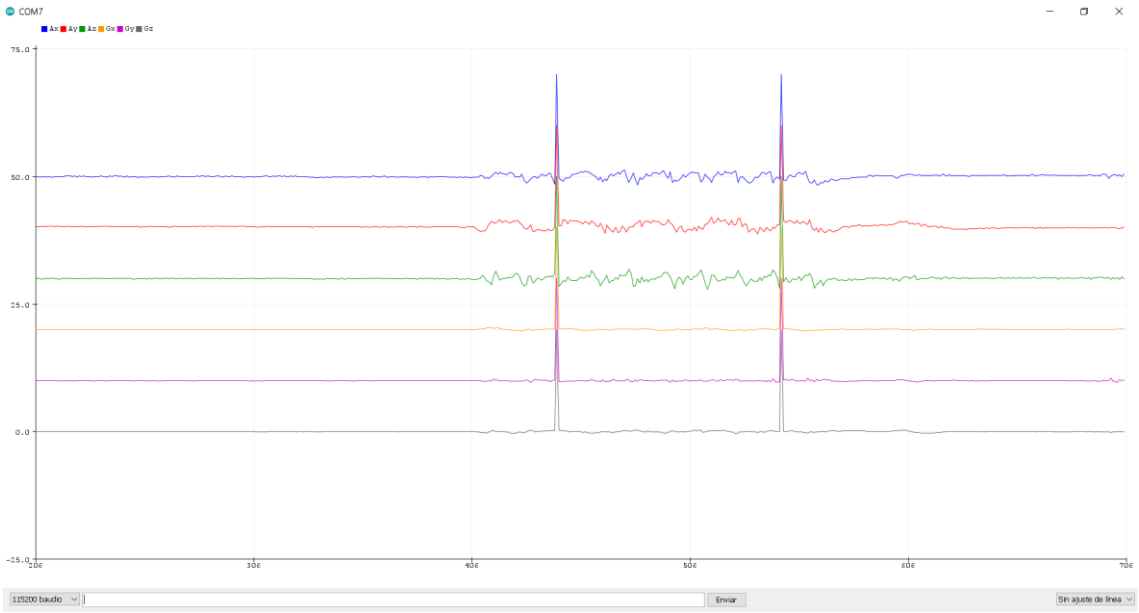


➔ 6M 2nd Round

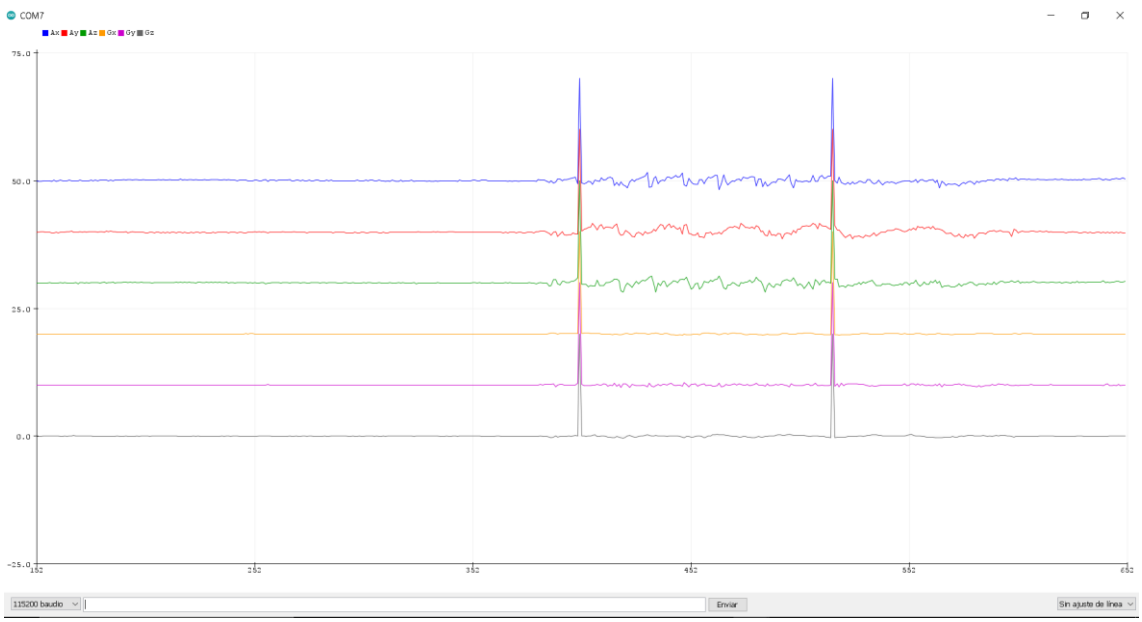


- **Patient 216:**

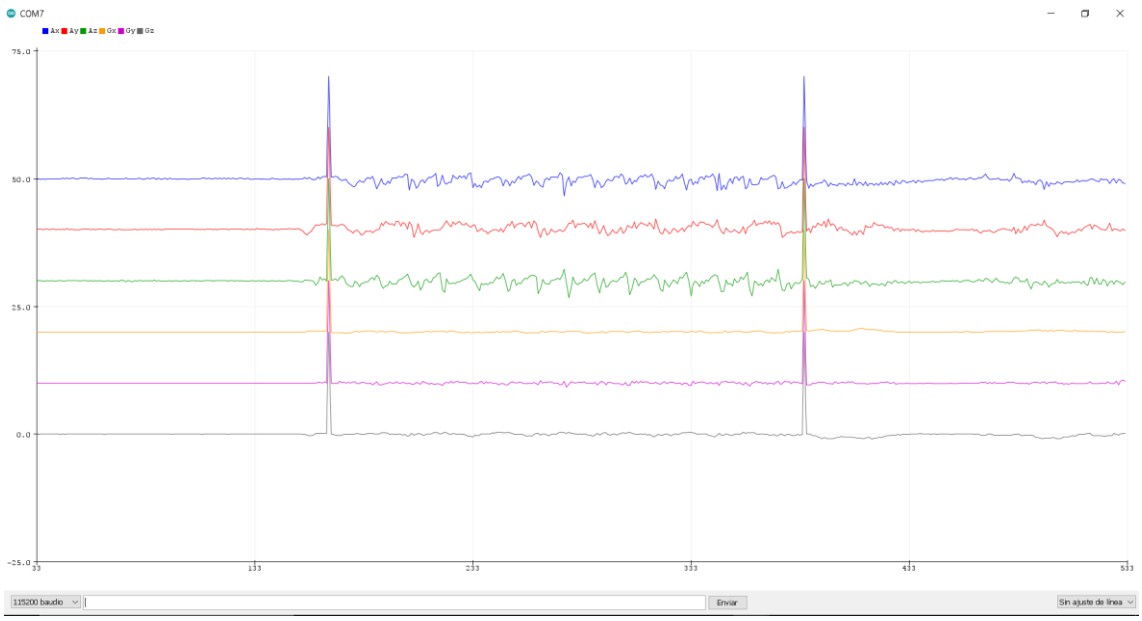
➔ 2M4 1st Round



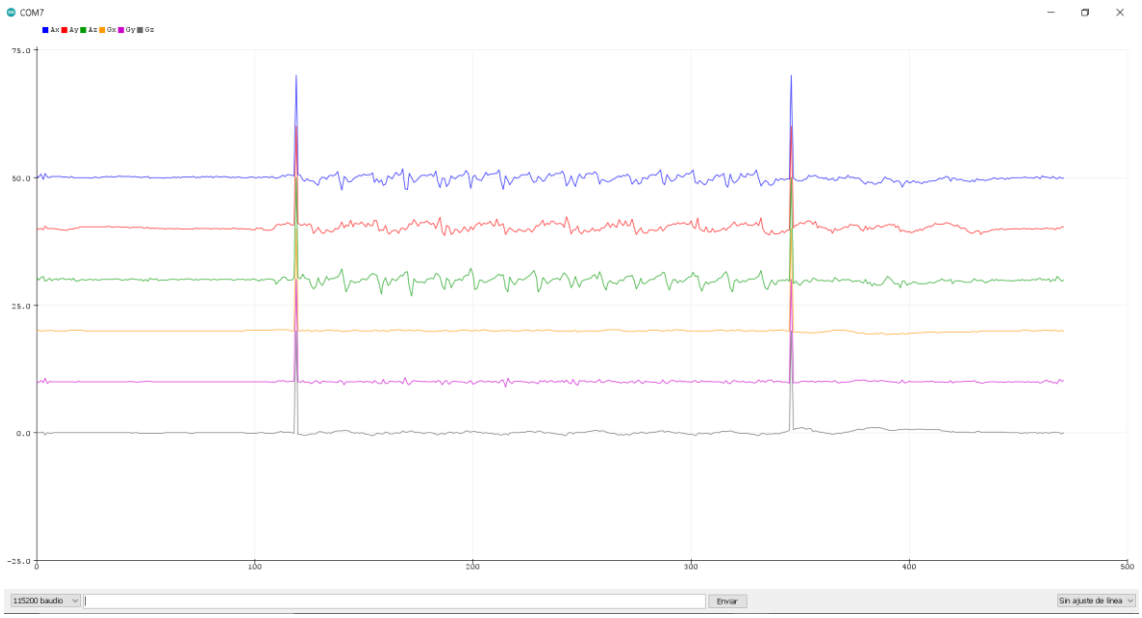
➔ 2M4 2nd Round:



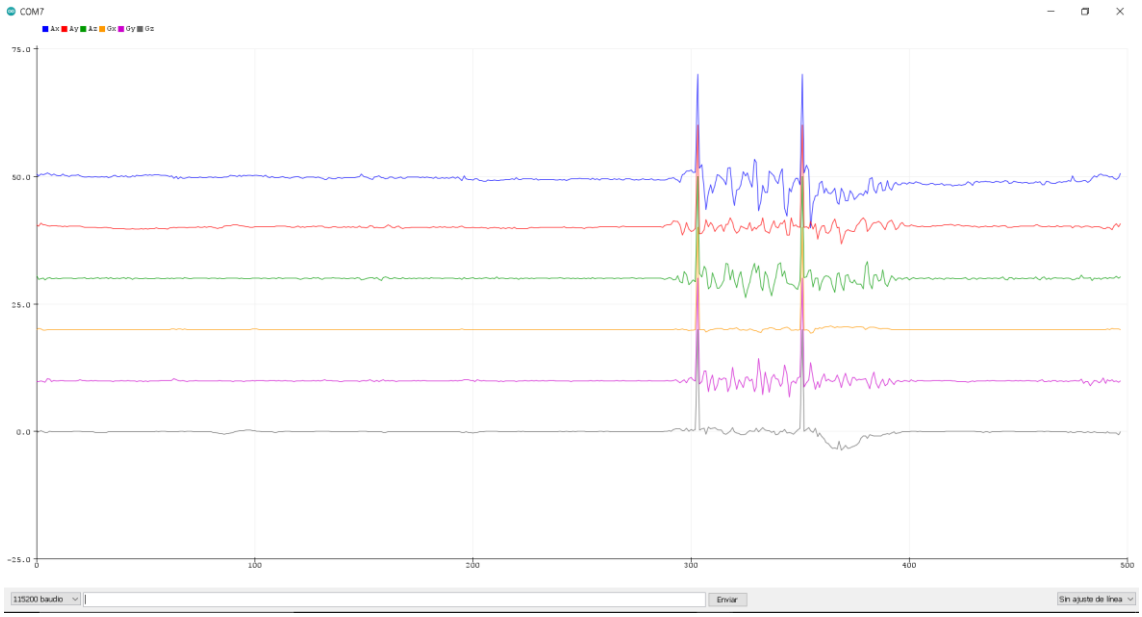
➔ 6M 1st Round



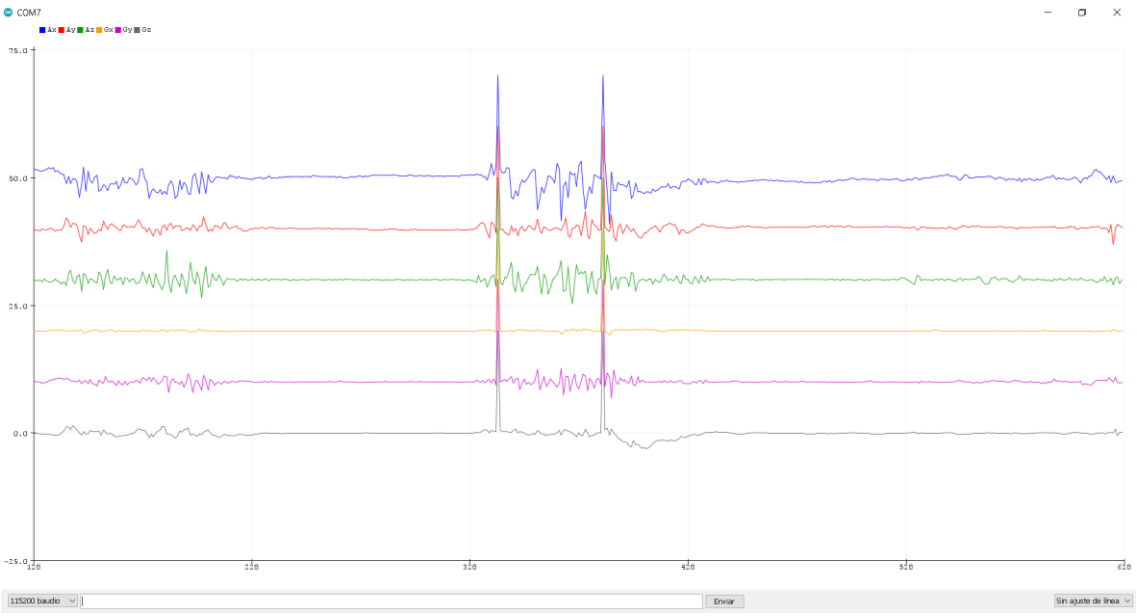
➔ 6M 2nd Round



2M4 1 Time

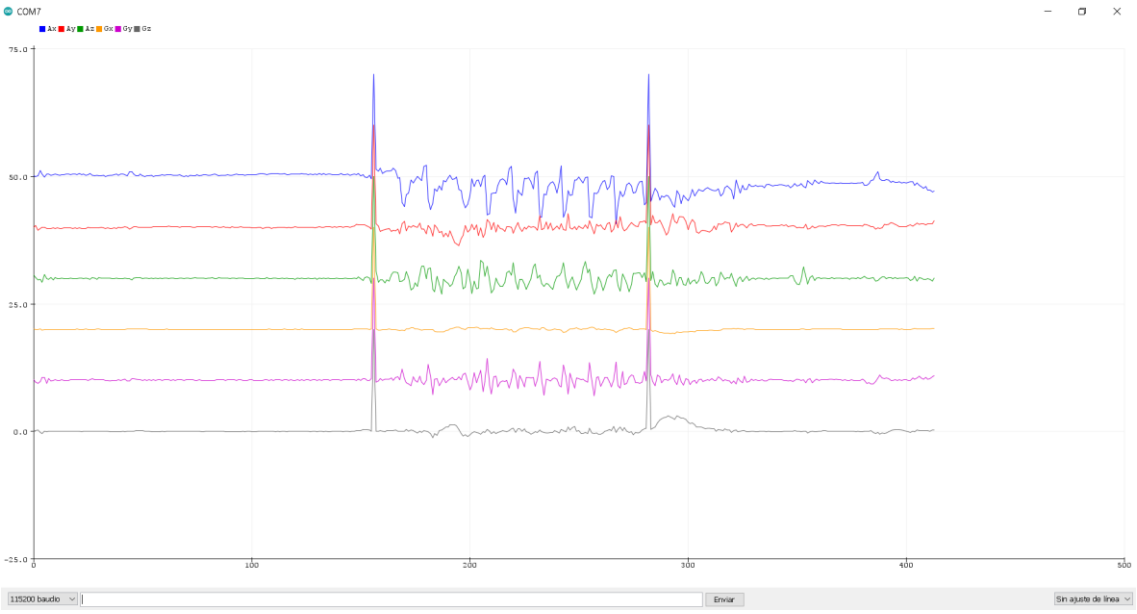


2M4 2 Time

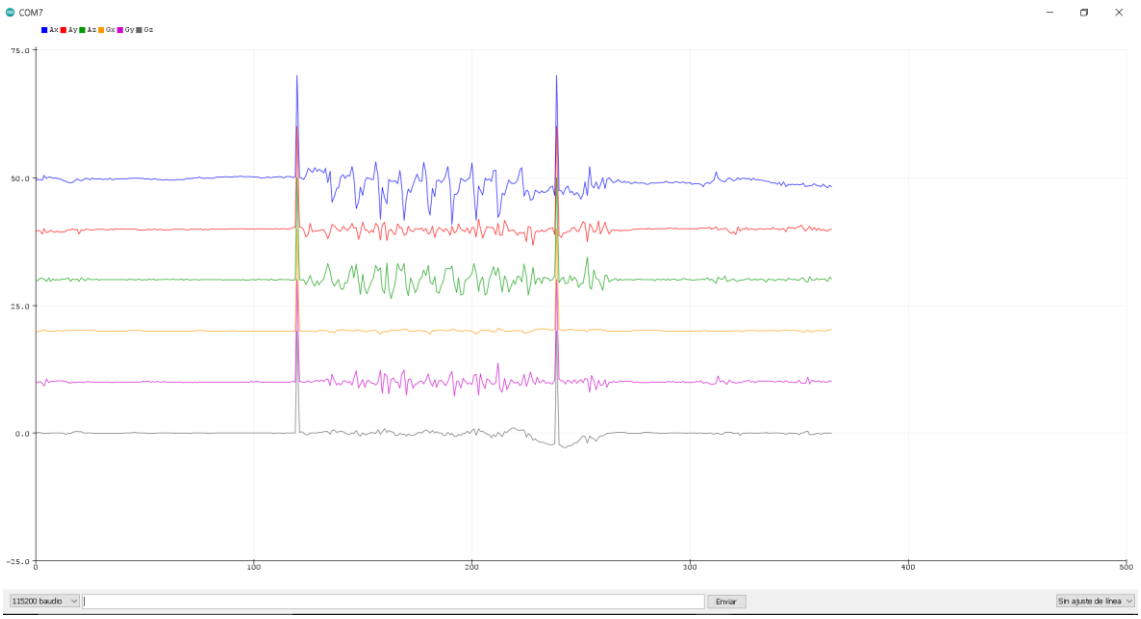


6M 1 Time

477



6M 2 Time


COM7
Ax Ay Az Gx Gy Gz
75.0
50.0
25.0
0.0
-25.0
0
100
200
300
400
500
115200 baudio
Enviar
Sin ajuste de línea

C Usability questionaires

- **Patient 110:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**				X	
2.	**El sistema me resultó innecesariamente complejo.**			X		
3.	**Creo que el sistema es bastante fácil de utilizar.**				X	
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**			X		
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**			X		
6.	**Opino que hubo demasiada inconsistencia en el sistema.**			X		
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**				X	
8.	**Me sentí algo incómodo al utilizar este sistema.**			X		
9.	**Me sentí muy seguro al utilizar este sistema.**				X	
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**				X	

- **Patient 111:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**			X		
2.	**El sistema me resultó innecesariamente complejo.**		X			
3.	**Creo que el sistema es bastante fácil de utilizar.**				X	
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**					X
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**					X
6.	**Opino que hubo demasiada inconsistencia en el sistema.**				X	
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**				X	
8.	**Me sentí algo incómodo al utilizar este sistema.**	X				
9.	**Me sentí muy seguro al utilizar este sistema.**				X	
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**					X

- **Patient 113:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**					X
2.	**El sistema me resultó innecesariamente complejo.**		X			
3.	**Creo que el sistema es bastante fácil de utilizar.**					X
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**	X				
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**					X
6.	**Opino que hubo demasiada inconsistencia en el sistema.**		X			
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**				X	
8.	**Me sentí algo incómodo al utilizar este sistema.**	X				
9.	**Me sentí muy seguro al utilizar este sistema.**					X
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**	X				

- **Patient 115:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**				X	
2.	**El sistema me resultó innecesariamente complejo.**				X	
3.	**Creo que el sistema es bastante fácil de utilizar.**			X		
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**				X	
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**		X			
6.	**Opino que hubo demasiada inconsistencia en el sistema.**			X		
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**				X	
8.	**Me sentí algo incómodo al utilizar este sistema.**		X			
9.	**Me sentí muy seguro al utilizar este sistema.**			X		
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**					X

- **Patient 117:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**		X			
2.	**El sistema me resultó innecesariamente complejo.**		X			
3.	**Creo que el sistema es bastante fácil de utilizar.**				X	
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**	X				
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**			X		
6.	**Opino que hubo demasiada inconsistencia en el sistema.**			X		
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**				X	
8.	**Me sentí algo incómodo al utilizar este sistema.**					X
9.	**Me sentí muy seguro al utilizar este sistema.**		X			
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**				X	

- **Patient 123:**

	Pregunta	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
1.	**Creo que me gustaría usar este sistema frecuentemente.**				X	
2.	**El sistema me resultó innecesariamente complejo.**				X	
3.	**Creo que el sistema es bastante fácil de utilizar.**		X			
4.	**Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.**				X	
5.	**Creo que las diferentes funciones del sistema se encuentran muy bien integradas.**					X
6.	**Opino que hubo demasiada inconsistencia en el sistema.**		X			
7.	**Imagino que la mayoría de las personas aprenderá a utilizar el sistema rápidamente.**					X
8.	**Me sentí algo incómodo al utilizar este sistema.**		X			
9.	**Me sentí muy seguro al utilizar este sistema.**				X	
10.	**Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.**				X	