INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Brain-Computer Interfaces for mobility aids

JOÃO TOMÁS BARROS BARREIRA
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Brain-Computer Interfaces for mobility aids

João Tomás Barros Barreira

Dissertation to obtain the Master's Degree in

Informatics Engineering,

Area of Specialization in

Graphics and Multimedia Systems

Supervisor: Doutor António Abel Vieira de Castro, DEI/ISEP

Evaluation Committee:

President:

[Nome do Presidente, Categoria, Escola]

Members:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, October 2022

Acknowledgments

Naturally, I had to include in this document everyone who assisted me in completing this dissertation:

Prof. António Vieira Castro, for his availability, dedication, and commitment to the completion of this dissertation.

To my family, who have accepted and understood my absences during this time, as well as my best friends Sofia Canelas, Miguel Sousa, and Bruno Carvalho, I thank you all for your support, which has always accompanied me throughout this master's journey and pushed me to keep going until the very end.

Thank you to the research group SIIS (Interaction Systems and Social Innovation) for including me as a collaborator and to ISEP and its teachers who were instrumental in my educational development.

Abstract

At the time of writing, brain-computer interfaces are gaining popularity not only in the field in which they began but also among the general public. It is a good thing that this subject is becoming more recognized because it means that further research on it will not only benefit the average consumer but will also improve the way of life of people who really need it.

The current work intends to introduce the reader to the world of Brain-Computer Interfaces, explaining their inner workings and identifying a suitable one designed for everyday use, intended for people suffering from a mobility disability.

This Brain-Computer Interface would be integrated with the software being developed in the course of writing this work, trying to introduce a more independent lifestyle to anyone suffering from a mobility impairment, helping the user to move around more freely, for example, in an electric wheelchair.

It is expected that the user would not fully excel at first in moving freely, but it would be expected that, with time, the user would get accustomed to the device and software and be able to move around at will with ease.

During the testing phase, this prototype was tested by a group of randomly chosen people from all genders with a wide range of ages to evaluate if the developed solution was viable for a second phase of testing, to be performed along with future work.

Finally, to conclude this work some conclusions will be compiled and future work pointed out to inform which steps will be taken in order to improve the developed prototype.

Keywords: Brain-Computer Interfaces, Mobility aids, Emotiv Insight, Signal Processing

Resumo

No momento da escrita, as Interfaces Cérebro-Computador (ICC) estão a aumentar de popularidade, não só no campo em que começaram, mas também entre o público em geral. É bom que este tema esteja agora a ser mais reconhecido, pois significa que, com mais investigação sobre o assunto, não só o consumidor comum irá aproveitar este feito, como também irá melhorar o modo de vida das pessoas que realmente precisam dela.

O trabalho atual pretende apresentar o leitor ao mundo das Interfaces Cérebro-Computador, explicando o seu funcionamento e identificando um adequado ICC, projetado para uso diário, destinado a pessoas que sofrem de uma deficiência de mobilidade.

Esta Interface Cérebro-Computador seria integrada com o software que será desenvolvido no decorrer da escrita deste artigo, tentando introduzir um estilo de vida mais independente para quem sofre de uma deficiência de mobilidade, ajudando o utilizador a deslocar-se mais livremente, por exemplo, numa cadeira de rodas elétrica.

É de esperar que o utilizador não se acostume totalmente no início a mover-se livremente, mas seria de esperar que, com o tempo, o utilizador se habituasse ao dispositivo e ao software e fosse capaz de se deslocar à vontade com facilidade.

Durante a fase de testes, este protótipo foi testado por um grupo de pessoas escolhidas aleatoriamente de todos os sexos com uma ampla faixa de idades para avaliar se a solução desenvolvida era viável para uma segunda fase de testes, a ser realizada juntamente com o trabalho futuro.

Por fim, para concluir este trabalho algumas conclusões serão compiladas e trabalho futuro será delegado para informar quais os passos que serão dados para melhorar o protótipo desenvolvido.

Palavras-chave: Brain-Computer Interfaces, Ajudas à mobilidade, Emotiv Insight, Processamento de sinais

Contents

List of Figures

List of Tables

List of Code Snippets

Acronyms and Symbols

List of Acronyms

API	Application Programming Interface
BCI	Brain-Computer Interface
ECoG	Electrocorticography
EEG	Electroencephalogram
iEEG	intracranial Electroencephalography
LFP	Local Field Potentials
LIS	Locked-in Syndrome
MEA	Multi-Electrode Array
MEG	Magnetoencephalogram
RC car	Remote-controlled car
SU	Single-Unit
UI	User Interface

1 Introduction

"I regard the brain as a computer which will stop working when its components fail.

There is no heaven or afterlife for broken down computers;

that is a fairy story for people afraid of the dark."

Stephen Hawking

This work, written within the scope of obtaining the master's degree in Informatics Engineering, at the Instituto Superior de Engenharia do Porto, will engage in the **contextualization** and **analysis of Brain-Computer Interfaces** and what they require for them to function. It will also engage in the **design**, **development**, and **trials** of a possible solution to a problem that will be identified while this work continues.

In this chapter, starting with the contextualization, we provide clarity on the subject of Brain-Computer Interfaces, engaging the reader on current events and research recently presented to the public. Following with the recognition of the problem, it is recognized the target audience and provided an initial overview of the solution. Finally, a final overview of this work is provided with the identification of the objectives for this work, motivation and document structure.

1.1 Contextualization

According to the World Health Organization (WHO), the number of people dependent or in need of prostheses and/or other motor aids is around 30 million. (Anon., 2018)

It turns out that in many of these cases, the only way to communicate is through the brain, and it is necessary to work this area of interaction between it and the devices in question, prostheses, through Brain-Computer Interfaces.

"The research community has initially developed BCIs with biomedical applications in mind, leading to the generation of assistive devices." (Abdulkader, et al., 2014) Nowadays, at the time of writing, although its main purpose is still aimed at medical research and application, in recent years, its usage has become more common in other fields. For example, the most prominent

case where Brain-Computer Interfaces are being applied in non-medical fields is with Elon Musk's Neuralink project (Neuralink, 2021).

Neuralink is a neural implant that will let the user control a computer or a smart mobile device once they are connected. According to the official website, this device will allow the user to control the keyboard and mouse "directly with the activity of your brain" (Neuralink, 2021).

For the usage of Brain-Computer Interfaces in the medical field, their most relevant use, beyond mind-controlled prosthetics, is by patients with a rare neurological disorder called Locked-In Syndrome. This subject will be explored further in this article.

1.2 Problem

In a human body who is not suffering from a mobility disability, the brain communicates with the rest of body by sending and receiving information from the senses. Like a computer, it works with small bursts of electricity in order to send the required information around it, but for someone who suffers from this condition or any other that leads to it, it means that either the limb isn't able to receive that information or it's not there, either way, the brain loses the information that was sent.

Considering the scenario that someone could either lose their mobility by suffering from a chronic illness or by being involved in an accident, an improvement in the way of life by using BCIs could help many to reach their full potential again.

1.3 Objectives

Taking into consideration the previous contextualization, for our objectives, it is intended to carry out an investigation, consolidated with the development of a software prototype that serves as an interface (Brain-Computer Interface) between a user's brain and a device that he is dependent on. In this case, this device would be an electric wheelchair, but as a proof of concept the developed solution will be implemented in a small Remote-controlled car.

To achieve this main objective, it will be necessary to provide some analysis on the technologies that best fit the development of the intended interface, as well as an investigation followed by the choice of a brainwave capture hardware necessary for the communication process.

Succeeding this analysis phase comes the prototyping phase where it is required to mainly identify the use case and design various solutions, as well as evaluate them in order to consolidate most, if not all, positive aspects in one final prototype. Consecutively, we will proceed with the developmental phase following with the testing phase, if possible, with subjects that can benefit from this product.

Finally, it will be necessary to evaluate the extrapolated data and draw conclusions about the effectiveness of the developed prototype.

1.4 Motivation

Brain-Computer Interfaces was always a fascination of mine. The fact that a human being could, in the future, play videogames, work, move, or do various tasks just by thinking them would be a massive step forward in the technological community, and in human evolution.

As an engineer, I acknowledged this task, not only as the challenge that it is, but also as an opportunity to widen horizons, and to help people that suffer from this condition.

As a person who fortunately does not suffer from a mobility disability, but has family members that do, I accepted this challenge with the intention to give more independence and provide a better quality of life to my relatives.

1.5 Document Structure

This dissertation will be structured in seven chapters. Chapter 1 provided the introduction for this article, contextualizing the problem, objectives and motivation for the creation of this work. Chapter 2 is our State of the Art, where it is provided insight and analysis of devices known as Brain-Computer Interfaces; Chapter 3 is our Value Analysis where our value proposition is analyzed, as well as the expected architecture for the project, and many tools, frameworks, and other support technologies intended for its development. This will claim the first step of the developmental phase of our project. Chapter 4 supplies a set of possible solutions to be used with BCIs and with the chosen approach, as well as the choice of Brain-Computer Interface, we will be able to advance on the development of the prototype. It will provide some analysis on how we intend the subject to use the software, while also providing some use cases, as well as UI/UX insights for some of the mentioned solutions. Chapter 5 documents the development of

the software. Chapter 6 covers the testing documentation as well as some possible predictions expected from the analysis of data from the test subjects. Finally, in Chapter 7, conclusions are drawn describing the effectiveness of the prototype and future work will be identified in order to improve the developed software.

2 Analysis of Brain-Computer Interfaces

This chapter contains the State of the Art. Here, we start with an introduction to the history of Brain-Computer Interfaces, also introducing problems that could be, or have been, sorted out with the addition of Brain-Computer Interfaces in the medical care. Furthermore, we provide insight and analysis on these devices and their components, as well as within the various manufacturers in the world, select a few that could be assets in the development of the project, compare them and make a final choice on which one is going to be used.

2.1 Introduction

A Brain-Computer Interface is a link between the brain and an external device. Being a narrower field of research derived from the Human-Computer Interface, BCIs follow the same principle by capturing the necessary data to start an operation, leading to the processing of that same information into a specific output designed by the developer of the product.

As previously mentioned, a BCIs' main purpose is still towards biomedical research, but some of its applications can now be used in the medical field with patients that are unable to perform any kind of action due to a disorder or condition that renders their body paralyzed.

A good, but unfortunate, example of an application of BCIs in real-life medicine is when patients suffer from Locked-in Syndrome.

Locked-in syndrome (or LIS) is a rare neurological condition that causes full paralysis of all voluntary muscles except those that control eye movements. (Anon., 2018) They would need to rely on a caregiver for the rest of their lives, but with the progress made in recent years on BCIs, people who suffer from this disorder could start regaining some freedom from their own prison.

LIS was first discovered and defined in 1966 as "quadriplegia, lower cranial nerve paralysis, and mutism with preservation of consciousness, vertical gaze, and upper eyelid movement" but later in 1986 was redefined as "quadriplegia and anarthria with preservation of consciousness".

(Smith & Delargy, 2005) The reasoning for this redefinition was to “clarify that mutism could imply unwillingness to speak” instead of being unable to do it. (Smith & Delargy, 2005)

A well-known case of Locked-In Syndrome comes from a man named Richard March, who suffered a stroke in May of 2009 and “found himself fully conscious but unable to communicate”. (ABC, 2014) For more than 4 months, he was treated in a long-term care facility and was able to walk out on his own. Later, it was reported he had regained 95% of his mobility and that he exercises every day. (Amelia Hill, The Guardian, 2012)

Another case of LIS involves the journalist and writer Jean-Dominique Bauby, who also suffered a stroke and fell into a coma at 43 years of age. Bauby managed to wake up a few weeks after the fact, but soon found out he was only able to move or blink his left eye losing all range of motion on the rest of his body. (The New Atlantis, 2018) In 1997, against all odds, Bauby was able to write a book named “The Diving Bell and the Butterfly” while suffering from this condition. This was accomplished by him being read the Alphabet and, when he blinked his eye, someone would write the letter that was stopped in. (Thomas Mallon, The New York Times, 1997)

Two days after the French publication of his book, Bauby died of heart failure.

This condition is a silent killer, some leading to the death functioning beings whose only problem is physical, leaving the neurological side mostly intact.

Being a rare disorder, so the public is aware of this reality, medical television shows portray this condition in their own way, but without altering it too much from the truth.

Figure 1 and Figure 2 portray a scene from the series House MD where, in this episode, one of the patients shows signs he is suffering from LIS.

Figure 1 - House MD Season 5 Episode 19: "Locked In"[1]

Figure 2 is an augmentation of Figure 1, presenting the software used by the patient to answer the doctors' questions.

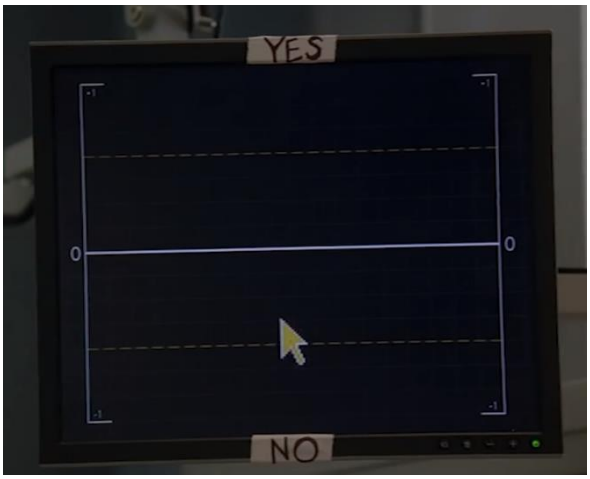


Figure 2 - Brain-Computer Interface seen in Figure 1 [2]

As seen in Figure 2, the Brain-Computer Interface shown in this episode only outputs a "simple" binary answer, and although its output is a simple one, the process behind it always follows the same path.

To understand how BCIs work, we need to first understand how they are built.

[1] Available in: https://youtu.be/mHtw75p6qRI
[2] Available in: https://youtu.be/mHtw75p6qRI

2.2 How do Brain-Computer Interfaces work?

A Brain-Computer Interface system normally involves 4 components: signal acquisition, feature extraction, feature translation, and device output. (Shih, et al., 2012) These components manage the human intent on making a specific movement and converting it into usable data so that the output module connected understands what it must do.

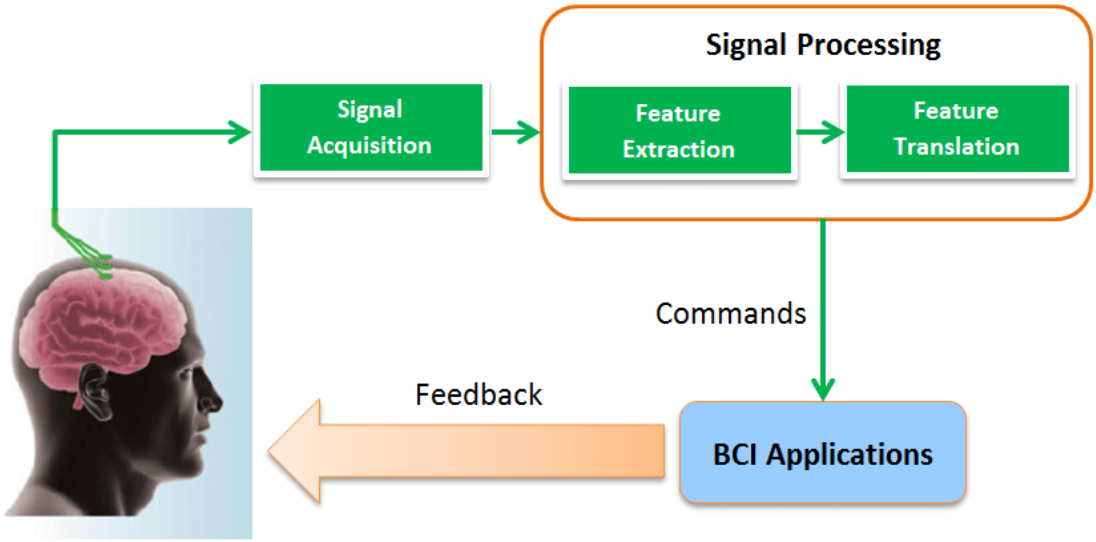


Figure 3 - Components of a Brain-Computer Interface[3]

2.2.1 Signal Acquisition

Signal Acquisition is the capturing and recording of the brain waves created by our brain using sensors.

These sensors can measure our brain signals with sensors divided into 3 categories:

- Non-Invasive;
- Partially Invasive;
- Invasive.

The Partially Invasive category is often dismissed because of the degree of invasiveness provided by some Brain-Computer Interfaces, which are not nearly as aggressive as others. Rather than differing Partially Invasive from Invasive BCIs, part of the scientific community

[3] Available in: (Nguyen, et al., 2015)

unites both categories into one. This not only widens the range for what is deemed as Invasive interfaces but also aggravates some of the less invasive interfaces that are not nearly as dangerous as trully Invasive Brain-Computer Interfaces.

While both of these 2 categories require the need to perform surgery in order to implant either type of interface, their main difference is: partially-invasive BCIs are devices "implanted inside the skull but rest outside the brain rather than within the grey matter" (Javaid, n.d.); while Invasive BCIs are devices "implanted directly into the grey matter of the brain during neurosurgery" (Javaid, n.d.).

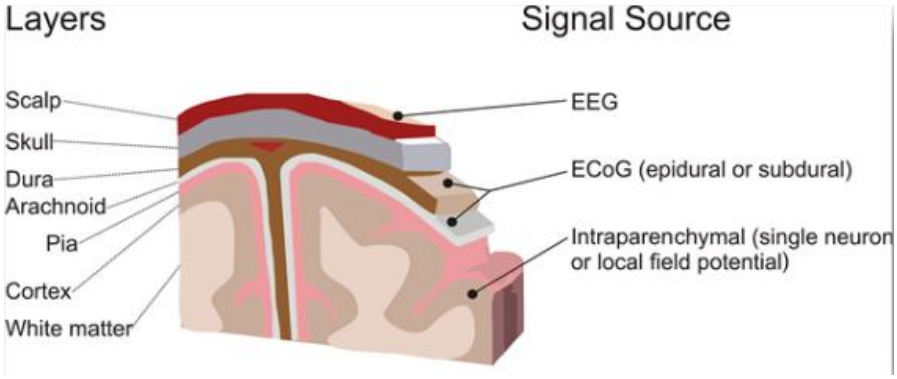


Figure 4 - Brain composition and possible signal sources to capture within the 3 different categories[4]

2.2.1.1 Non-Invasive BCIs

Non-Invasive Brain-Computer Interfaces, as the name asserts, are interfaces that do not require neurosurgery in order to start using them. These BCIs are mainly headsets that rest over the head of the user capturing their brain signals to be processed later.

Electroencephalograms (EEG) and Magnetoencephalograms (MEG) are the best examples of non-invasive BCIs.

Electroencephalograms capture and record the "electrical activity of the brain from the surface of the scalp" (NeuroTechEdu, n.d.). The electrical activity is captured using electrodes in predetermined positions, some devices use their electrodes positioned on an EEG cap like the example from Figure 5.

[4] Available in: (NeuroTechEdu, n.d.)

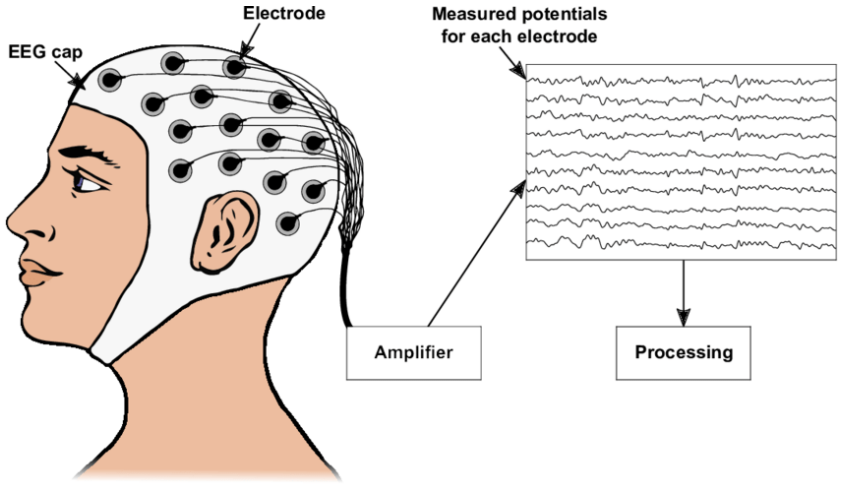


Figure 5 - Example of an EEG is recorded using a cap[5]

Other devices use fully-fledged headsets to capture the intended signals like in Figure 6.

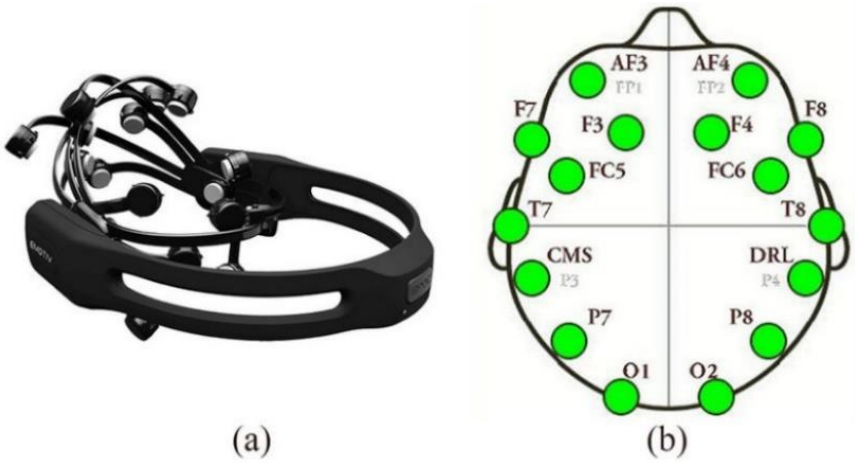


Figure 6 - Example of an EMOTIV Epoc+ EEG Headset (a), and its electrodes' positions (b)[6]

Magnetoencephalograms are "neuroimaging techniques for mapping brain activity by recording magnetic fields produced by electrical currents occurring naturally in the brain, using very sensitive magnetometers" (NeuroTechEdu, n.d.).

[5] Available in: (Nagel, 2019)
[6] Available in: (Kotowski, et al., n.d.)

Figure 7 - Example of a MEG scanner[7]

MEG scanners "can be used to study any aspect of sensory or cognitive function and provides high temporal resolution maps of cortical activity" (Cardiff University, n.d.).

2.2.1.2 Partially Invasive BCIs

Partially Invasive Brain-Computer Interfaces, as mentioned previously, already require neurosurgery in order to be used. These devices are mainly Cortical implants which, after being implanted, can interface with different areas of the cortex, as seen in Figure 4.

An Electrocorticography (ECoG), or intracranial electroencephalography (iEEG), is the process used to capture and record the electrical activity from the brain. This process is essentially the same as an Electroencephalogram with more accuracy, as seen in Figure 8, although ECoGs are more widely used for managing epilepsy.

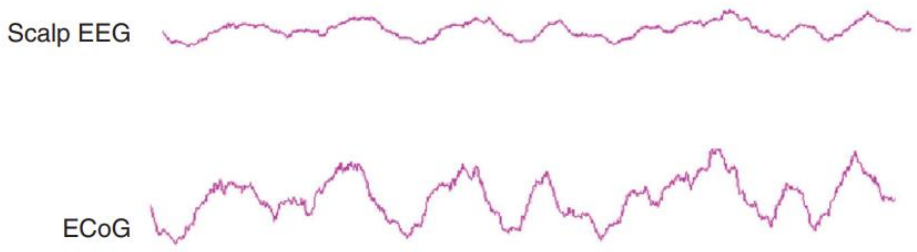


Figure 8 - Comparison between an EEG and an ECoG signal[8]

[7] Available in: (Cardiff University, n.d.)
[8] Available in: (Ortiz-Rosario & Adeli, 2013)

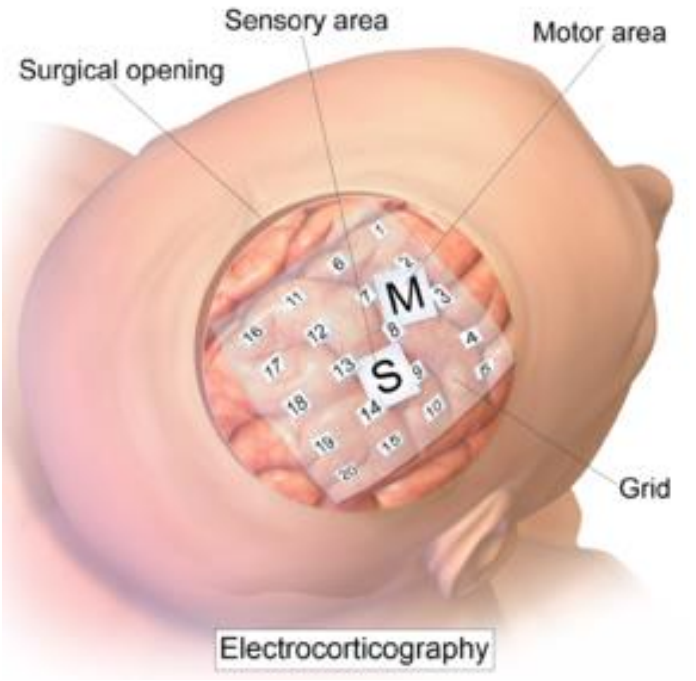


Figure 9 - Example of an Electrocorticography[9]

2.2.1.3 Invasive BCIs

Invasive Brain-Computer Interfaces, as mentioned earlier, are devices "implanted directly into the grey matter of the brain during neurosurgery" (Javaid, n.d.). These devices can be single-unit BCIs or multi-unit ones.

Single-unit (SU) BCIs only "detect the signal from a single area of brain cells", while multi-unit BCIs "detect (the signals) from multiple areas" (NeuroTechEdu, n.d.).

Multi-unit interfaces are normally Local Field Potentials (LFP) which, in turn, capture "electric potential in the extracellular space around neurons" (Destexhe & Bedard, 2013).

These units are, normally, square-shaped containing several electrodes (pins). These electrodes "can have arbitrary lengths up to several cm or, for example, up to 1.5 mm (Utah, Blackrock Microsystems) or 10 mm (FMA, MicroProbes) in a multi-electrode array (MEA)" (Waldert, 2016).

[9] Available in: (NeuroTechEdu, n.d.)

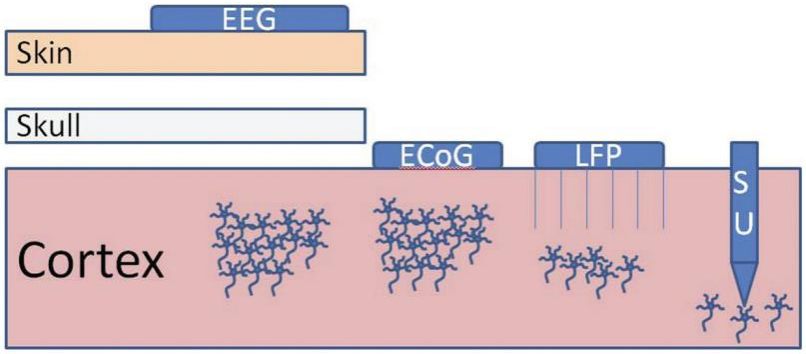


Figure 10 - Electrodes locations for different interfaces[10]

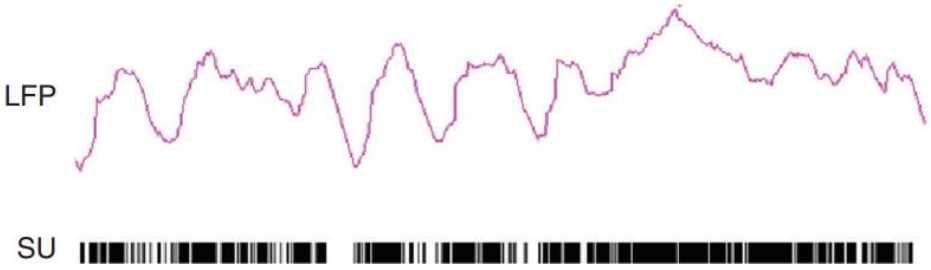


Figure 11 - Comparison between a LFP and a SU signal[11]

Elon Musk's Neuralink technology, although never specified, seems to use technology ranging between ECoG and LFP interfaces to capture the electrical brain signals they need to use on their application. Similar interfaces are also currently used for the treatment of patients with Parkinson's disease.

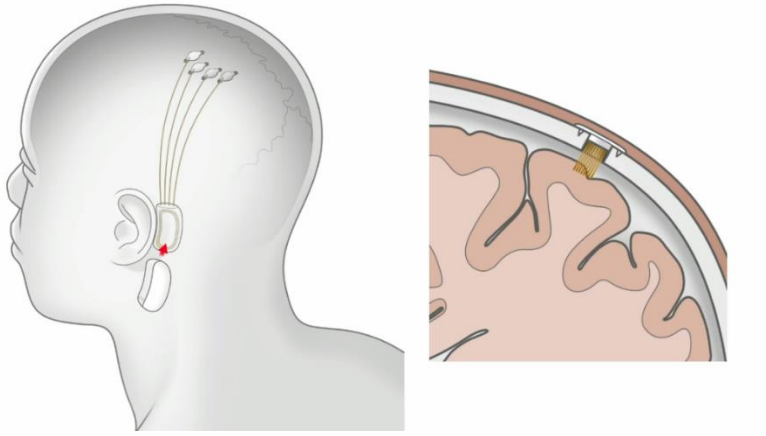


Figure 12 - Neuralink electrodes' implants[12]

[10] Available in: (Ortiz-Rosario & Adeli, 2013)
[11] Available in: (Ortiz-Rosario & Adeli, 2013)
[12] Available in: (Neuralink, 2019)

2.2.2 Feature Extraction

Feature Extraction is the second component to a Brain-Computer Interface, this component analyzes the incoming signals from the previous component and filters them to keep the relevant data, related to the user's intentions, from the excess noise generated by our brain for the body's involuntary actions (like, breathing, etc.).

"These features should have strong correlations with the user's intent" (Shih, et al., 2012), meaning that the number of samples collected from brain signals are then normalized into one or more signals, depending on the position of the electrodes, that are then sent to the next component where they get "translated" into commands that the device output understands.

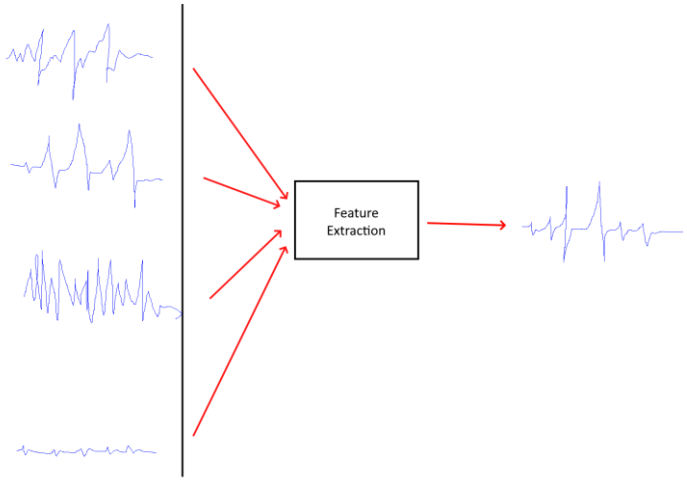


Figure 13 - Representation of brain signals being processed in the Feature Extraction

2.2.3 Feature Translation

Feature Translation is the third of the four components, where the signals processed in the Feature Extraction get converted "into the appropriate commands for the output device" (Shih, et al., 2012).

For this to happen, a model reading of the user's intent is to be recorded several times in order to have a decent representation of his brain signals for the intended action. This process must be repeated beforehand, for each intended action, to train the Interface on what the user is thinking with the purpose of reaching the intended output.

2.2.4 Device Output

The device output is the final component of the Brain-Computer Interface, and it is where the command from the Feature Translation gets sent to the device operated by the user. It can be a predetermined device where the developed software only works on it, but it also could be a more dynamic software where, instead of working as a standalone application, it functions as a middleware platform where it connects the user's intent to a preset action configured by the end-user without having the need to develop or write any code on it.

At the time of writing, BCIs are used mainly in Medical Research fields with patients suffering from a wide range of illnesses, but there are other uses that a BCI can be applied to, including, but not limited to:

- Communication with Locked-in Syndrome patients;
- Multiple Sclerosis patients for communication and some movement;
- Prosthetics control;
- Clinical Trials; and
- Game interaction. (Gu, et al., 2020)

2.3 Existing Brain-Computer Interfaces in the market

Most of the BCIs available to the public consumer are expensive and only for research purposes. These manufacturers need to certify their products with specific certifications for them to be able to sell them.

A well-known manufacturer of Brain-Computer Interface Headsets, called EMOTIV®, specifically warns that their products are only intended for personal and research purposes and not as medical devices under the EU Medical Devices Directive 93/42/EEC. (Emotiv, 2020)

According to Market Research Future, the key players on the Brain-Computer Interface market are (Market Research Future, 2020):

- EMOTIV;
- Natus Medical Incorporated;
- NeuroSky Inc;
- Compumedics Limited;

- NIHON KOHDEN CORPORATION;
- G.tec;
- Advanced Brain Monitoring Inc.;
- Brain Products GmbH;
- BrainCo Inc.;
- Neuroelectrics;
- MindMaze.

The applications of brain-computer interfaces are not limited to this industry. As mentioned before, other industries have already started using this technology to improve their work sector, such as defense and aerospace, as well as education. (Market Research Future, 2020)

For the current work, a few Brain-Computer Interfaces were analyzed with the intent of purchasing one to develop the intended software to use with a mobility aid.

The BCIs analyzed were:

- EMOTIV Epoc X (EMOTIV);
- EMOTIV Insight 2 (EMOTIV);
- Muse 2 Headband (InteraXon);
- Muse S (Gen 2) Headband (InteraXon);
- Gelfree BCI Electrode Cap Kit (OpenBCI).

2.3.1 EMOTIV Epoc X

Manufactured by EMOTIV®[13], the Epoc X, according to their product page, is designed for human brain research, "providing access to professional-grade brain data with an improved and easy-to-use design". (Emotiv, n.d.)

[13] Available in: https://www.EMOTIV.com/

Figure 14 - Pictures of the EMOTIV Epoc X[14]

It is $849, containing 14 sensors to capture brain signals accurately, it can be connected wirelessly for more mobility, the headband is adjustable in order to be more comfortable for the wearer, and its sensors are saline-based electrodes.

2.3.2 EMOTIV Insight 2

Also from EMOTIV®, the Insight 2, is a less feature-heavy device, but still usable on multiple situations. According to the product page, the Insight "boasts advanced electronics that are fully optimized to produce clean, robust signals anytime, anywhere" (Emotiv, n.d.)

Figure 15 - Pictures of the EMOTIV Insight 2[15]

[14] Available in: (Emotiv, n.d.)
[15] Available in: (Emotiv, n.d.)

With only 5 sensors, it is still capable of measuring the activity “from all cortical lobes of the brain” (Emotiv, n.d.), and like the aforementioned EMOTIV Epoc it can be used wirelessly. However, unlike it, it holds 10 hours more of battery life, and its sensors are made of a semi-dry polymer. This device will cost the customer $499.

2.3.3 Muse 2 Headband

The Muse 2, manufactured by InteraXon, is a proprietary device that is intended for the end-user more than developers. It provides EEG readings, as well as heart rate, body relaxation, and breath monitoring if the user is interested.

Figure 16 - Pictures of the Muse 2 Headband[16]

Built with 7 EEG sensors, “2 on the forehead, 2 behind the ears plus 3 reference sensors” (MindtecStore, n.d.), the Muse 2 can accurately detect and record the activity from the brain. In addition to the 7 existing sensors, and contrary to its predecessor, the Muse 2 also contains heart sensors located in the front of the device, as well as PPG and pulse oximetry sensors for other applications. It also includes a gyroscope and accelerometer located behind the ears.

2.3.4 Muse S (Gen 2) Headband

The Muse S, also produced by InteraXon, incorporates all of the features of the previously mentioned model but also integrates the ability to monitor the user’s sleep.

[16] Available in: (InteraXon, n.d.)

Figure 17 - Pictures of the Muse S Headband[17]

The only main differences between the Muse 2 and the Muse S are the addition of the sleep monitoring feature and the battery life. Considering this, the cost for the Muse 2 is $249.99, and for the Muse S is $399.99.

2.3.5 EEG Electrode Cap Kit

The EEG Electrode Cap, provided by OpenBCI, is a 21-channel electrode cap compatible with most open-source software available on the average consumer market. Its contents include the white cap with 21 electrodes, a syringe to irrigate the electrodes, and a cleaning brush. (OpenBCI, n.d.)

This cap was developed with the goal to provide good signal quality as well as comfort, as it was designed for the possibility of being used in sleep studies.

Figure 18 - Pictures of the EEG Electrode Cap Kit[18]

[17] Available in: (InteraXon, n.d.)
[18] Available in: (OpenBCI, n.d.)

2.3.6 Comparison Table

The following table's information was gathered from EMOTIV products' webpages, as well as from InteraXon's Muse products and OpenBCI products' webpages. Furthermore, to complete the table's information, some of the missing information was collected from tech review websites.

	EMOTIV Epoc X	**EMOTIV Insight 2**	**Muse 2**	**Muse S (Gen 2)**	**EEG Electrode Cap Kit**
Brand	EMOTIV		InteraXon		OpenBCI
Number of sensors	14	5	7		Up to 21
Sensor placement	AF3, AF4, F3, F4, FC5, FC6, F7, F8, T7, T8, P7, P8, O1, O2	AF3, AF4, T7, T8, Pz	Forehead, behind ears		FP1, FP2, F3, F4, F7, F8, FZ, C3, C4, CZ, T7, T8, P3, P4, P7, P8, PZ, O1, O2
Connectivity	Bluetooth 5.0, BLE		Bluetooth 4.2		Wireless
	Proprietary USB receiver, USB 2.0		Micro USB		Serial
Sensor location	Fixed		Fixed		interchangeable
Battery life	9 hours	20 hours	5 hours	10 hours	N/A
Price (USD)	$849	$499	$249.99	$399.99	$499.99

Table 1 - Comparison table between chosen Brain-Computer Interfaces[19]

[19] Data gathered from: (Emotiv, n.d.); (Emotiv, n.d.); (InteraXon, n.d.); (MindtecStore, n.d.); (Wareable, n.d.); (OpenBCI, n.d.); (OpenBCI, n.d.)

2.4 Selecting device

After the analysis of the previously demonstrated options, the chosen device would have to meet the following minimum requirements, in order to accommodate the end-user:

- Long-lasting battery life, for all day-to-day responsibilities;
- Easily concealed, but functional;
- Easy to use;
- Not very expensive, considering the market we are in.

The EMOTIV Insight 2 was chosen as the Brain-Computer Interface to meet these objectives.

3 Value Analysis

This chapter contains the project's Value Analysis, which analyzes our value proposition, as well as the project's expected architecture and many tools, frameworks, and other support technologies intended for its development. This will be the first step in our project's development process.

3.1 Value Proposition

A value proposition is a company's description of the product or service that summarizes how it is being delivered, acquired, and experienced. A value proposition, essentially, specifies what makes a company's product interesting, why consumers could use it, and how its value differs from the competition.

Answering the previous questions, our final value proposition is a software that, in short, will link a Brain-Computer Interface with a mobility aid, for example, an electric wheelchair. The software would learn from the user’s intentions and apply those intentions into actions on the chair. Although this software is intended for someone with low mobility, its uses could be applied in various fields and scenarios like moving a shopping cart on its own, or playing computer games where the player moves without any keyboard or mouse input. Although still small, this area of Brain-Computer Interfaces is evolving exponentially, and this software is a step forwards in the communication between a human and a computer.

3.2 Architecture

The expected architecture for this solution, shown in the following Figure, will include the usage of a Brain-Computer Interface connected to the software, that is going to be developed, installed in a mini-PC, like a Raspberry Pi, that consequently is connected to the intended device output, for example, the electric wheelchair.

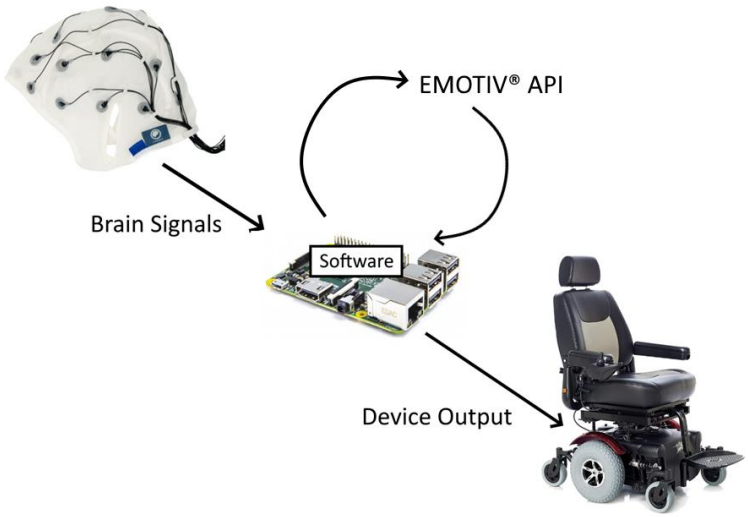


Figure 19 - Representation of the expected Architecture of the Project
(BCI sends the brain signal to the Raspberry Pi, which converts them into commands performed by the electric wheelchair)

3.3 Technologies

Asserting the first stage of the developmental process of the current project, this chapter will provide an analysis and choice of the possible technologies, APIs, and other support tools that are planned to be used.

But before we proceed with the analysis and choice of any of the following technologies, a few requirements will have to be met for the software to function as intended. The list of requirements is:

- The software will have to control hardware;
- The software will have to connect to the internet, in order to connect to any API necessary for the brain reading to function properly;
- The software will have to be easy to use;
- The software will have to interact with the user.

3.3.1 Programming Language

As it is well known within the developer community, the choice of the programming language to use in any project is something not to be taken lightly as it will be the backbone for everything

that comes after, and so it must be debated. Some languages take advantage on the efficiency they provide the user, some on the mass-scale calculations they are able to deliver without slowing down too much. These are some of the conditions we would need to be careful in order to have a functional and swift interface.

Python would be the best fit to develop the spine of this project since it is an open-source project, multiplatform, and, at the time of writing, it has a very wide and active community. It is very complete and versatile, having a comprehensive number of modules and libraries available, while still being able "to interact with a wide variety of programming languages". (Alonso-Valerdi & Sepulveda, 2011)

Figure 20 - Python Programming Language Logo[20]

As the base language for our project, Python would be able to easily control any hardware with the desired movement; with help from some communications' modules, it would be able to connect the needed support APIs provided by our device's manufacturer, as well as to create an interactive interface easy to use for the device's operator.

Functionality being the crucial objective for the device's software, its Frontend User Interface would fall behind the requirements as a secondary goal to providing a complete interface.

3.3.2 Support Technologies

There are many support technologies that could be implemented in the project in order for it to reach a completed stage more swiftly. These technologies could be implemented as Modules or Libraries to the project's codebase in order to simply add more functions or methods to ease on the link between the hardware or software, like a cloud-based resource; or it could be implemented with APIs where we generate the needed data on-site and, after making a connection with the manufacturer's API, send that data to their servers where it gets processed

[20] Available in: (Python, n.d.)

and sent back with the necessary or readable data to generate movement on the device's output.

3.3.2.1 Modules/Libraries

Modules or Libraries are extensions of the existing codebase that, after being imported, provide the on-development software the necessary tools to more easily complete the tasks they were provided. (Python, n.d.)

This being a pre-developmental phase, it is impossible to provide an accurate list of all the modules/libraries that will be necessary to complete the software, but given the previous requirements, some necessary modules would be:

- A mathematical module to swiftly process any kind of equation or signal;
- An algorithm optimization module to mainly complement the previous mathematical module and its methods;
- A data analysis module to convert the previous processed data into readable commands for the output to follow;
- A connectivity module to either link the software with the BCI, with the cloud or with the output's hardware.

With the aforementioned list, the focus of the project's development gets directed to the implementation and simplification of reading the brain signal data provided by the user, and consequently producing movement on the output device's hardware.

3.3.2.2 Application Programming Interfaces (APIs)

API is a "set of definitions and protocols for building and integrating application software" (Red Hat, 2017).

As previously mentioned, this being a pre-developmental phase, it is impractical to identify which APIs would be absolutely necessary to the development of this software, but, given the already chosen input hardware, it is almost certain we would, at least, need access to the EMOTIV API (EMOTIV, n.d.) in order to process the user's brain signal accurately.

4 Solution planning and analysis

This chapter will continue with the planning of potential solutions and their corresponding analysis after the initial stage of development. Each solution will be presented with a description of the approach to follow, the types of difficulty and the possible constraints that could arise with their development, along with the identified advantages and disadvantages to the developer and the end-user for each approach.

One thing should be remembered at all times, first and foremost. This software will be used as an interface "directly" connected to the human brain to control an electronic object. This means that there are core components to the development of the solutions that cannot be changed.

4.1 Solutions

A project like the one currently being hypothesized, can have many solutions to the list of problems previously identified in this document. Each solution will depend on a variety of factors, including target audience (who will benefit from the program) and usability (how readily anyone who requires it may use the product). In order for us to come up with workable ideas and then determine which of them should be implemented, each of these factors will need to be carefully taken into account.

For maintenance of any possible solution, the product must be regularly evaluated in the fields of safety, usefulness, effectiveness, and efficiency. To do this, users should be able to provide feedback regarding some questions that will serve as evaluation metrics:

- Safety – How safe is it to use this product?
- Utility – Are there any superfluous functions? Is the response time adequate? Are there any errors?
- Effectiveness – Is appropriate support to the product provided? (i.e.: a user interface with task coverage and detailed information about the procedures to be followed when using the product; an instructions manual; etc.); Is the user satisfied with the product?

- Efficiency – Is the performance of this product good (meaning that, it performs all of the tasks with success)? Is the success rate high?

This evaluation should be carried out on a regular basis, as the software and its UI may need to be updated at any time.

4.1.1 Direct approach

With the direct approach the user would simply have the need to power on both the BCI headset as well as the mobility aid. The communication between them would be attempted automatically, and in case of success the user would receive either an audible alert or a visual one, alerting that the device could be operated. The same would happen in case of an unsuccessful connection with a different kind of audible/visual alert.

Figure 21 - Direct connection between BCI headset and mobility aid

This, concerning usability, would be the easiest way for the end-users to use the product successfully, despite the main problem of needing a long period of time to adapt themselves to the movement commands.

Concerning development, such a "plug and play"[21] approach would require longer time of development not only to develop the main software of controlling the mobility aid - which would also necessitate compatible hardware and software to make it happen - but also to

[21] Term used when setup, on a device, given by the end-user is not necessary

create safety measures, for instance in the case the device disconnects or does not connect successfully on the first try.

4.1.2 Semi-direct approach

Following the principle of the previous solution, the semi-direct approach would need the BCI headset device to communicate with the mobility aid indirectly with the assistance of a middleware. This middleware would be a secondary system with the ability to connect with the BCI headset and the mobility aid would connect to it in order to receive the necessary information to generate movement.

Figure 22 - Indirect connection between BCI headset and mobility aid

Regarding usability for the end-user, this solution would follow in line with the previous, direct, approach where the user would only need to power on the mobility aid, and it would give the necessary power to all secondary systems (middleware included) in order for the connection with the headset to be successful.

The development of this system would largely follow the same path as the prior approach, but with the added constraint of a second connection with the possibility of failing between the middleware and the mobility aid. However, the use of a middleware device greatly increases the combination of devices that can end up being used and adapted to create a “smart” mobility aid.

4.1.3 Dynamic approach

With the Dynamic approach, the development would not be focused as much on the end result but on the features the software could provide. For example, the main difference to be identified with the previous approaches would be that, instead of learning how to think in order to move the mobility aid, the user would teach the software how he would think so the device could move.

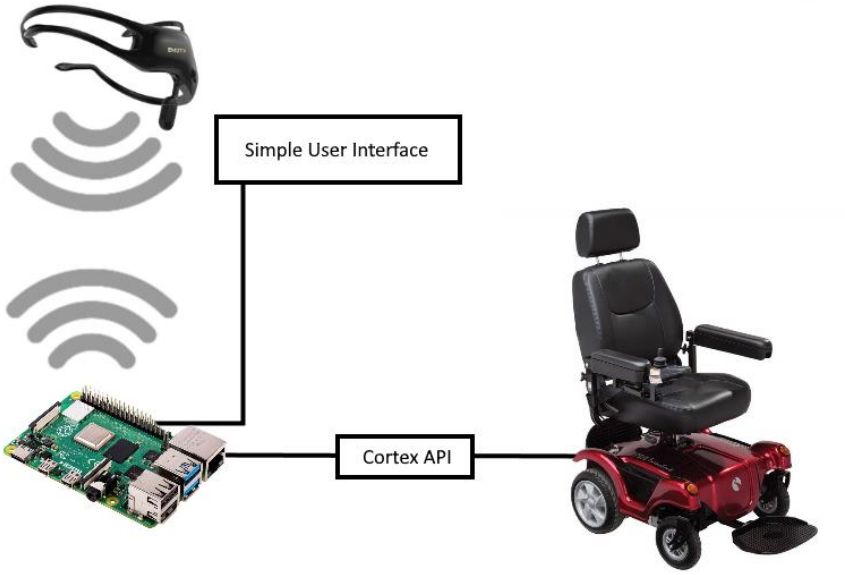


Figure 23 - Indirect connection with Dynamic approach for communication between the user and the device

This approach would include a slew of preliminary tests that would capture the user's brain waves while performing said tests on the mobility aid. This way, after the tests were completed and the captured data compiled, the software would know exactly how to act and perform the commands from the user's thinking.

In terms of usability, this approach would not be as direct as the previous ones requiring some setup from the end-user and interaction with an interface for him to start the initial tests.

Regarding development, this strategy has a high degree of difficulty which means it would take a long time just to have functioning prototype. Given the possibility of problems during the development process, such as the user's thinking overlapping in the taught commands, resulting in the software failing to perform the desired task, or a failure in converting the desired

command into the incorrect one, this approach may be deemed too unreliable to be developed successfully.

4.1.4 Complete solution

The Complete solution would follow the previous, dynamic, approach as a basis for its development with added elements and features. The software was hypothesized to be capable of maintaining direct communication – for users that would prefer to follow the previously mentioned direct approach – as well as indirect communication with the headset – following the semi-direct approach. With the indirect communication strategy, through the means of the middleware, the users would be given access to more features and functions, allowing the software to perform as more than just a mobility aid.

Said features could include but not be limited to:

- The control of a mouse/keyboard on a computer connected to the middleware device;
- The control of simple devices in the close vicinity of the user, such as, remote controls, light switches, etc…

From a usability standpoint, the target audience for this approach would be users with still some mobility, at least, above the waist for them to use the extra features effectively. In order to do this, they would need to use a screen and a manual interface, like a touch screen, whose default function would be to send movement commands to the mobility aid. However, the user could choose other functionalities at any time to fulfill their needs.

Concerning development and given that it would include more features than the previous approaches mentioned, the development of this approach would be the most time-consuming out of the previous ones, most likely requiring a multi-year plan just for the software to be reliably used by one end-user. This is due to each person's brain and way of thinking being unique. This fact may also make this solution too unreliable to develop without a long-term commitment.

4.2 Possible User Interfaces

Following the establishment of the previous solutions, if we were to develop one of the more complex ones, Dynamic Approach or Complete Solution, some User Interfaces would be required for the software to look effective and informative but also easy to use by the end-user.

When designing a possible user interface so that the users can establish better communication with the technological equipment, many aspects would have to be taken into account, as human beings can vary in many dimensions:

- Their sensing and memory capabilities may vary with their age;
- Their abilities may affect the suitability of the interface – for example, disabilities like blindness, deafness or even the user's dexterity require a special attention to an inclusive design or even the use of assistive technology (i.e. braille displays, screen readers, hearing aids, typing aids, dedicated speech devices, etc.);

The usability objectives of this product are related to its efficiency, effectiveness, safety and usefulness. Thus, it is intended that users who will use it feel helped and that the probability of needing assistance from another person in using the product is reduced or disposable.

For each previously mentioned solution - with the exception of the direct approach, which does not allow a graphical interface because it does not employ a middleware; and the semi-direct approach, which would follow the same principles as the direct approach while only adding a layer of compatibility between headsets and mobility aids -, some sketches that could be included in a possible user interface will be presented below. These sketches could be demonstrated on a device (i.e. a tablet) that would be attached to the wheelchair, with the purpose of giving feedback to the user and guiding him through the different actions that he can perform.

In order for the design to be inclusive for people with hearing impairments, the information will be present both visually and audibly. In the case these deficiencies existed, or even in the case of the impossibility of movement of the hands by the user, the use of this device would have to be carried out with the help of another person.

4.2.1 Dynamic Approach

In this solution, the software would be presented with an informative section, not requiring the physical interaction of the user in these parts, displaying feedback regarding the stability of the connection between the wheelchair and the headset (Figure 24); and direction in which the chair is being moved (Figure 25). However, the user's input (click) would be required in the sections where the equipment would be intended to learn the user's way of thinking. It could also be necessary to reprogram this thought, so the user would be able to incite this action on the device at any time (Figure 26).

Figure 24 - Feedback of stability of connection

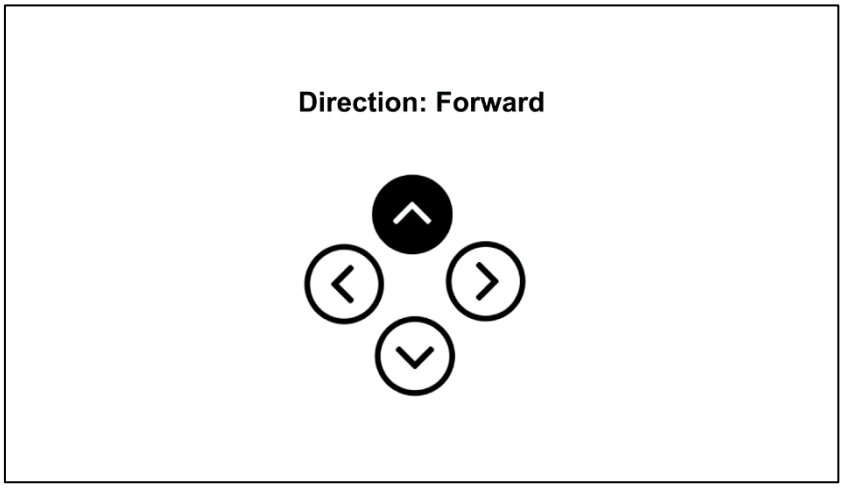


Figure 25 - Visual feedback of the movement being enacted

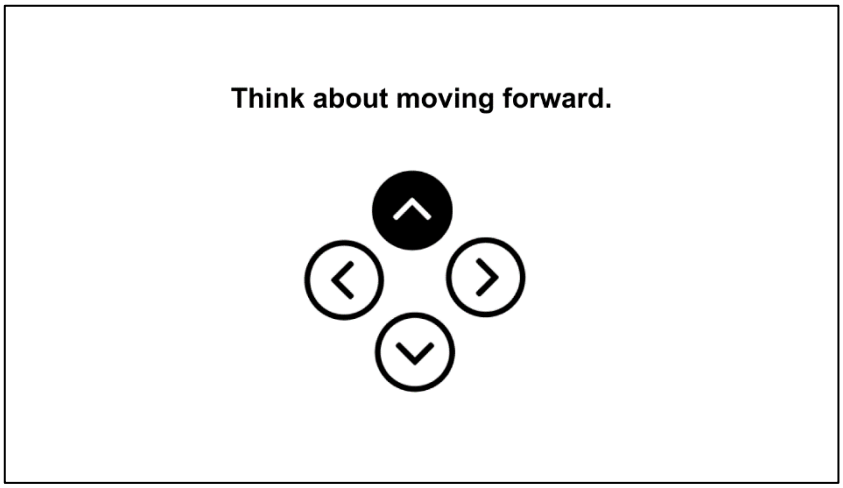


Figure 26 - Indication for the software to learn the user's thinking

4.2.2 Complete Solution

The complete solution would foresee the use of the sketches presented in the previous approach, with the addition of the extra functionality of controlling more devices, in other than the wheelchair. For this, a specific UI for each device would have to be studied, so Figure 27 would only present a possible outline of a menu for choosing the desired device.

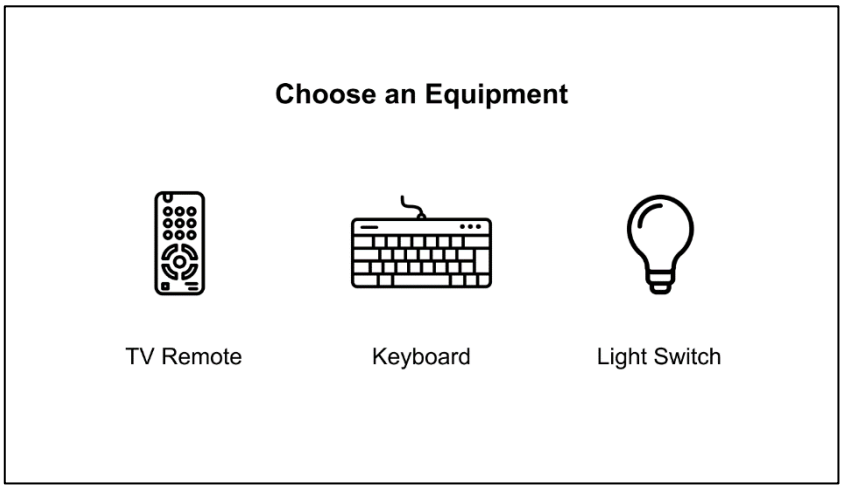


Figure 27 - UI for the user to choose other equipments

4.3 Comparative Analysis

Following the creation of some possible User Interfaces and for a better understanding and synthesis of the solutions previously discussed in terms of their development, their advantages and disadvantages will be demonstrated in the following table.

	Advantages	Disadvantages
Direct approach	• Direct communication between the BCI headset and the mobility aid • Simple and straightforward development of the solution	• High incompatibility between hardware devices (headset – mobility aid)
Semi-direct approach	• Simulates a direct communication between the BCI headset and the mobility aid • Less incompatibility between hardware devices	• Likelier to fail if not introduced to the end-user with the necessary precautions for a strong and secure connection (It should retry the connections a few times before alerting the user of an error)
Dynamic approach	• Instead of the user learning how to think, the system will learn how the user thinks compiling the commands in order to move the device	• Long-term commitment for research and development • Possible incompatibilities with the software integrated with the hardware devices (Teaching intentions is harder than learning them to perform a task)
Complete Solution	• The end-user would be able to use the software not just as a mobility aid, being able to control other devices connected to it	• Extremely long-term commitment for research and development • High incompatibility between hardware devices with software implications (Some devices could not be connected to easily) • More complex/detailed UI

Table 2 - Comparative analysis table between the discussed solutions

With the analyzed solutions and information gathered up to the point of writing this section of the article, the semi-direct approach was chosen to begin implementing because it would be, hopefully, a straightforward process on the development of the prototype.

5 Development of the chosen solution

With the approach selected beforehand, this chapter will focus on the development of the prototype for this solution.

Following the research conducted in this article and the earlier selection of auxiliary hardware and software, the development stage finally begins with the goal of developing a working prototype to be presented alongside this work.

It is important to note that some constraints prevented the development of this prototype from running at full capacity due to time constraints, given the level of research required, and some development limitations, such as working alongside proprietary, deprecated, first-party software. (EMOTIV, n.d.) The selected headset, Emotiv Insight 2, had the limitation of needing to be connected to the company's provided software in order to be used by the developed prototype.

5.1 Technologies and Architecture

Reiterating and reassessing the selected technologies, taking into account the chosen approach and the aforementioned limitations, the to-be-developed semi-direct approach would require:

- the development of controls for the prototype's movement;
- a simple interface for the developer to use while debugging;
- a connection to the first party software in order to connect the prototype with the headset.

As it is not possible for the developer to get ahold of an electric wheelchair, the proof-of-concept prototype will be a remote-controlled car equipped with a minicomputer, on which the developed software by the author will operate in conjunction with the previously mentioned elements.

Figure 28 - Proof of concept RC car + Emotiv Insight headset

To proceed with this stage, the chosen technologies began with the use of the Python programming language for source code development, as well as a few other dependencies for the software to function properly, which are as follows:

- the "Cortex" module, as well as the extensions required to connect the prototype to the headset software (EMOTIV, n.d.);
- some RC car manufacturer-supplied modules, such as motor functions and battery power information (Freenove - through Github, n.d.);
- the "PyQt5" module - to create a simple debugging interface (Riverbank Computing Limited, n.d.);
- threading and dispatch modules - to manage the flow of data between multiple concurrent operations.

5.2 Technical documentation

The development of the software will be divided and documented in three parts, the first where the movement's control of the prototype is developed, establishing automatic and manual testing of the movement; the second where the data readout from the headset is captured and displayed to the developer; and the third and final part where the previous ones are combined into one software to capture, compile and use the data to control the movement of the RC car.

As indicated in attachment A, all documented software will reside and be available in: https://github.com/BarreiraTom/ISEP-TMDEI-Thesis_Project

5.2.1 Control of the prototype

Since the proof-of-concept remote-controlled car was acquired with software to control it from the manufacturer, the Motor module was retrieved in order to be utilized in the development of the software of the prototype.

With this, automatic and manual test phases were developed with the intention to test the prototype's responsiveness to the commands given by the software.

5.2.1.1 Automatic Testing

Starting with the automatic testing, a method was developed to make the prototype generate movement in all 4 directions (front, back, turn right and turn left) for one second each and then terminate.

The program would access a class provided in the Motor module to call on the movement that would be generated by the servos on the wheels.

```
import time
from Motor import *

def main():
    try:
        PWM.setMotorModel(1000, 1000, 1000, 1000)  # Forward
        print("The car is moving forward")
        time.sleep(1)
        PWM.setMotorModel(-1000, -1000, -1000, -1000)  # Back
        print("The car is going backwards")
        time.sleep(1)
        PWM.setMotorModel(-1500, -1500, 2000, 2000)  # Left
        print("The car is turning left")
        time.sleep(1)
        PWM.setMotorModel(2000, 2000, -1500, -1500)  # Right
        print("The car is turning right")
```

```
            time.sleep(1)
            PWM.setMotorModel(0, 0, 0, 0)  # Stop
            print("\nEnd of program")
        except KeyboardInterrupt:
            PWM.setMotorModel(0, 0, 0, 0)
            print("\nEnd of program")

    if __name__ == '__main__':
        main()
```

Code Snippet 1 - Automated movement test for the prototype

5.2.1.2 Manual Control testing

Following the automated testing, this manual testing phase would not only require the Motor module provided but also the PyQt5 module to generate a basic UI displaying the keys being pressed by the manual controls input.

This program is supposed to highlight which key was pressed to simulate the movement. The keys bound to the four directions are known as the movement keys, WASD (W - Forward; A - Left; S - Backwards; D - Right) in the technological/gaming industry, and their function will be included in the final prototype for debugging purposes.

To create this test using the PyQt5 module, two classes were required: one to display the program's window and wait for user input, and another to generate the UI (buttons, labels, etc...) within the window.

The following code snippet is applied to the UI generating class to demonstrate how the software will construct the windows using the portrayed code.

```
    class Ui_Client(object):
        def setupUi(self, Client):
            Client.resize(600, 100)
            Client.setWindowTitle('Keyboard Input Window Test')
            font = QtGui.QFont()
            font.setFamily("3ds")
            font.setPointSize(40)
            Client.setFont(font)

            self.W = QtWidgets.QLabel(Client)
```

```
            self.W.setGeometry(QtCore.QRect(0, 0, 100, 100))
            self.W.setText("W")
            self.W.setVisible(False)

             ### Previous 4 lines of code would be repeated 3 more times in
    order to display the text for the “ASD” labels
```

Code Snippet 2 - UI generating class

Following the creation of this class, the main class would be created with an extension to the previous one, allowing the software to display the UI on the generated window.

```
    class mywindow(QMainWindow, Ui_Client):
        def __init__(self):
            global timer
            super(mywindow, self).__init__()
            self.setupUi(self)

            self.Key_W = False
            self.Key_A = False
            self.Key_S = False
            self.Key_D = False

        def keyPressEvent(self, event):
            if not event.isAutoRepeat():
                if event.key() == Qt.Key_W:
                    print('W')
                    self.Key_W = True
                    self.Key_S = False
                elif event.key() == Qt.Key_S:
                    print('S')
                    self.Key_S = True
                    self.Key_W = False
                elif event.key() == Qt.Key_A:
                    print('A')
                    self.Key_A = True
                    self.Key_D = False
                elif event.key() == Qt.Key_D:
                    print('D')
                    self.Key_D = True
```

```
                    self.Key_A = False

            self.W.setVisible(self.Key_W)
            self.A.setVisible(self.Key_A)
            self.S.setVisible(self.Key_S)
            self.D.setVisible(self.Key_D)

        ### There would be a keyReleaseEvent method following the previous
    function structure doing the reverse of that method, waiting for the release
    of the pressed key on the keyboard
```

Code Snippet 3 - Main class to display window with the keypress listeners

The developed program can be presented operating in the oncoming Figures.

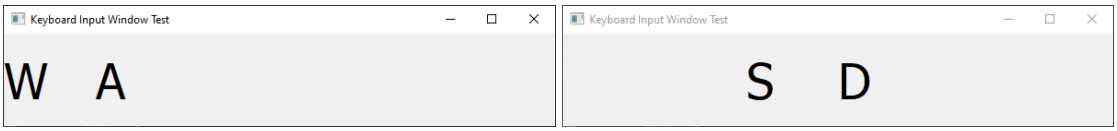


Figure 29 - Compiled test program showing keys being pressed
(left figure – W, A keys pressed | right figure – S, D keys pressed)

5.2.2 Capture readout from the BCI headset

Moving on to the BCI headset capture readout phase, the module in use was provided by the Brain-Computer Interface's company, as was some of their open-source code, to easily test if the reading captured from the headset was being correctly interpreted.

Because this test was created in conjunction with the company's module, the following code snippet will only display the information required to effectively receive and transmit the compiled data to the user captured by the headset. Presenting an example of the captured data as well, in the incoming Figure.

```
    class LiveAdvance(Dispatcher):
        _events_ = ['brainData']

        def __init__(self, app_client_id, app_client_secret, **kwargs):
            self.c = Cortex(app_client_id, app_client_secret,
                            debug_mode=True, **kwargs) #
            self.c.bind(new_com_data=self.on_new_com_data) #
```

```
    def start(self, profile_name, headsetId=''):
        if profile_name == '':
            raise ValueError('Empty profile_name. The profile_name cannot
be empty.')
        self.profile_name = profile_name
        self.c.set_wanted_profile(profile_name)

        if headsetId != '':
            self.c.set_wanted_headset(headsetId)
        self.c.open()

    def on_new_com_data(self, *args, **kwargs):
        data = kwargs.get('data')
       ### Here is where the relevant data will be output
        print('mc data: {}'.format(data))
        self.emit('brainData', data=data)

### REMOVED UNNECESSARY CODE

def main():
    # Please fill your application clientId and clientSecret before running
script
    your_app_client_id = [REMOVED FOR PRIVACY REASONS]
    your_app_client_secret = [REMOVED FOR PRIVACY REASONS]

    # Init live advance
    l = LiveAdvance(your_app_client_id, your_app_client_secret)

     trained_profile_name = 'ISEP-TMDEI'  # Please set a trained profile
name here
    l.start(trained_profile_name)

if __name__ =='__main__':
    main()
```

Code Snippet 4 - Test to capture the headset's reading

```
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341019.2304}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341019.3554}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341019.4804}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341019.6054}
mc data: {'action': 'push', 'power': 0.2, 'time': 1665341019.7304}
mc data: {'action': 'push', 'power': 0.3, 'time': 1665341019.8554}
mc data: {'action': 'push', 'power': 0.6, 'time': 1665341019.9804}
mc data: {'action': 'push', 'power': 0.5, 'time': 1665341020.1054}
mc data: {'action': 'push', 'power': 0.2, 'time': 1665341020.2304}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341020.3554}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341020.4804}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341020.6054}
mc data: {'action': 'pull', 'power': 0.3, 'time': 1665341020.7304}
mc data: {'action': 'pull', 'power': 0.5, 'time': 1665341020.8554}
mc data: {'action': 'pull', 'power': 0.6, 'time': 1665341020.9804}
mc data: {'action': 'pull', 'power': 0.2, 'time': 1665341021.1054}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341021.2304}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341021.3554}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341021.4804}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341021.6054}
mc data: {'action': 'left', 'power': 0.1, 'time': 1665341021.7304}
mc data: {'action': 'left', 'power': 0.2, 'time': 1665341021.8554}
mc data: {'action': 'left', 'power': 0.2, 'time': 1665341021.9804}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341022.1054}
mc data: {'action': 'neutral', 'power': 0.0, 'time': 1665341022.2304}
mc data: {'action': 'right', 'power': 0.1, 'time': 1665341022.3554}
mc data: {'action': 'right', 'power': 0.1, 'time': 1665341022.4804}
```

Figure 30 - Output from the capture reading test

From the previous Figure of the output data, we can see that it was possible to capture a readout of all four directions (with the ones identified as "push" and "pull" being, respectively, forward and backwards).

5.2.3 Combine the prototype's control with the capture readout from the BCI headset.

Having developed all the previous tests, it’s possible for us to continue with the development of the final prototype.

As the integration of the previously tested software, was made with the creation of classes, the software now can be deployed in a more dynamic way, while also making easier to understand to the developer and whoever reads the source-code in the future.

Following the already developed programs, the final prototype will include all of the previously mentioned features, as well as a few others for prototype debugging.

It's also important to remember that the program currently in development will use the Semi-direct approach, which eliminates the need for the user to interact with the product or its user interface. Because it is a "plug and play" solution, the user only needs to turn on the prototype and wait a few moments for the software to load properly.

Beginning with module imports, these were done in such a way that the software would know if any of the requirements for complete prototype use were not met, revolving the software into debug mode.

```
import sys
from threading import Thread
from PyQt5 import QtCore, QtGui, QtWidgets
from pydispatch import Dispatcher

# Remote Controlled Car
try:
    import Motor
    import ADC
except Exception as e:
    print("Car module import failed")
    print(e)

# Headset
try:
    from LiveAdvance import LiveAdvance
except Exception as e:
    print("Emotiv Headset import failed")
    print(e)
```

Code Snippet 5 - Module imports of the prototype

As we can see, the first imports are generic in nature, granting the software access to the System, multithreaded tasks, the UI generator, and control over the flow of data within it. Following with the module imports for the remote-controlled car, these allow the software to access information about motor movement and battery power. Finalizing with the module imports for the headset so that the software can access the data from the BCI headset.

Afterwards, some custom classes and methods were developed in order to easily send or obtain the required information from the previous imported elements.

The following method will be used to send information to the motor to initiate or stop movement on the wheels.

```
    def callMovement(_leftWheels, _rightWheels):
        try:
            Motor.PWM.setMotorModel(-_leftWheels, -_leftWheels,
                                    -_rightWheels, -_rightWheels)
        except Exception:
            print('Motor not reached')
```

Code Snippet 6 - Call for movement method

The next class is the one that handles all the communication and data coming from the BCI headset. It will load the information to connect to the API from a .txt file holding the credentials and will throw an error if any of the requirements are not met, not allowing the connection to be fulfilled and the device only running in debug mode with manual commands.

```
    class emotivHandler(Thread, Dispatcher):
        _events_ = ['brainData']

        def run(self):
            try:
                print('Headset handler Loading...')

                try:
                    clt = open('./cortexClient.txt', 'r').read()

                    self.c_client_id = clt.split('\n')[0]
                    self.c_client_secret = clt.split('\n')[1]
                    self.c_profile_name = 'ISEP_TMDEI'
                except Exception as e:
                    raise ValueError(
                        '\nThere is missing information about the client data.
    \nPlease insert the client id and client secret in a file named
    "cortexClient.txt" respectively in separate lines. Then restart the app')

                self.l = LiveAdvance(self.c_client_id, self.c_client_secret)
                self.l.bind(brainData=self.on_new_brain_data)
                self.l.start(self.c_profile_name)

            except Exception as e:
                print('Headset handler Error')
```

```
                print(e)

        def on_new_brain_data(self, *args, **kwargs):
            _brainValue = kwargs.get('data')
            self.emit('brainData', data=_brainValue)
```

Code Snippet 7 - BCI headset handler class

The following class retrieves the information from the battery providing the debug interface with the battery percentage of the device.

```
    class getBattery(Thread, Dispatcher):
        _events_ = ['batData']

        def run(self):
            try:
                _adc = ADC.Adc()
                ADC_Power = _adc.recvADC(2)*3
                batValue = str(int((float(ADC_Power)-7)/1.40*100))+'%'
                self.emit('batData', data=batValue)

            except Exception as e:
                print('Battery not reached')
                print(e)
                batValue = 'N/A'
                self.emit('batData', data=batValue)
```

Code Snippet 8 - Battery information class

Moving on from the import dependent classes, we arrive at the main classes, where the UI for the debug interface is created and the necessary functions for movement and communication between the headset and the prototype are inserted. Because the source-code is quite extensive, only a few sections will be highlighted because they are important to the prototype's successful operation.

```
class Ui_Client(object):
    def setupUi(self, Client):
    (...)
    self._brainHandler = emotivHandler()
```

```
    self._brainHandler.bind(brainData=self.on_new_brain_data)
    self._brainHandler.start()


    def on_new_brain_data(self, *args, **kwargs):
        self._brainValue = kwargs.get('data')
        self.brnShow.setText('Direction:' + str(self._brainValue['action']))

        if self._brainInput and self._brainValue['action'] != self._lastMovement:
            self._lastMovement = self._brainValue['action']

            if self._brainValue['action'] == 'push':
                self._leftWheels = 1000
                self._rightWheels = 1000
            elif self._brainValue['action'] == 'pull':
                self._leftWheels = -1000
                self._rightWheels = -1000
            elif self._brainValue['action'] == 'left':
                self._leftWheels = -1500
                self._rightWheels = 2000
            elif self._brainValue['action'] == 'right':
                self._leftWheels = 2000
                self._rightWheels = -1500
            elif self._brainValue['action'] == 'neutral':
                self._leftWheels = 0
                self._rightWheels = 0

            callMovement(self._leftWheels, self._rightWheels)
```

Code Snippet 9 - Main class that allows the data compiled from the headset to reach the motor function calling for movement

As we can see, the "on_new_brain_data" method, which is bound to the "emotivHandler" class, enables us to establish a layer of communication between all the data gathered and compiled from the BCI headset and to trigger movement in response to the user's thoughts of an action.

As previously stated, manual control could also be utilized to remove the prototype of any situation in case of a problem.

```
        def keyPressEvent(self, event):
            if self._keyboardInput:
                if not event.isAutoRepeat():
                    if event.key() == QtCore.Qt.Key_W:
                        self._leftWheels = 1000
                        self._rightWheels = 1000
                    elif event.key() == QtCore.Qt.Key_S:
                        self._leftWheels = -1000
                        self._rightWheels = -1000
                    elif event.key() == QtCore.Qt.Key_A:
                        self._leftWheels = -1500
                        self._rightWheels = 2000
                    elif event.key() == QtCore.Qt.Key_D:
                        self._leftWheels = 2000
                        self._rightWheels = -1500

                callMovement(self._leftWheels, self._rightWheels)

        def keyReleaseEvent(self, event):
            if self._keyboardInput:
                if not event.isAutoRepeat():
                    if (event.key() == QtCore.Qt.Key_W
                        or event.key() == QtCore.Qt.Key_S
                        or event.key() == QtCore.Qt.Key_A
                            or event.key() == QtCore.Qt.Key_D):
                        self._leftWheels = 0
                        self._rightWheels = 0

                    callMovement(self._leftWheels, self._rightWheels)
```

Code Snippet 10 - Manual control methods

During the development of the software, the prototype was intensely tested by the developer allowing him to present a finalized solution. These tests included training himself on how to control the device's movement, attempting to train the device on how to better read the developer's thoughts, and when necessary, he used the manual controls. The following Figure displays a preview of the debug interface provided to the developer in case an error occurs.

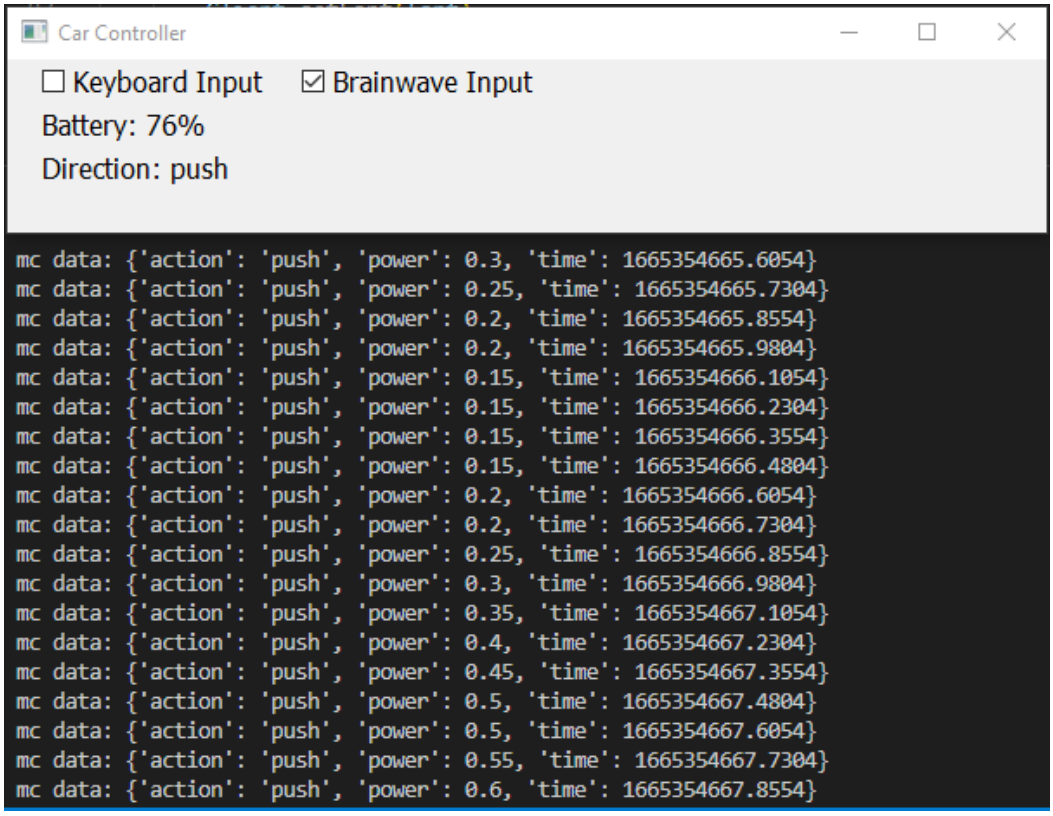


Figure 31 - Preview of the debug interface

6 Testing of the prototype

This chapter will cover how the solution would be tested and evaluated, documenting the results and analyzing their data.

6.1 Solution assessment

Knowing that the main purpose of this project is to help people who suffer from a mobility disability, like mentioned before, the tests aim to guarantee that the solution is functional, natural and swift on its response speeds.

These tests would primarily follow a Checklist-based testing model, where the software testing would be based on a "pre-planned 'to-do' list of tasks" (QATestLab, 2018), testing for the primary movement functions of the device; the system elements, like the structure of the software – for example, if the main function would work from the very beginning without problems.

The tests would be taken on with a group of random people willing to sign up to the testing stage.

Afterwards, after each subject had taken the test, they would be provided with a survey, where their answers would bring to the final analysis the ability to determine if this solution was well implemented and what could be better implemented in order to make the software more usable.

6.2 Tests description

As was already mentioned, the planned tests would mainly employ a checklist-based testing model, followed by a questionnaire that the subject would complete at the end of the test course.

It is worth highlighting that, during the development process, the testing performed with the Brain-Computer Interface headset became more effective when the developer had wet hair, allowing the leads/sensors to collect a clearer readout.

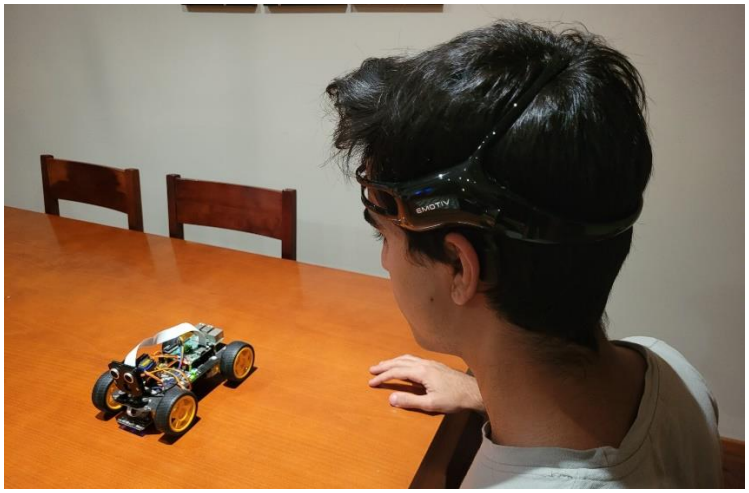


Figure 32- Developer testing the device

The subjects would be anonymous, although, as baseline questions, their gender and age would be documented for later analysis. The subjects would be given some time to comprehend the device and examine it on their own before the supervised tests started. These supervised tests would be divided into two parts, one where the subject would have dry hair and another where they would have wet hair, and each part would contain the Checklist test and the questionnaire that the subjects would have to answer afterwards.

The same questions would be asked in both sections to create comparison graphs for later analysis and to ascertain whether this technology is practical for widespread use by those who need it.

This survey, created with Google Forms, would include a first section for identifying the subject's gender and age group, followed by two sets of two sections each. The first section was to be filled out by the developer while the subject is tested, and the second to be answered by the subject after that round of testing. Attachment B provides a clearer vision of the created survey.

These subjects do not have motor difficulties, so it was only intended to evaluate the correct performance of the intended tasks, in terms of movement of the equipment, as these users do not belong to the target audience of the product.

On the questionnaire part of this survey, the subject would be presented with the following questions:

1. How much time did it took for the software to first start reacting to your thoughts? (Multiple choice)
    a. Under 5 minutes;
    b. Between 5 and 19 minutes;
    c. Between 20 minutes and 1 hour;
    d. Over 1 hour / Not Completed.

2. How much time did it took for the software to react to all 4 commands? (Multiple choice)
    a. Under 20 minutes;
    b. Between 20 and 44 minutes;
    c. Between 45 and 90 minutes;
    d. Over 90 minutes / Not Completed.
3. Could you switch from action to action (Forward, Backwards, Right, Left) repeatedly? (Multiple choice)
    a. Yes;
    b. No;
    c. With difficulty.

4. Did you experience any misreadings? (Linear scale - 1 to 5)
    a. 1 – “No misreadings were captured”
    b. 5 – “A lot of misreadings were captured”.

After being confronted, ten people agreed to assist with the testing phase, providing us with a large enough group to conduct a preliminary study of the developed prototype. In addition, an eleventh person was picked, who, given her age and some of her physical characteristics, was considered to be useful for the current investigation. It's worth noting that the answer of the eleventh person will be highlighted throughout the graphs presented later in the article.

The subjects will also have a final section where, if they like, they can provide comments or leave feedback regarding their use of the device.

6.3 Analysis of subjects' answers

After the subjects complete the survey, the results will be analyzed to determine the efficacy and viability of the prototype. The analysis would also compare the stages of dry and wet hair to determine the optimal situation for using the software and, subsequently, mobility aid, such as the one being tested.

It is important to remember that during the development stage, wet hair produced far superior results because it allowed the leads and sensors to better capture the necessary brain waves for the rest of the device to function. With that said, the results compiled from the test subjects are expected to show an improvement, if not a significant one, throughout the second stage of testing.

6.3.1 Baseline questions

Starting with the baseline questions, we can see that the test group was well divided in terms of gender (Figure 33) as well as age groups, ranging from 15 to 64 years old, with our eleventh subject being female over the age of 64 (Figure 34).

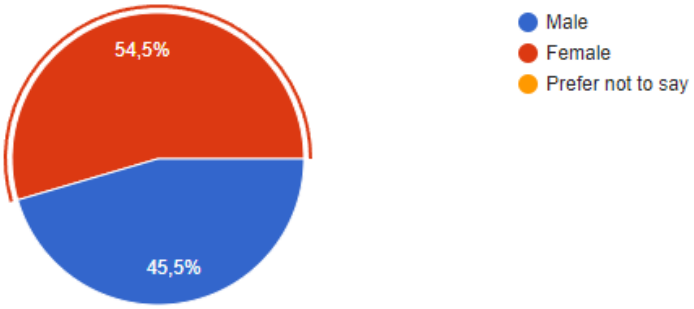


Figure 33 - Survey answer – Gender of the subjects

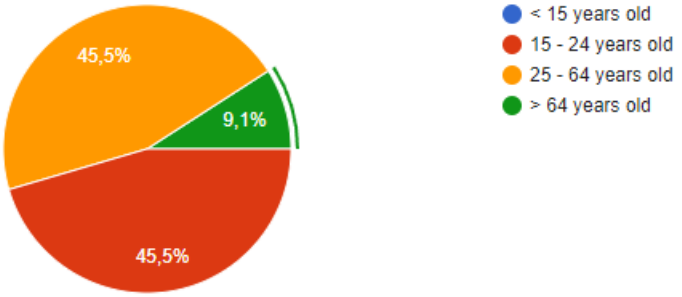


Figure 34 - Survey answer – Age group of the subjects

6.3.2 Side by side comparison – Tasks to perform checklist

Continuing with a side-by-side comparison of the dry and wet hair states, respectively, the next graphs display how the subjects performed during the testing of the prototype.

The following graphs are preceded by a legend indicating whether the subjects completed an action successfully, with difficulty, or not at all.

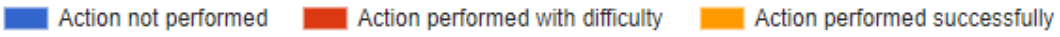


Figure 35 - Legend for Task to perform checklist graphs

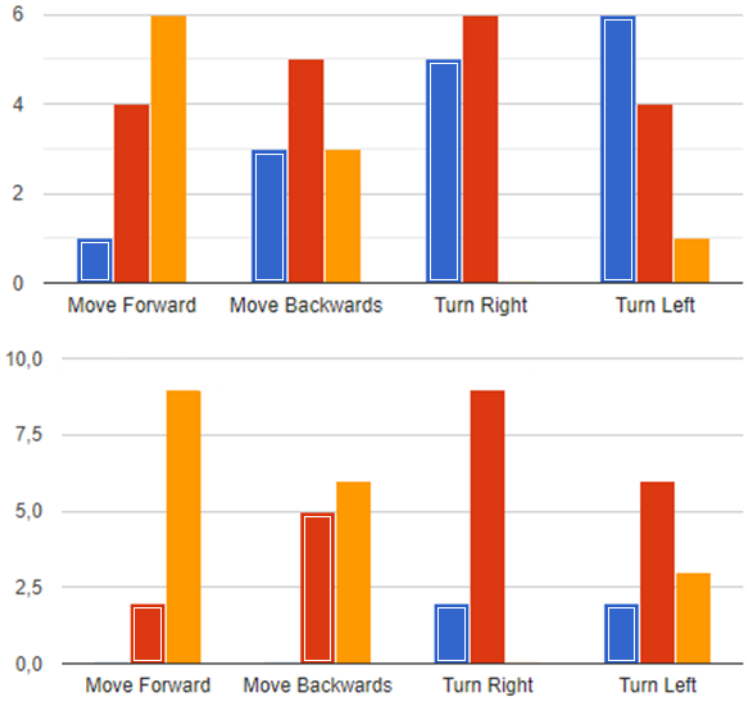


Figure 36 - Tasks to perform graph (dry hair - top | wet hair - bottom)

As we can observe from the previous graphs, there is a visible improvement in the tasks between the two states, particularly in the forward and backward tasks, which are easier to visualize because these are actions that humans perform on a daily basis. It is also interesting to note that the forward and backwards tasks were easily achieved by the subjects once they started walking themselves alongside the prototype. This can be explained by the presence of visual and physical input, which aids the software in understanding that movement has occurred.

In the wet hair state, we get an almost perfect result on the moving forward task but fall short due to two of our subjects possessing long or thick hair that wouldn't allow the leads to capture the necessary data correctly.

Regarding the turning tasks, although small, there were some noticeable improvements from not performing them to performing them but with difficulty. Even if the subjects followed along with the prototype, the result of these actions would remain consistent between the two stages of testing.

6.3.3 Side by side comparison – Questionnaire

Following the conclusion of the Tasks to Perform checklist, a survey was presented to the subjects after each state, in which they answered questions about how they performed on the test in general.

The answers will be presented as a side-by-side comparison, as was the norm in the previous segment, while reiterating that the answers given after the wet hair tests - graphs on the right side - are expected to be an improvement over the answers given after the dry hair tests - graphs on the left side.

It is worth mentioning that, due to time constraints, both of these tests were conducted on the same day, one after the other, and were influenced not only by the subject's wet hair state, but also by the learned experience on how to easily control the prototype.

6.3.3.1 How much time did it took for the software to first start reacting to your thoughts?

Beginning with the first question, the subject was asked how long it took the software to understand the first command it was given to follow. Before taking these tests, the subject was given time with the device and instructed on how to think in order to move it.

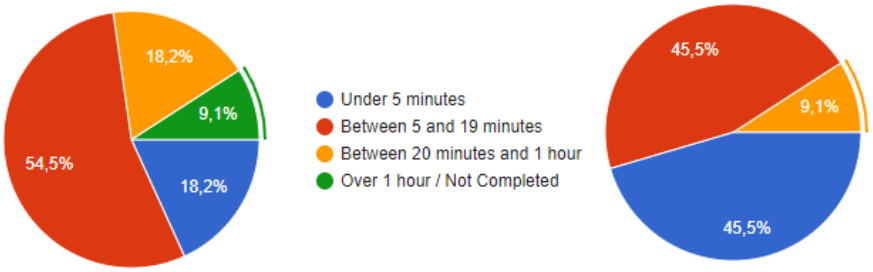


Figure 37 - Answers to the first question (dry hair - left | wet hair - right)

As we can see, most of the subjects were able to generate movement with one command with dry hair, which led to them all doing it with wet hair. We can also observe that the first ten candidates improved overall, with two of them only being able to perform the first action after the first twenty minutes, with dry hair, to all of them performing it under twenty minutes and half of them performing it under five, with wet hair.

6.3.3.2 How much time did it took for the software to react to all 4 commands?

Continuing with the second question, the subject was then asked how long it took for the software to recognize all four commands.

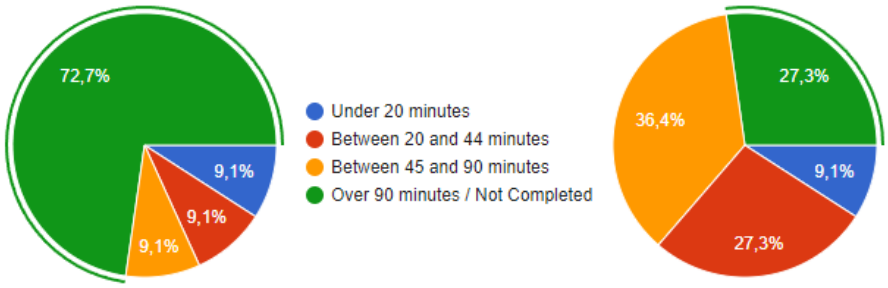


Figure 38 - Answers to the second question (dry hair - left | wet hair - right)

Observing the result, we can acknowledge that the wide majority was not able to enact all four commands, but during the second stage of testing, with wet hair, that number dropped down to almost a quarter of the subjects.

6.3.3.3 Could you switch from action to action (Forward, Backwards, Right, Left) repeatedly?

The subjects were then asked if they could interchange the commands at will, with half of them being unable to do so on the first stage, rising to more than half being able to accomplish this with difficulty, and even one successfully performing the command switches, on the second stage of testing.

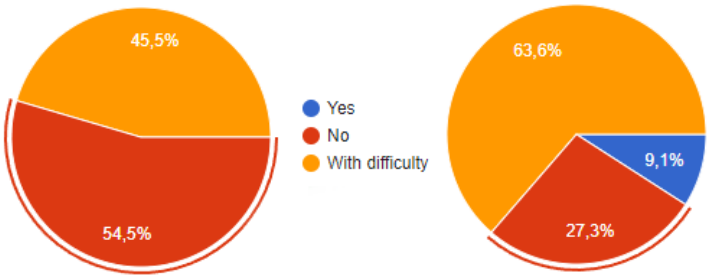


Figure 39 - Answers to the third question (dry hair - left | wet hair - right)

6.3.3.4 Did you experience any misreadings?

As the survey's final question, the subject was asked if they had experienced any misreadings while controlling the prototype, to which the developer had to expand on the question to whether their way of thinking would command the prototype continuously or intermittently.

The scale would range from 1 to 5, with 1 indicating that no misreadings occurred and 5 indicating that a large number of misreadings had occurred during testing.

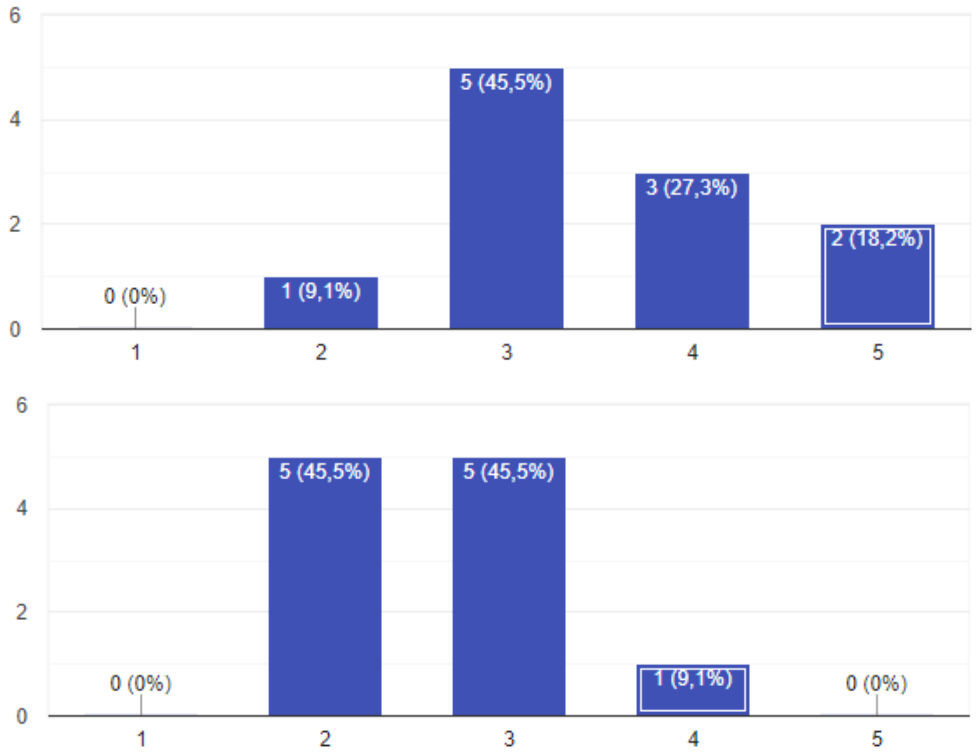


Figure 40 - Answers to the fourth question (dry hair - top | wet hair - bottom)

As we can see, the likelihood of misreadings decreased from one stage to the next, indicating that wet hair would certainly be a significant factor in the prototype's effectiveness.

Given that the subjects approached ranged from bald to long haired, this survey attempted to assess whether the use of this type of prototype, given enough time, is possible by anyone, contributing to the possibility of other people who suffer from a mobility deficiency becoming able to test and use a similar device in the future.

6.4 Observations

After finishing the development and compiling the responses from the previous survey, some accomplishments should be noted.

Despite the semi-direct approach, which could cause some delays, latency, and raise some concerns about the connections and security between devices, the prototype's responsiveness to commands from the headset was outstanding.

Although unable to confirm it with complete certainty, the use of wet hair greatly increased the user's chances of success when performing a task. Having short hair also benefited to accomplish the task effectively.

To conclude, a field for feedback was also provided during the survey for the subject to include any comments they might have about their experience. Some of their comments include the following statements:

- “The RC kart is very sensitive capturing my thinking. In the beginning, after being able to make it move, the excitement distracts me and the car stops moving”;
- “Hard to control even with guidance on what to think”;
- “Very inconsistent until I wet my hair and even after it was quite difficult to control”;
- “Only after I was taught how to think, was when I started better controlling the car”;
- “Took a while to perform the turning left and right tasks but overall, it's fairly easy to control”.

7 Conclusion and Future Work

7.1 Conclusion

The Brain-Computer Interface field is a thriving research field where humans still have a lot to explore. From simple interfaces to control the movement of a remote-controlled car to performing real work on a large scale without the need of leaving a chair, this topic will produce great studies in the near future while enhancing the quality of life for many individuals who require it.

The current project aimed to develop a safe, effective, and versatile solution to assist those suffering from any sort of mobility disability, including but not limited to paralysis or the absence of limbs due to an accident or birth defect.

The developed prototype was able to achieve the desired outcome, albeit with a few unexpected shortcomings, such as the need for the user to maintain wet hair when using the device in order for the signal to be as reliable as possible. The testing group, which included people of all genders and a wide range of ages, succeeded to achieve the intended behavior, indicating that this type of solution is indeed feasible and achievable given enough time.

7.2 Future work

In regards to future work, it is planned to continue the development of the existing prototype, improving on the current result in order to maybe obtain software that could eventually be similar to the previously described "Dynamic approach" or even the "Complete solution".

This would include evolving into a phase of scientific writing where we may also be able to contact some mobility disability groups willing to experiment with this prototype, for example, through the Portuguese National Institute of Rehabilitation, giving us with additional relevant and significant data to compile and assess.

References

ABC, 2014. *Breaking out of locked-in syndrome with Richard Marsh.* [Online] Available at: https://www.abc.net.au/radionational/programs/archived/bodysphere/breaking-out-of-locked-in-syndrome-with-richard-marsh/5695088
[Accessed 5 February 2022].

Abdulkader, S. N., Atia, A. & Mostafa, M.-S. M., 2014. *Brain computer interfacing: Applications and challenges,* Cairo, Egypt: Cairo University.

Alonso-Valerdi, L. M. & Sepulveda, F., 2011. *Python in Brain-Computer Interfaces (BCI): Development of a BCI based on Motor imagery,* Colchester, Essex: University of Essex.

Amelia Hill, The Guardian, 2012. *Locked-in syndrome: rare survivor Richard Marsh recounts his ordeal.* [Online]
Available at: https://www.theguardian.com/world/2012/aug/07/locked-in-syndrome-richard-marsh
[Accessed 5 February 2022].

Anon., 2018. *Facts About Limb Loss | Shirley Ryan AbilityLab.* [Online] Available at: https://www.sralab.org/research/labs/bionic-medicine/news/facts-about-limb-loss
[Accessed 27 December 2021].

Anon., 2018. *Locked In Syndrome - NORD.* [Online] Available at: https://rarediseases.org/rare-diseases/locked-in-syndrome/
[Accessed 31 January 2022].

Cardiff University, n.d. *Brain scanners - Research - Cardiff University.* [Online] Available at: https://www.cardiff.ac.uk/research/explore/research-facilities/brain-scanners
[Accessed 8 February 2022].

Destexhe, A. & Bedard, C., 2013. *Local field potential.* [Online] Available at: http://www.scholarpedia.org/article/Local_field_potential
[Accessed 4 February 2022].

Emotiv, 2020. *Are EMOTIV products medical devices?.* [Online] Available at: https://www.emotiv.com/knowledge-base/are-emotiv-products-medical-devices/
[Accessed 9 February 2022].

EMOTIV, n.d. *Developers - EMOTIV.* [Online] Available at: https://www.emotiv.com/developer/
[Accessed 15 February 2022].

Emotiv, n.d. *EMOTIV EPOC X - 14 Channel Wireless EEG Headset - EMOTIV.* [Online] Available at: https://www.emotiv.com/epoc-x/
[Accessed 12 February 2022].

EMOTIV, n.d. *Getting Started - Cortex API.* [Online] Available at: https://emotiv.gitbook.io/cortex-api/
[Accessed 12 August 2022].

Emotiv, n.d. *Insight Brainwear® 5 Channel Wireless EEG Headset - EMOTIV.* [Online] Available at: https://www.emotiv.com/insight/
[Accessed 12 February 2022].

Freenove - through Github, n.d. *Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi: Apply to FNK0043.* [Online] Available at: https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi
[Accessed 15 April 2022].

Gu, X. et al., 2020. *EEG-based Brain-Computer Interfaces (BCIs): A Survey of Recent Studies on Signal Sensing Technologies and Computational Intelligence Approaches and their Applications,* s.l.: s.n.

InteraXon, n.d. *Compare Muse Products.* [Online] Available at: https://choosemuse.com/compare/
[Accessed 12 February 2022].

InteraXon, n.d. *Muse 2: Brain Sensing Headband.* [Online] Available at: https://choosemuse.com/muse-2-guided-bundle/
[Accessed 13 February 2022].

InteraXon, n.d. *Muse S: Brain Sensing Headband.* [Online] Available at: https://choosemuse.com/muse-s/ [Accessed 13 February 2022].

Javaid, M. A., n.d. *BRAIN–COMPUTER INTERFACE,* Online: Nexus Academic Publishers.

Kotowski, K., Stapor, K., Leski, J. & Kotas, M., n.d. *Validation of EMOTIV EPOC+ for extracting ERP correlates of emotional face processing.* [Online] Available at: https://www.emotiv.com/independent-studies/validation-of-emotiv-epoc-for-extracting-erp-correlates-of-emotional-face-processing/ [Accessed 8 February 2022].

Market Research Future, 2020. *Medgadget.* [Online] Available at: computer-interface-market-2020-worldwide-overview-by-size-share-segments-emerging-technology-growth-leading-players-application-and-regional-trends-by-forecast-2024.html [Accessed 10 February 2022].

MindtecStore, n.d. *InteraXon Muse 2 EEG Headset.* [Online] Available at: https://www.mindtecstore.com/InteraXon-Muse-2-EEG-Headset_1 [Accessed 12 February 2022].

Nagel, S., 2019. *Towards a home-use BCI: fast asynchronous control and robust non-control state detection,* s.l.: s.n.

Neuralink, 2019. *Neuralink Launch Event.* [Online] Available at: https://youtu.be/r-vbh3t7WVI [Accessed 9 February 2022].

Neuralink, 2021. *Approach - Neuralink.* [Online] Available at: https://neuralink.com/approach/ [Accessed 30 January 2022].

NeuroTechEdu, n.d. *Intro to Brain Computer Interface.* [Online] Available at: http://learn.neurotechedu.com/introtobci/ [Accessed 8 February 2022].

Nguyen, T., Khosravi, A., Creighton, D. & Nahavandi, S., 2015. Fuzzy system with tabu search learning for classification of motor imagery data. *Biomedical Signal Processing and Control,* p. 10.

OpenBCI, n.d. *EEG Electrode Cap Kit - OpenBCI Online Store.* [Online] Available at: https://shop.openbci.com/collections/frontpage/products/openbci-eeg-electrocap
[Accessed 12 February 2022].

OpenBCI, n.d. *Electrode Cap Getting Started Guide | OpenBCI Documentation.* [Online] Available at: https://docs.openbci.com/AddOns/Headwear/ElectrodeCap/
[Accessed 12 February 2022].

Ortiz-Rosario, A. & Adeli, H., 2013. *Brain-computer interface technologies: from signal to action,* s.l.: s.n.

Python, n.d. *Modules - Python Documentation.* [Online] Available at: https://docs.python.org/3/tutorial/modules.html
[Accessed 15 February 2022].

Python, n.d. *Welcome to Python.org.* [Online] Available at: https://www.python.org/
[Accessed 15 February 2022].

QATestLab, 2018. *What Is Checklist-Based Testing?.* [Online] Available at: https://qatestlab.com/resources/knowledge-center/checklist-based/
[Accessed 18 Februay 2022].

Red Hat, 2017. *What is an API?.* [Online] Available at: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces
[Accessed 15 February 2022].

Riverbank Computing Limited, n.d. *PyQt5 · PyPI.* [Online] Available at: https://pypi.org/project/PyQt5/
[Accessed 16 April 2022].

Shih, J. J., Krusienski, D. J. & Wolpaw, J. R., 2012. *Brain-Computer Interfaces in Medicine,* Jacksonville, FL: Mayo Foundation for Medical Education and Research.

Smith, E. & Delargy, M., 2005. *Locked-in syndrome,* Dun Laoghaire, County Dublin, Ireland: National Rehabilitation Hospital.

The New Atlantis, 2018. *Locked In: What It's Like to Be Fully Paralyzed.* [Online] Available at: https://www.thenewatlantis.com/practicing-medicine/locked-in-what-its-like-to-be-fully
[Accessed 5 February 2022].

Thomas Mallon, The New York Times, 1997. *In the Blink of an Eye.* [Online] Available at: https://archive.nytimes.com/www.nytimes.com/books/97/06/15/reviews/970615.mallon.html?_r=1
[Accessed 5 February 2022].

Waldert, S., 2016. *Invasive vs. Non-Invasive Neuronal Signals for Brain-Machine Interfaces: Will One Prevail?,* London: Sobell Department of Motor Neuroscience and Movement Disorders, University College London Institute of Neurology.

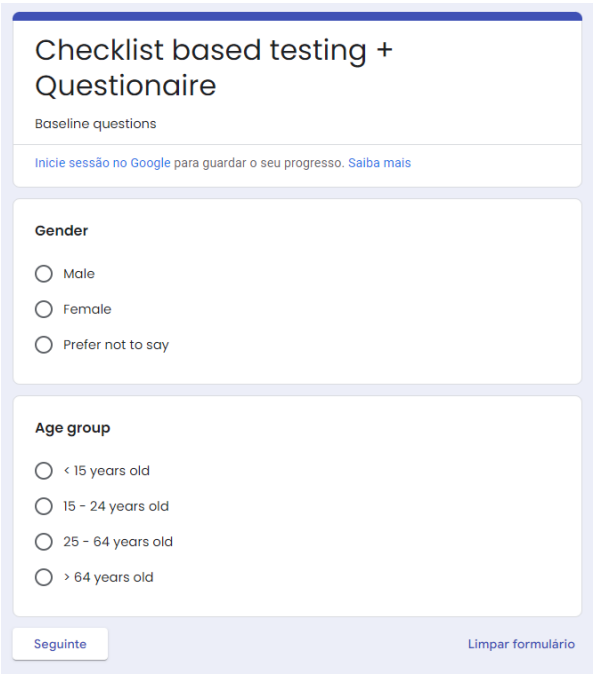
Wareable, n.d. *Muse S review: meditation and sleep wearable is no dream come true.* [Online] Available at: https://www.wareable.com/wearable-tech/muse-s-review-8255
[Accessed 12 February 2022].

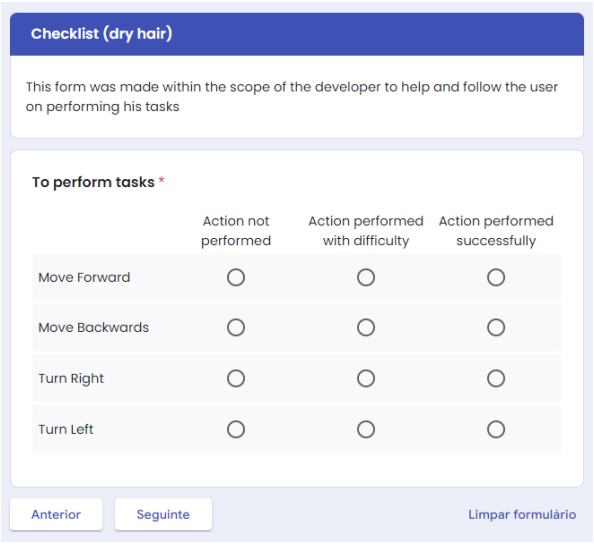
Attachments

Attachment A

All documented software will reside and be available in: https://github.com/BarreiraTom/ISEP-TMDEI-Thesis_Project

Attachment B





Questionnaire (dry hair)

To be filled by the subject

How much time did it took for the software to first start reacting to your thoughts?

- ◯ Under 5 minutes
- ◯ Between 5 and 19 minutes
- ◯ Between 20 minutes and 1 hour
- ◯ Over 1 hour / Not Completed

How much time did it took for the software to react to all 4 commands?

- ◯ Under 20 minutes
- ◯ Between 20 and 44 minutes
- ◯ Between 45 and 90 minutes
- ◯ Over 90 minutes / Not Completed

Could you switch from action to action (Forward, Backwards, Right, Left) repeatedly?

- ◯ Yes
- ◯ No
- ◯ With difficulty

Did you experience any misreadings?

	1	2	3	4	5	
No misreadings were captured	◯	◯	◯	◯	◯	A lot of misreadings were captured

Anterior Seguinte Limpar formulário

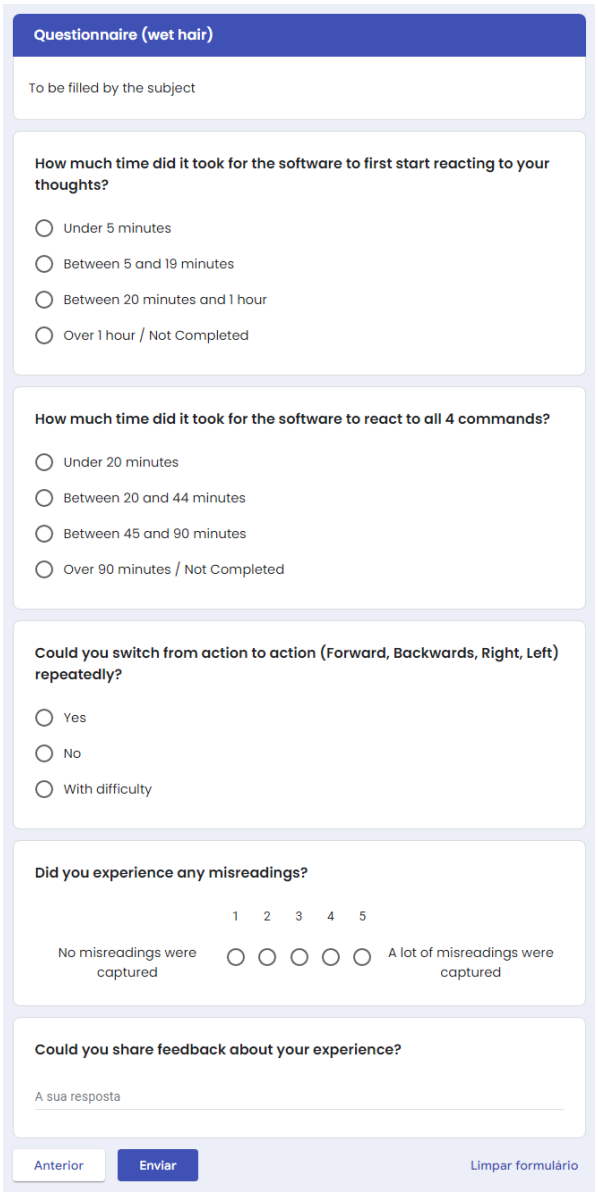
Checklist (wet hair)

This form was made within the scope of the developer to help and follow the user on performing his tasks

To perform tasks *

	Action not performed	Action performed with difficulty	Action performed successfully
Move Forward	◯	◯	◯
Move Backwards	◯	◯	◯
Turn Right	◯	◯	◯
Turn Left	◯	◯	◯

Anterior Seguinte Limpar formulário



Questionnaire (wet hair)

To be filled by the subject

How much time did it took for the software to first start reacting to your thoughts?

- ◯ Under 5 minutes
- ◯ Between 5 and 19 minutes
- ◯ Between 20 minutes and 1 hour
- ◯ Over 1 hour / Not Completed

How much time did it took for the software to react to all 4 commands?

- ◯ Under 20 minutes
- ◯ Between 20 and 44 minutes
- ◯ Between 45 and 90 minutes
- ◯ Over 90 minutes / Not Completed

Could you switch from action to action (Forward, Backwards, Right, Left) repeatedly?

- ◯ Yes
- ◯ No
- ◯ With difficulty

Did you experience any misreadings?

	1	2	3	4	5	
No misreadings were captured	◯	◯	◯	◯	◯	A lot of misreadings were captured

Could you share feedback about your experience?

A sua resposta

Anterior | Enviar | Limpar formulário