POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Biomedical Engineering

DEVELOPMENT OF A WEARABLE AND COST-EFFECTIVE BRAIN-COMPUTER INTERFACE

Supervisor: Luca Mainardi

Co-Supervisors: Matteo Matteucci

Emanuele De Bernardi

Thesis of:

Davide Marzorati

Student ID: 851686

Academic Year 2016-2017

Acknowledgments

I'm deeply grateful to my supervisor, professor Luca Mainardi, as well as to my co-supervisors, professor Matteo Matteucci and research assistant Emanuele De Bernardi. Their suggestions, support, and help made it possible to complete this thesis.

I would like to extend my gratitude to professor Hananeh Esmailbeigi from the University of Illinois at Chicago, as she gave me the exciting chance to start working on this project.

Many more people indirectly contributed to this thesis work. I'm very thankful to my entire family. My parents have always supported me: thank you, mom and dad, for every single thing you did for me, and for giving me the chance to live an amazing academic experience in Chicago. As the youngest brother, I thank the best sisters I could have ever dreamed of: thank you Ila, Franci, and Mery. Then, thank you granny Pia, because you always have me in your thoughts. I would also like to thank my uncle and aunt, Aldo and Brunella, for their support.

While studying in Chicago, I was lucky to spend my time with special people. Thank you Ale, Andre, Ele, Jack, Luca, and Marco. In particular, thank you Ele for all the joyful times we had in Oak Park, and Andre for all the talks and lunches we had together.

During my university years at Politecnico di Milano, I had the chance to become friends with many people. I would particularly like to thank Andre, Eddy, Fra, and Lore for their true friendship.

Many other people have always been close to me. I'm not going to list all my friends here, because both it would be too long and the day after submitting this manuscript I would probably realize that I forgot someone. Even if I don't mention all of my friends, I would like to thank each one of them.

Special and important thanks go to Lisa. Thank you for being by my side every day: you believe in me, you support me, and you give me the strength to get through hard times. I don't know what I would do without you.

Abstract

Individuals living with debilitating conditions, caused by brain or spinal cord injuries, amyotrophic lateral sclerosis or other diseases, are no longer able to voluntary control muscles movement. For these individuals, assistive technologies are required to allow, or ease, communication and mobility. One example of assistive technologies are Brain-Computer Interfaces (BCI). In this work, we present the development of a wearable and cost-effective BCI assistive device. The proposed assistive device functions based on the P300 response of the human brain. The device allows the disabled individual to type words on a computer screen or on a mobile device, and also to control a robotic arm via a computer. Therefore, augmenting communication capabilities and allowing for the control of external devices.

The presented assistive device was validated on two healthy subjects. Amongst them, the average maximum information transfer rate for word composition was 12.19 *bits/min*, while for two different modes of robotic arm control it was 9.95 *bits/min*, and 4.09 *bits/min*.

The results show that the performance of our wearable and cost-effective device is comparable to the performance of previously published studies involving clinical grade EEG acquisition systems. Improvements of the device and a validation by disabled users are required to complete the transition from laboratory setting to in-home use, allowing many disabled individuals to improve their quality of life.

Sommario

Introduzione

Ogni anno, nel mondo, tra le 250.000 e le 500.000 persone subiscono una lesione a livello della spina dorsale. Questo tipo di lesione può portare a numerose condizioni debilitanti, tra cui la tetraplegia, paralisi di arti inferiori e superiori. Oltre a una lesione a livello della spina dorsale, esistono numerose altre malattie che possono intaccare i percorsi neurali utilizzati dal cervello per comunicare con i muscoli periferici del corpo. Alcuni esempi di queste malattie sono: sclerosi multipla, ictus cerebrale e, in generale, lesioni a livello del cervello. Gli individui affetti da queste malattie possono esserne colpiti con gradi diversi di gravità. I più gravi non sono più in grado di controllare i muscoli in modo volontario e, di conseguenza, non possono comunicare con altre persone, esprimere i propri sentimenti ed utilizzare qualsiasi tipo di dispositivo in modo autonomo. Per permettere, o facilitare, la mobilità e la comunicazione di queste persone, sono necessari dispositivi di tecnologia assistiva.

Un esempio di tecnologia assistiva è dato dalle Interfacce Cervello Computer (Brain Computer Interfaces, BCI). Una definizione di BCI è data da *Wolpaw et al.*: "Una BCI è un sistema di comunicazione in cui i messaggi o i comandi che un individuo invia verso il mondo esterno non passano attraverso i normali percorsi neurali del cervello, costituiti dai nervi e dai muscoli periferici". Di conseguenza, un individuo che utilizza una BCI è in grado di mantenere un'interazione con il mondo esterno, anche se i suoi percorsi neurali sono fortemente danneggiati.

Per poter funzionare, le BCI hanno bisogno di monitorare l'attività cerebrale. Questo tipo di monitoraggio può essere ottenuto con tecniche di tipo invasivo o non-invasivo. Le tecniche di tipo invasivo richiedono delle operazioni chirurgiche per il posizionamento di elettrodi sulla corteccia cerebrale. Al contrario, quelle di tipo non invasivo non richiedono alcun tipo di operazione chirurgica. Una tecnica non invasiva per il monitoraggio dell'attività cerebrale è l'analisi del segnale elettroencefalografico (EEG) generato dall'attività elettrica del cervello. Questo segnale

può essere registrato in maniera totalmente non invasiva con degli elettrodi posizionati sullo scalpo.

La ricerca nel campo delle BCI ha ormai una lunga storia, ma non si è ancora completata la transizione dall'utilizzo delle BCI all'interno di laboratori a quello in un contesto domestico. La realizzazione di BCI con hardware portatile e a basso costo può aiutare il completamento di questa transizione.

Scopo del lavoro

L'obiettivo di questo progetto è lo sviluppo di una BCI wearable e a basso costo basata sull'analisi del segnale EEG che permetta all'utente di comporre parole su di un monitor e di controllare un braccio robotico.

I soggetti colpiti da lesioni a livello del sistema nervoso, non possedendo controllo volontario dei muscoli, non sono in grado di esprimere i propri sentimenti verso il mondo esterno e di controllare in modo autonomo qualsiasi tipo di dispositivo. Per questo motivo, fornire loro una possibilità di interagire con altre persone e con il mondo esterno può portare a un notevole miglioramento della loro qualità di vita. Inoltre, la realizzazione di un sistema affidabile e dal costo accessibile fa sì che esso possa essere utilizzato dalla maggior parte delle persone che ne hanno bisogno.

Materiali e metodi

Hardware

Oggi, grazie al progresso della tecnologia, la strumentazione necessaria per l'acquisizione di segnali biologici non è più così costosa come lo era in passato. Di conseguenza, è possibile ottenere dei segnali di buona qualità senza utilizzare della tecnologia dal costo elevato.

La BCI proposta in questo lavoro è basata sulla scheda di acquisizione OpenBCI Cyton. Questa scheda, oltre ad essere open-source e a basso costo, è stata progettata per l'acquisizione di segnali biologici. Un supporto stampato in 3D consente di mantenere in posizione gli elettrodi a secco utilizzati per l'acquisizione del segnale EEG. Nell'ambito di questo progetto, sono state apportate delle modifiche alla board, sia a livello hardware che a livello firmware, per l'invio di pacchetti di dati a qualsiasi dispositivo dotato di modulo Bluetooth.

Software

Diversi tipi di risposte cerebrali possono essere utilizzate per il funzionamento di una BCI. Nella BCI proposta in questo lavoro la risposta cerebrale utilizzata è l'onda P300, un potenziale evento relato del cervello. In generale, un potenziale evento relato si genera quando a un individuo viene presentato uno stimolo visivo, uditivo o somato-sensoriale. L'onda P300, in particolare, si genera quando uno stimolo, che possiede un contenuto significativo per l'individuo, viene presentato in modo infrequente e intervallato da stimoli non significativi. La risposta P300 è osservabile in un tracciato EEG come un picco positivo a 300 ms dall'istante temporale di presentazione dello stimolo. Dal momento che l'onda P300 è una risposta innata del cervello umano, all'utente non è richiesto alcun tipo di processo di apprendimento per poter utilizzare la BCI.

Un software è stato sviluppato utilizzando il linguaggio di programmazione Processing per svolgere le seguenti funzioni: stimolazione visiva dell'utente finalizzata alla generazione dell'onda P300; lettura del segnale EEG inviato dalla scheda OpenBCI Cyton; elaborazione del segnale EEG; feedback visivo e controllo del braccio robotico in seguito all'elaborazione del segnale EEG e all'identificazione delle onde P300.

Inoltre, al fine di avere una BCI completamente portable, è stata sviluppata un'applicazione per dispositivi mobile con sistema operativo Android. Questa applicazione permette all'utente di utilizzare la BCI in qualsiasi luogo si trovi, senza dover avere sempre a disposizione un computer. L'unico requisito necessario per avere un'applicazione funzionante è che il dispositivo mobile sia dotato di un modulo Bluetooth. Le funzioni che l'applicazione Android può svolgere sono le stesse svolte dal software descritto in precedenza, ad eccezione del controllo del braccio robotico.

Identificazione dell'onda P300

L'identificazione delle onde P300 presenti nel segnale EEG rappresenta una componente fondamentale di ogni sistema BCI basato sull'onda P300. Per poter identificare le onde P300 che si generano a seguito della stimolazione dell'utente, è necessario effettuare una classificazione del segnale EEG. In questo lavoro, per la classificazione del segnale EEG e l'identificazione delle onde P300 è stata utilizzata la tecnica della regressione logistica.

Al fine di classificare il segnale EEG applicando la tecnica della regressione logistica, è essenziale stabilire quali sono le caratteristiche del segnale da tenere in considerazione. Effettuare questo processo di selezione in modo manuale può portare ad un risultato sub-ottimale. Per tale motivo, nella BCI sviluppata in questo progetto è stato utilizzato un algoritmo genetico con lo scopo di effettuare una selezione automatica delle caratteristiche del segnale da utilizzare nel processo di identificazione delle onde P300.

Risultati

Per verificare la funzionalità della BCI sviluppata in questo progetto, sono stati effettuati dei test sperimentali su due soggetti sani. Il valore medio tra i due soggetti di massimo trasferimento di informazione ottenuto nella composizione di parole è stato pari a 12.19 *bits/min*, mentre per quanto riguarda le due diverse modalità di controllo del braccio robotico è stato di 9.95 *bits/min* e di 4.09 *bits/min*.

Conclusioni

I risultati ottenuti in questo lavoro sono confrontabili con quelli ottenuti in altri studi relativi a BCI basate su onda P300, in cui però è stata utilizzata della strumentazione clinica di alta qualità per l'acquisizione del segnale EEG.

In conclusione, il dispositivo descritto in questo lavoro prova che sistemi di acquisizione di segnali biologici a basso costo, uniti a tecniche robuste di classificazione del segnale, possono essere utilizzati per lo sviluppo di dispositivi di tecnologia assistiva portatili e wearable. Miglioramenti del dispositivo e una validazione da parte di soggetti disabili permetteranno, in futuro, di completare la transizione dall'utilizzo del dispositivo all'interno di un laboratorio all'utilizzo in un ambiente domestico: ciò consentirà di migliorare notevolmente la qualità di vita di soggetti disabili.

Contents

List of Figures

List of Tables

List of abbreviations

BCI	Brain Computer Interface
EEG	Electroencephalography
ERP	Event Related Potential
VEP	Visual Evoked Potentials
DoF	Degrees of Freedom
GUI	Graphical User Interface
GA	Genetic Algorithm
ITR	Information Transfer Rate

Chapter 1

Introduction

Annually, across the world 250,000 to 500,000 people suffer an injury at the level of the spinal cord [1]. These injuries can lead to a variety of debilitating conditions, such as tetraplegia, which is a paralysis of all the four limbs. Other than spinal cord injury, there are several disorders which may affect the neural paths normally used by the brain to communicate with peripheral muscles of the body, and some examples of these disorders are multiple sclerosis, stroke at the level of the brainstem and brain injuries in general [2]. Individuals who suffer from these injuries or diseases can be affected with varying degrees of severity. Those who are mostly affected are no longer able to voluntary control muscles, therefore being unable to communicate with others and express their feelings, or use any external device that may help them. For these people, assistive technologies are required to allow, or ease, mobility, communication and domestic activities.

In order to restore the original functions affected by these conditions, it is possible to use a Brain-Computer Interface (BCI). *Wolpaw et al.* define BCI as: "A BCI is a communication system in which messages or commands that an individual sends to the external world do not pass through the brain's normal output pathways of peripheral nerves and muscles" [2]. Therefore, an individual who makes use of a BCI is able to maintain an interaction with the external world even if his/her neural pathways are severely injured.

BCI are based on monitoring the activity of the brain. This goal can be achieved with invasive or non-invasive techniques. Invasive techniques require surgical procedures to place electrodes directly onto the cerebral cortex. Non-invasive techniques, as the name suggests, do not require any surgical and invasive procedure, and are based on the analysis of images of the brain or of the electrical signals generated by the brain. An example of a non-invasive technique for brain activity monitoring is the electroencephalography (EEG) signal, that shows the electrical activity of the brain and that can be recorded with electrodes placed directly on the scalp [3].

The goal of the present work was to develop a wearable and cost-effective BCI system, with the idea of allowing the end-user to spell words on a screen and to control a robotic arm. Building an affordable but, at the same time, reliable system means that it could be used by the majority of the disabled people. Today, thanks to the advances in the technology, EEG equipment is not as expensive as it was years ago, so it is possible to have good quality EEG recording without having to rely on high - level and expensive technology. Tetraplegic subjects have no absolute control of voluntary muscles, so there is no residual movement of the upper limbs. For this reason, providing them an affordable way of interaction with the external environment and communication with people can improve their quality of life.

The BCI proposed in this work is based on the OpenBCI Cyton [4]. The OpenBCI Cyton is an open-source, cost-effective electronic board designed for biological signal acquisition. A 3D printed headset can be used together with the board to house the dry electrodes used for EEG recording. The BCI is designed to function based on the innate P300 response of the human brain, in order to improve the accessibility of the device by eliminating intensive subject training. A P300 response is an event related potential (ERP), elicited when an auditory, visual, or somatosensory stimulus, which is significant for the subject, is presented to him/her infrequently and interlarded with insignificant stimuli. The P300 response is visible in the EEG trace as a positive peak at about 300 ms after the stimulus onset [5].

A software was developed to perform the following tasks: visual stimulation of the user, through flashing symbols on a grid, to elicit a P300 response; retrieval of the EEG signal streamed by the OpenBCI board; processing of the EEG signal; control of a robotic arm and visual feedback to the user after the processing of the EEG signal. Regarding the task of EEG signal processing, online detection of the P300 response was achieved using logistic regression, and a genetic algorithm was used to perform automatic feature extraction.

Furthermore, in order for the device to be completely wearable, an Android application was developed. This application allows to use the device wherever the user wants to, without the need of a laptop. The application can run on any smartphone or tablet that has, among its features, a Bluetooth module. In order to stream data from the OpenBCI board to any Android device, hardware and firmware modifications were applied to the board. The tasks that the Android application can perform are the same as the ones of the developed software, expect for the control of a robotic arm.

In order to assess the functionality of the device, experimental tests were carried out on two subjects. The achieved results are comparable to other BCI studies, in which clinical instrumentation was used for EEG signal acquisition. Our results show that it is possible to obtain comparable results in terms of accuracy and information transfer rate when using cost-effective hardware for EEG acquisition. Therefore, the proposed device serves as a proof of concept that cost-effective EEG acquisition systems along with robust classification techniques could be employed in the development of portable and wearable assistive technology.

The present manuscript is organized in seven chapters. This chapter served as an introduction to the topic of BCI and to define the scope of this thesis work. In Chapter 2 a review of BCI, and of P300 - based BCI, is provided. In Chapter 3, the hardware components of the proposed assistive device are presented. In Chapter 4, the developed computer - based and Android - based BCI are described. In Chapter 5, the techniques used for feature extraction and classification of the EEG signal are shown. In Chapter 6, the results of the experimental studies are presented. In Chapter 7, the results are discussed, and future improvements are proposed.

Chapter 2

Brain-Computer Interfaces

Spinal cord injury, stroke at the level of the brainstem, brain injuries and amyotrophic lateral sclerosis are just a few examples of injuries and diseases that can affect the neural paths normally used by the brain to communicate with peripheral muscles. These injuries and diseases can lead to a variety of debilitating conditions [2]. In order to restore the original functions of nerves and muscles, different options can be exploited:

- the first option would be to increase the functioning of the neural pathways and of the muscles unaffected by the injury or by the disease. If there are muscles that are still under voluntary control, it is possible to use them in order to substitute the injured muscles. *Damper et al.* presented a system for the non-vocal disabled, allowing them to use their hands to answer questions [6] . *Kubota et al.* presented an ocular movement detector that disabled patients can use to communicate [7];

- the second option would be to detour around the breaks in the damaged neural pathways. As an example, individuals who are suffering from a spinal cord injury can use the electromyographic signal generated by muscles, which are controlled by nerves situated above the spinal lesion, to electrically stimulate those muscles which are controlled by nerves situated below the lesion. *Kobetic et al.* developed an hybrid system able conjugate an exoskeletal bracing together with a device for functional electrical stimulation [8];

- the last option would be to provide to the brain a new channel for communication and control, not based on muscular pathways. Brain monitoring techniques, such as the electroencephalography signal (EEG), optical imaging, positron emission tomography (PET) and functional magnetic resonance imaging (fMRI) can serve as a way to analyze the brain activity and detect the user's intent. With these techniques, a Brain-Computer Interface

(BCI) can be created. Using a BCI, the individual would then be able to transmit messages or other commands to the external world, even if neural pathways are damaged. *Chaudhary et al.* developed a BCI for subjects with advanced amyotrophic lateral sclerosis [9]. In this study, the involved subjects, that live without any available mean of communication, managed to learn how to answer some questions, requiring a "yes" or "no" answers, by means of measurements of oxygenation changes in the frontocentral region of the brain. These signals were obtained with functional near-infrared spectroscopy.

Among all the several options that can be used to restore the original functions of the affected nerves and muscles, BCI represents the best choice. Considering all the possible methods that can be used to monitor the brain activity and build a BCI, the EEG signal is the one that has a the shortest time constant and that requires simple and inexpensive acquisition devices. So, this type of brain activity monitoring is the only one who offers the possibility, for the locked-in subject, to use the brain signal to communicate and control devices in an efficient way. *Luo et al.* developed a smart house system with an EEG based BCI [10] . This device makes it possible to open or close curtains, turn light switches on and off, and control the air-conditioning by simply focusing on images flashing on the screen.

In this chapter, we start with a brief description of the anatomy of the cerebrum, and with an explanation of the EEG signal. Later in the chapter, we provide a review of BCI, focusing on BCI functioning based on the P300 event related potential.

2.1 The Cerebrum

The encephalon, that is commonly referred to as the brain, is divided in two main parts: the telencephalon and the diencephalon [11]. These parts have different functions: the telencephalon is responsible of high-level functions, as thinking, planning and storing, while the diencephalon works on low-level functions, as maintaining the homeostasis and controlling the circadian rhythm.

The cerebral cortex represents the outside layer of the telencephalon. In humans, the cerebral cortex is folded, so that it provides a very high amount of surface area in the confined volume of the skull [11]. The cerebral cortex is separated into two cortices by the longitudinal fissure, that divides the brain into the right and left hemispheres.

In the cerebral cortex it is possible to identify areas in which the neurons share the same function [12]: these are generally referred to as Brodmann areas, and the current number of

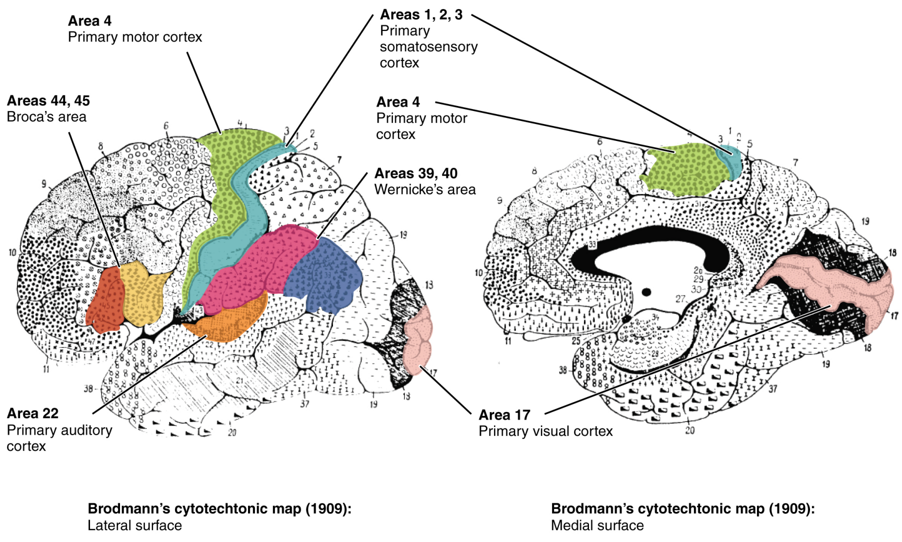


Figure 2.1: Broodman areas of the brain. Image taken from: *OpenStax Anatomy and Physiology*, licensed under CC-BY-4.0.

identified areas is 52. Examples of Brodmann areas, shown in Figure 2.1, are the motor cortex, which is responsible of all the voluntary movements in the body, the sensory cortex, that receives and processes the signals coming from the sensory neurons of the body, and the visual cortex, which receives the signals of the receptors of the retina. The cerebral cortex is mainly made up of gray matter, which is constituted by neuron bodies. The underlying white matter consists of the axons of the neurons. The neurons, characterized by the basic structure shown in Figure 2.2, are connected through long fibers, the axons. The axons, that have their origin in the cell body, represent the output terminals of the neuron. The dendrites, instead, are those parts of the neurons that receive the input from other neurons.

A synapse is a junction present between the end tip of a neuron axon and the dendrites of another neuron. The synapse is used by the neurons to communicate [11]. The neuronal signal coming from the presynaptic cell crosses the junction thanks to the release of chemical substances, the neurotransmitters. The neurotrasmitters, when bonding with the receptors present on the membrane of the postsynaptic cell, cause a variation of the membrane potential of the postsynaptic cell. If the membrane potential goes above a certain threshold, an action potential is generated, and this potential will be transmitted along the axon and will reach other neurons to transmit the neuronal activity. The threshold for the generation of the action potential depends on the type of neuron [11].

Among all the possible methods used to monitor the activity of the cerebrum, two main groups can be identified: methods which measure directly the activity of the brain (for example,

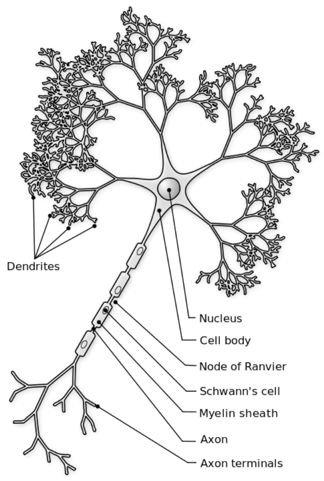


Figure 2.2: Basic structure of a neuron. Licensed under *GNU Free Documentation License 1.2.*

the EEG signal), and methods which estimate the neuronal activity in an indirect way (with a measure of the blood flow, as an example). When using the direct methods, the measurement of the electrical activity of the brain is a nonlinear, spatial and temporal combination of the action potentials of all the present neurons.

2.2 The EEG Signal

The electroencephalographic signal (EEG) was discovered in 1929 by Hans Berger [3]. Since then, it has been used for multiple purposes. Today, its main applications are: diagnoses of diseases (epilepsy, for example); development of BCI; monitoring of the patient status during anesthesia.

In order to record the EEG signal in a noninvasive way, it is possible to use electrodes placed outside the brain and directly on the scalp. These electrodes are able to detect the variation of the electric potential, generated by the presence of action potentials and by the basal activity of the neurons. Electrodes are typically made of materials such as gold or silver, so that their impedance is low (ideally, below 10 kOhm). Electrodes can be also classified according to their being active or passive. Active electrodes are electrodes that feature some internal circuitry that is used to provide higher-quality signals, while passive electrodes do not feature any built-in circuitry. Regarding the positions of the electrodes, a standard was defined: the 10-20 system [13].

The electrodes locations defined in the 10 - 20 system are shown in Figure 2.3. In this standard, four fixed points are defined. The first point is the nasion, which is situated between

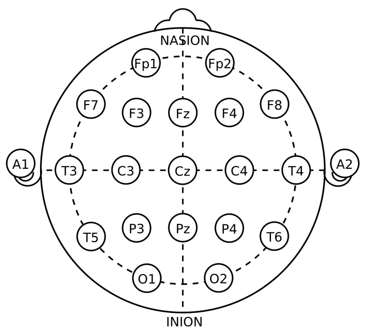


Figure 2.3: Electrodes locations in the 10-20 international standard for EEG recording.

F	Frontal Lobe
C	Central Lobe
P	Parietal Lobe
T	Temporal Lobe
O	Occipital Lobe

Table 2.1: Electrodes identifiers in the 10-20 international standard for EEG recording.

the forehead and the nose. The second point is the inion, which can be located as the lowest point of the skull from the back of the head, and situated along the antero-posterior plane as the nasion. The last two points are the preauricular points, situated in the anterior part of each ear. The number '10' and '20' were chosen because the distance between each electrode is equal to the 10 or 20% of the nasion-inion or right preaurical point - left preaurical point distances on the skull. Each electrode position is identified with a letter and a number. The meaning of each letter is shown in Table 2.1 (note that the letter C, which stands for central lobe, is used only for identification purposes, since there is no central lobe in the brain). To identify electrodes of the right hemisphere, even numbers are used. Left hemisphere electrodes are marked with odd numbers.

With the 10-20 standard, it is possible to identify a total number of 21 electrodes positions, as shown in Figure 2.3. Additional positions can be added by using the 10 - 10 standard, in which the distance between each electrode is always equal to the 10% of the total back-front or left-right distances of the skull.

The aim of EEG is to record the variation of the electric potential on the scalp. In order to do this, it is necessary to have a reference level for these measurements. There are several possible ways to define a reference level. One way is to choose an electrode that acts as a fixed reference for all the other electrodes, and that is typically positioned at the level of the mastoid or on the earlobe: this technique is called *common reference*. Another way is to compute the average value of all the signals recorded from each channel and the subtract this value from the signal recorded at each channel (*average reference*). The reference can also be defined for each couple of electrodes, so that each channel has its own reference (*bipolar configuration*).

2.3 BCI

In this section, brain-computer interfaces (BCI) will be described. Firstly, a formal definition of BCI is provided. Secondly, a classification of the different types of BCI is presented. Thirdly, the main components of a BCI are described. Lastly, non-invasive BCI are introduced.

2.3.1 Definition

Wolpaw et al. define BCI as follows: "A BCI is a communication system in which messages or commands that an individual sends to the external world do not pass through the brain's normal output pathways of peripheral nerves and muscles" [2]. Therefore, an individual who makes use of a BCI is able to maintain an interaction with the external world even if his/her neural pathways are severely injured.

2.3.2 Classification

According to *Wolpaw et al.* [2] BCI can be classified as follows:

- **Dependent and independent BCI**: a dependent BCI doesn't use the normal output pathways of the brain to carry a command or a message. At the same time, to generate the required brain activity it is necessary to have residual activity in these pathways. To understand what a dependent BCI is, consider a BCI based on Visual Evoked Potentials (VEP). To elicit a response from the subject, he is presented with a matrix of flashing letters. Then, he has to gaze at a specific letter in order to have a VEP that is greater when that letter flashes in comparison to the others that are recorded when other letters flash. So, the output channel of the brain is the electric signal recorded with EEG, but its generation still depends on the gaze direction, which is in turn dependent on the eye muscles. For the mentioned reasons, a dependent BCI is another method to detect messages that are present in output paths of the brain (in the described example, the direction of the gaze isn't monitored by observing the position of the eye, but rather by recording the EEG signal). Instead, an independent BCI is completely independent, as the name suggests, on the normal output pathways of the cerebrum, and for this reason no activity in those paths is necessary for the generation of the brain signal of interest. To better understand how an independent BCI works, consider a BCI that is based on the P300 response, and that presents the user with flashing letters. The P300 response is an event related potential, that is elicited when stimuli, which are significant from the subject's point of view, are presented to him/her. When the letter the user is focusing on flashes, a P300 will be elicited. No P300 response will be present when other letters flash. The EEG signal that is measured is not dependent on the gaze direction of the eyes, but rather on the user's intent. As it is possible to understand from the provided examples, independent BCIs are of much more interest, because they provide completely new channels to the brain.

- **BCI employing invasive or non-invasive techniques for brain monitoring:** BCI can also be classified according to the techniques used for brain activity monitoring. An example of a non-invasive technique is the EEG signal, that can be recorded from the scalp without requiring surgery or invasive procedures. Invasive techniques require the placement of matrices of electrodes directly on the brain, and for this reason surgeries are necessary. There are both advantages and disadvantages for the two methods. The most important advantage of non-invasive BCI is that the end-user is not required to undergo surgeries. The disadvantage of non-invasive BCI, which represents an advantage of invasive BCI, is the quality of the electric signal recorded. *Ball et al.* compared invasive and non-invasive EEG measurements [14]. The main goal of the study was to determine if blink related artifacts, which are always present in non-invasive recordings, were also present in invasive recordings. As expected, eye blinks caused artifacts in non-invasive recordings and, unexpectedly, these artifacts were also present in the invasive recordings, particularly in the prefrontal region. After this analysis, by computing the ratio between the amplitude of the artifacts and the amplitude of the background brain activity, it was possible to determine that the quality of the invasive EEG signal was from 20 to 100 times better than the non-invasive EEG recorded simultaneously. Reduction in noise and eye-blink artifacts would allow to improve the quality of the EEG signal and to obtain a better implementation of a BCI system, since classification of EEG signal can be performed easily.

  Despite of the advantages in using invasive techniques, the majority of researchers consider the non-invasive techniques as more appropriate that the invasive ones, since they have an advantage that overcomes all the potential advantages of the invasive techniques: they do not require a complex and dangerous surgery for the subject. [2].

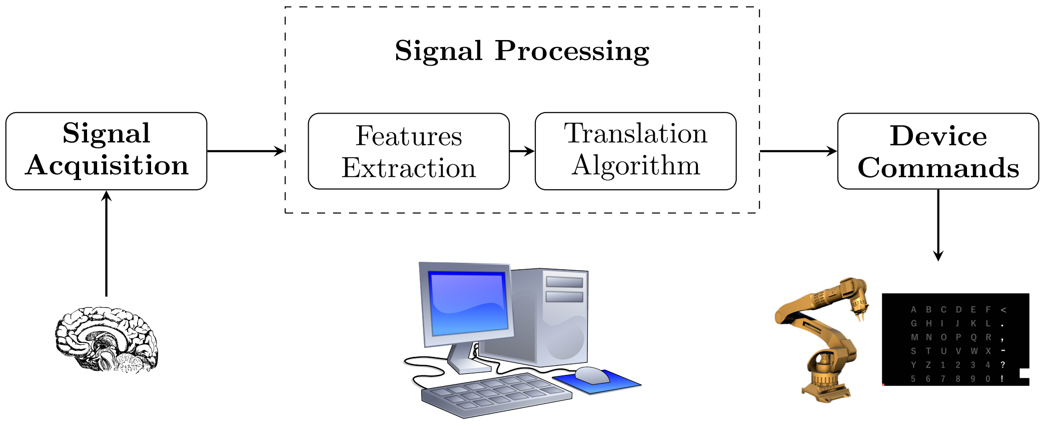


Figure 2.4: Basic design of a generic BCI system. In this diagram, the different parts of a BCI are shown: *(1) Signal acquisition, (2) Signal processing (3) Output Device.*

2.3.3 Components

In Figure 2.4 the main components of a BCI are shown. A BCI can be described as a generic control system. A control system has an input, and components that process the input signal and transform it into the output signal of the system. In the case of a BCI, the input signal is the signal that is being used to monitor the cerebral activity (*e.g.* the EEG signal). The components that process this input signals perform a processing of the input signal (*e.g.* filtering and classification), while the output signal may consist of commands sent to a robotic arm or by letters shown on a screen.

Signal acquisition In Section 2.3.2, we discussed the difference between invasive and non-invasive recordings of the brain activity. What we would like to highlight here is that it is of utmost importance that the quality of the recording is the best possible. The introduction of noise and artifacts could affect and greatly reduce the overall performance of the BCI. For non-invasive EEG recording, there are some entities that need to be defined: the number of channels to be used, the position of the electrodes on the scalp, the EEG acquisition device and so on. Not only these entities affect the ability of the BCI to extract features from the EEG signal, but they also have an effect on the portability of the system. If a bulky EEG system is chosen for the recording, then it would be difficult for the subject to use the BCI outside of a hospital/laboratory.

Signal Processing As visible in Figure 2.4, the signal processing block is composed of two methods: feature extraction and translation algorithm. In the feature extraction method, the signals are subjected to a feature extraction procedure. Example of feature extraction procedures are filtering or spectral analysis. Hopefully, this analysis determines the features representing what the user wants to communicate. Features that are commonly extracted are related to specific brain waves or rhythms that mirror events currently happening in the brain.

The translation algorithm method converts the previously extracted features into device commands. Effective translation algorithms must adapt to the user's signals characteristics [2]. Three different levels of adaptation can be identified. At the first BCI access of each user, the algorithm has to adapt to him/her by analyzing the features of the signal. EEG signals typically display variability on both short and long term. A translation algorithm that only has the first level of adaption would be completely ineffective on the short and long term usage of the BCI. An additional level of adaptation, with periodic modifications aimed at reducing the impact of these variations, is required. The third and last level of adaptation is the most challenging one. When a signal feature that has always been a reflection of the brain function starts to be an output signal encoding the user's intent, it is then subjected to the adaptive capabilities of the brain. So, the outcome of the BCI will also affect the input signal of the BCI, and an effective translation algorithm should take this into consideration.

The Output Device For the majority of BCI systems, the output device is a screen and the real output is the result of the selection of symbols presented on the screen. These symbols can be letters, icons, or any other type of stimuli. According to the choice that is selected by the user, a letter is displayed on the screen, or the action of an external device (wheelchair, robotic arm, . . .), correspondent to the selected icon, is performed.

2.3.4 Non - Invasive BCI

Both invasive and non-invasive techniques can be used to monitor the cerebral activity and build a BCI. Example of non-invasive techniques are the following ones:

1. **Magneto - Encephalography (MEG)**: MEG is a technique that makes use of magnetic fields to monitor the brain activity. Intracellular currents occur in a natural way in the brain, and magnetic fields are produced by these currents. In order to detect these magnetic fields, sensitive magnetometers have to be used [15]. With this technique, areas that are active during cerebral processes can be identified. Main problems related to MEG are: sensitivity to external sources of noise, and expensive, cumbersome and non-portable equipment;

2. **Functional Magnetic Resonance Imaging (fMRI)**: fMRI is an imaging technique that makes use of the magnetic resonance to evaluate the status of the brain. This imaging technique is complementary to the morphologic imaging, which is focused on analyzing the morphology of the organ. The signal that is measured with fMRI-based BCI is the BOLD (Blood Oxygenation Level Dependent) signal [16]. This measure is an indirect measure of the activity level of a cerebral area. A greater amount of oxygen consumption corresponds to a higher level cerebral activity. The problems associated with fMRI-based BCI are the same as the one described for the MEG. In addition to these, a delay of $\sim 3-6$ seconds, between the cerebral activity and the recorded signal, could be present;

3. **Slow Cortical Potentials (SCP)**: SCP are low-frequency (DC-2Hz) voltage changes recorded at the level of the scalp. These voltage changes are associated to cognitive and sensorimotor events. Negative SCP are related to functions that cause an activation at the level of the cortex. Positive SCP are associated with a reduction in the cortical activation [2]. SCP - based BCI depend on the ability of people to learn how to control these voltage changes. *Birbaumer et al.* developed a BCI in which subjects could use SCP for movement control (upwards, downwards) of a cursor displayed on a screen [17].

4. **Mu Rhythms**: in awaken people, motor cortex displays an activity in the frequency range of 8-12 Hz. This activity is called *mu rhythm*, and is comprised of a variety of different rhythms. Each rhythm can be distinguished according to the location, frequency and relationship to a contemporaneous motor output. The prior preparation to the movement and the movement itself correspond to a decrease in the mu rhythm in a controlateral way (event-related desynchronization). An increase of the rhythm occurs after movement and during relaxation. For the mentioned reasons, a pattern visible in the mu rhythms can be associated to a specific movement. The identification of this pattern can then be used to control external devices. No real movement is necessary, since mu rhythms are also present with motor-imagery [18].

2.4 P300 Event-Related Potential

Event Related Potentials (ERPs) are stereotyped electrophysiological responses that happen after a sensory stimulation. ERPs are visible in the EEG signal and can be distinguished from the background electric activity of the brain [11]. Evoked potentials can be classified in two main categories. Those dependent on the nature of the stimulus are defined as exogenous potentials.

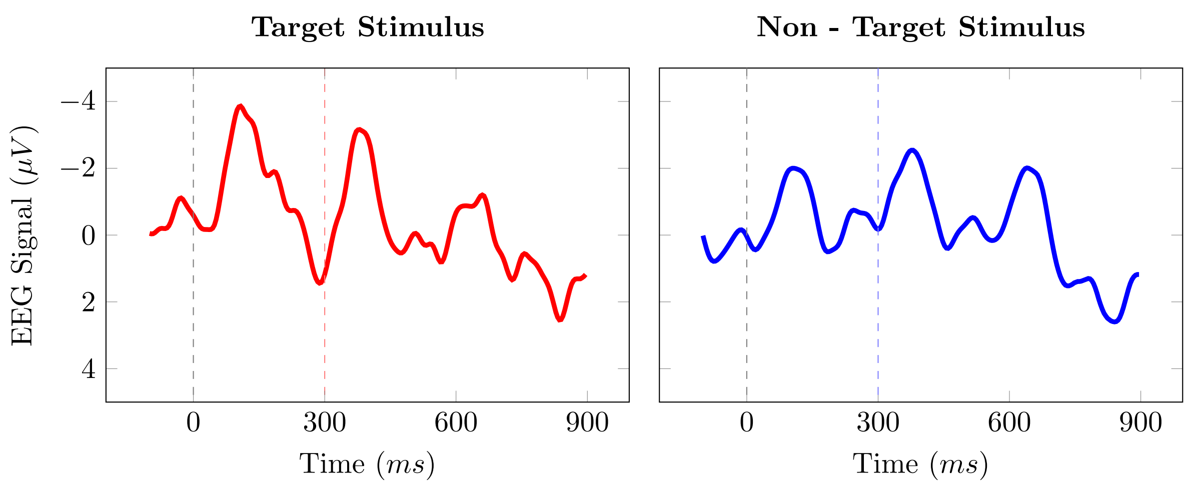
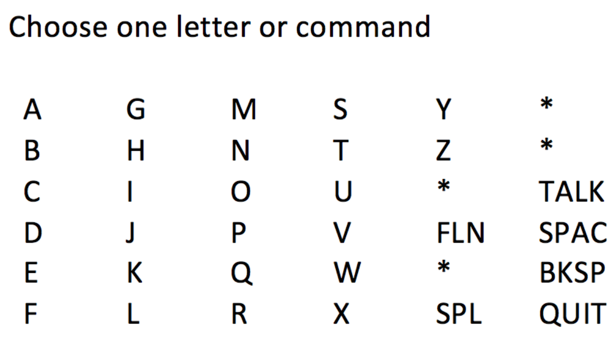


Figure 2.5: *Left:* EEG signal when a desired stimulus is presented. *Right:* EEG signal when an undesired stimulus is presented. Time *0* represents the stimulus onset.

Those dependent on the meaning of the stimulus, rather than on its nature, are defined as endogenous potentials.

One of the most known ERP is the P300 response. A P300 response is elicited when an auditory, visual, or somatosensory stimulus, which is significant for the subject, is presented to him/her infrequently and interlarded with insignificant stimuli. The P300 response is visible in the EEG trace as a positive peak at about 300 ms after the stimulus onset [5]. The reason why this ERP is called P300 is that it is a positive (**P**) deflection that happens **300** ms after the stimulus onset. The P300 response is commonly referred to as the "oddball" response. As shown in Figure 2.5, it can be noticed that P300 response is predominant in responses elicited by the stimulus representing the user's intent (so, when the stimulus is the *target* one). Instead, no P300 response is present when the provided stimulus does not reflect the user's intent (so, when the stimulus is not the *target* one).

The main properties of a P300 response are the magnitude and the latency of the response. These properties are dependent on the chosen stimulation paradigm. *Duncan et al.* showed that the amplitude of the peak of the P300 response is inversely proportional to the frequency of the target stimulus [19]. Therefore, the higher the frequency of the presentation of the target stimulus, the lower is the amplitude of the wave. This makes it clear that the frequency of the stimulation affects the resulting generated response. *Ruchkin et al.* showed that the latency of the P300 response is related to the time necessary for the subject to fully recognize and understand the presented stimulus [20]. So, the latency depends on the complexity of the stimulus and on the amount of information carried by the stimulus.



Choose one letter or command

A	G	M	S	Y	*
B	H	N	T	Z	*
C	I	O	U	*	TALK
D	J	P	V	FLN	SPAC
E	K	Q	W	*	BKSP
F	L	R	X	SPL	QUIT

Figure 2.6: Grid used by *Farwell and Donchin* for the first P300 - based BCI [21].

The P300 response has a great potential in the BCI field. *Farwell and Donchin* [21] described the development of the first P300 based BCI. In this BCI, the user is presented with a 6 by 6 grid, with both letters and one-word commands. This grid is shown in Figure 2.5. A 100 ms flash of a row or a column happens every 125 ms, and in each repetition, composed of 12 flashes, each letter/number/command flashes twice. For the user, it is possible to make a selection by focusing on the desired symbol. Counting how many times the symbol flashes helps the user to stay focused on the task. In order to identify the presence of a P300 response in the EEG trace, the authors used stepwise discriminant analysis as a classification technique. This procedure yields to a score measuring the distance between each epoch, composed of 1200 ms of EEG signal extracted in a symmetric way around the time at which the stimulus is provided, and the average of a group of epochs known to include a P300. With the value of this measure, it is possible to determine if that epoch contains a P300, and so if that stimulus is the one on which the user was focusing his/her intent.

The greatest advantage of employing the P300 response in BCI field is that subject training is not required. This is due to the fact that the P300 response is an innate response of the human brain. In contrast, slow cortical potentials and mu rhythms require a training phase for the subject to be successfully employed in a BCI, since the individual has to learn how to control them in order to achieve the desired result.

The P300 response may change over time. Periodic adaptation of the translation algorithm to the current characteristic of the wave is required in order to have a constant good performance. When employing the P300 response in BCI, the communication rate is low. In fact, the response of the subject is time-locked to the presentation of the stimulus. For this reason, it is necessary to set a value of inter-stimulus interval (ISI) that allows to increase the transfer rate of information but, at the same time, allows to obtain a good performance in the identification of the P300 response [22].

One of the goals of our wearable BCI device is to control a robotic arm. In order to perform this control through a BCI, the choice of the P300 response as the electrophysiological signal to be used to control the arm was the best one. Using a stimulation grid similar to the one employed by *Farwell and Donchin* in [21] and presenting a predetermined set of possible movements the robotic arm can perform, the user would be able to select the movement he would like the robotic arm to perform. We also wanted to develop a system that doesn't require any training of the subject. The P300 response is an innate response of the brain, and so no training of the subject is required.

Our aim is that this work will be useful in helping the definitive transition of BCI from hospital or laboratories to patients' houses. Using wearable and affordable EEG equipment, a small robotic arm and a computer screen, the disabled could make use of a reliable method for communicating and interacting with other people.

2.5 Related Works

Several BCI based assistive devices were developed in the past years. As we mentioned earlier, *Farwell and Donchin* were the first group who presented a P300 based BCI to type words [21]. Other examples of BCI spellers can be found in *Donchin et al.* and in *Krusienski et al.* [23, 24].

Assistive devices for the control of electric wheelchairs were developed. *Iturrate et al* and *Rebsamen et al.* both presented P300 based BCI systems that could be used to control electric wheelchairs [25, 26].

Congedo et al. presented a prototype of a P300-based video game working, using OpenVIBE [27, 28]. *Finke et al.* were able to develop a P300 video-game obtaining a classification accuracy of 65% on single trials [29].

In this section, we review three studies related to the BCI control of external devices. In two of them, P300 is employed. In one of them, steady state visual evoked potentials are used.

2.5.1 P300 BCI Robotic Arm Control

Palankar et al. developed a system to control a 9 DoF wheelchair mounted robotic am using a P300 based BCI [30]. The robotic arm used in this system was designed to fulfill the specifications necessary to perform daily tasks. They used a relatively expensive and powerful robotic arm. The most widely used robotic arms, similar to this one, such as the Jaco Kinova [31], typically cost around \$ 30000 to \$ 40000 [32] . Using a more cost effective robotic arm would allow more

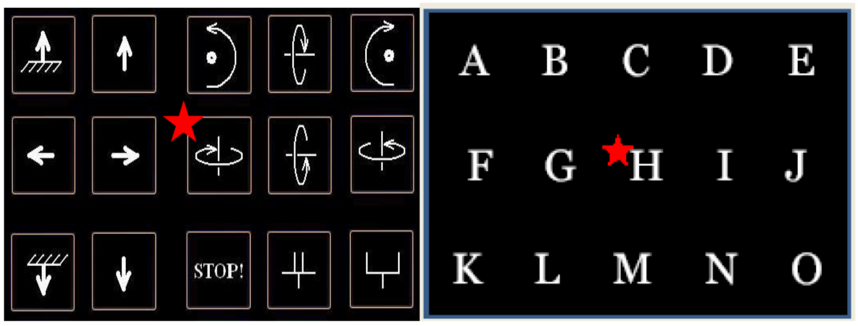


Figure 2.7: *Left:* mapping between letters and robotic arm movements. *Right:* The grid presented to the user with flashing letters [30]. ©*[2011] IEEE.*

people to use such a system at home too.

In this study, the control of the robotic arm was achieved with a P300 based BCI. The authors used the BCI2000 software [33] to stimulate the user, record and process the EEG signal. In this case, the application presented to the user a grid, very similar to the one used by *Farwell and Donchin* [21], showing letters from A to 0. The stimuli grid presented to the user is shown in Figure 2.7. The cells are arranged in a 5 x 3 matrix. A 75 ms intensification of a column or a row happens every 125 ms. So, each sequence of flashes, equal to 8 intensifications, lasted for a total time of 1 second. Each letter is mapped to a movement of the robotic arm. However, the subject is not presented with movements of the robotic arm directly, but with letters that are associated to these movements. The mapping between letters and movements is shown in Figure 2.7.

Other than the head cap used for EEG recording, no information about the EEG acquisition system is provided in the paper.

The authors report an important issue related to safety. Since the complete process of scanning the matrix and detecting the selected stimulus takes about 15 seconds, a delay in the response of the robotic arm is introduced. Should a dangerous situation arise, the robotic arm wouldn't be able to respond in time, and a possible crash could take place.

It is possible to identify three main improvements that need to be done in order to build a cost effective P300-based BCI system for the control of a robotic arm. First, it is necessary to present to the user stimuli visually representing movements of the robotic arm; then, a cost effective robotic arm should be used; last, safety issues have to be solved. It would be relatively difficult for locked-in individuals to use such system in the daily life, because the user would have to always remember the mapping between letters and movements of the robotic arm.

2.5.2 P300 BCI Humanoid Robot Control

Bell et al. developed a P300 BCI to control a small humanoid robot [34]. This robot was programmed in a way so that it could be controlled with high-level commands. No low level commands had to be used to control the robot.

To develop the system, the authors used a dynamic image-based BCI. The user is presented with flashing images, recorded from cameras mounted on the robot, and the P300 response is utilized to determine the image that the user is focusing on. Thus, enabling the robot to pick up the correspondent object.

EEG signal was recorded using a BiosemiActiveTwo system, which is an acquisition system designed for electro physiology research [35]. This acquisition system is expensive ($ 16000) for the 16 channels version) and non-portable.

During a generic selection, the border of each image is intensified in a random sequence; the flashing happens every 250 ms, and each intensifications lasts for 125 ms. The subject has to focus attention on the image of interest. After 10 flashes per image, a classification on EEG is performed. This classification is used to determine, according to the selected image, where the humanoid robot should direct and which object it should pick up. Average classification accuracy was 98.4% amongst subjects.

The main outcome of this study is that non-invasive BCI can be used to control complex devices, such as humanoid robots. The authors state that this system could be potentially used in the field of helper robots for disabled patients. These robots could move in a home environment, performing actions that the paralyzed individuals are no longer able to do.

The authors also suggest that other types of EEG responses, such as SSVEPs and mu rhythms, could be used for command selection and for the control of devices with better results than P300 response. However, these approaches require more training data and show wider variation across subjects, which are not true for the P300 response.

2.5.3 SSVEP BCI Hand Orthosis Control

Ortner et al. developed a system to control an orthosis using Steady State Visual Evoked Potentials [36]. These potentials are natural responses to visual stimulations occurring at a fixed frequency f_0. The eye, when excited with a visual stimulus at frequency f_0 generates an electrical activity in the brain at the same frequency or at multiples of the stimulation frequency.

SSVEP signals can be used in the BCI field for many tasks, ranging from games to the control of hand orthosis [37]. In general, the use of SSVEP in the BCI field requires a lower

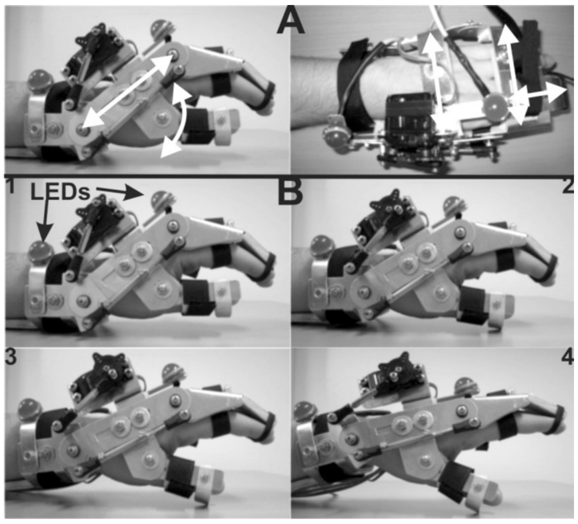


Figure 2.8: Hand orthosis used for the development of the SSVEP-based BCI [36]. © *[2011] IEEE.*

amount of training than other types of potentials. In fact, the subject is required to only gaze at a light source, without performing any complex task. The drawback of SSVEP control is that the user is required to maintain and focus attention on the blinking lights, and this may cause fatigue.

The hand orthosis used in this study shown in Figure 2.8. For the development of this system, two LEDs flashing at 8 and 13 Hz were positioned on the orthosis. If the subject gazed at the LED mounted on the left part of the orthosis, the hand would open gradually. Gazing at the LED mounted on the right part of the orthosis caused the hand to close gradually.

To record the EEG signal, the authors used the g.bBSamp EEG amplifier [38] with only one bipolar channel placed in a position close to O1 and the ground electrode placed on Fz. EEG signal was sampled at 256 Hz. This acquisition system is very powerful and used in hospital and medical clinics, but it is very expensive.

The results that the authors obtained in these studies were good, with a positive predicted value of 79% ± 21%. The advantage of this type of control is that the end-users can send information whenever they want to, without waiting for a specific stimulus to be presented, as it happens with synchronous BCIs. One of the problem that were highlighted in this study is that the number of false positive detection (i.e. the number of times an SSVEP was identified even if the subject was not focusing on any LED) was high. According to the authors, this could be due to the fact that, even without focusing on the light sources, the subjects still had the light sources in their visual fields.

Chapter 3

Hardware

In this chapter, we describe the hardware components that were used in the present work. The hardware mainly consists of two main modules: the EEG acquisition system and the robotic arm.

3.1 EEG Acquisition System

The goal of this project was to build a wearable and cost-effective BCI assistive device. For this reason, two cost effective EEG recording devices available on the market were compared. The two devices we compared were the Emotiv Epoch [39] and the OpenBCI Cyton board [4]. Emotiv Epoch is a 14 channels wireless (Bluetooth communication) EEG acquisition device. The electrodes are saline based wet sensors. The technical specifications of Emotiv Epoch are described in Table 3.1a. The market price of the device is of 800$. OpenBCI Cyton is an open-source electronic board with a variable number of channels (up to 16). OpenBCI Cyton is compatible with any type of electrodes. The technical specifications of the OpenBCI Cyton board are described in Table 3.1b. The price of the 8 channel board is of 499$. Recently, a new model of the board was released: the OpenBCI Ganglion board. The OpenBCI Ganglion board has only 4 channels, but it features Bluetooth 4.0 communication protocol. The market price of the OpenBCI Ganglion Board is of 199$.

The Emotiv Epoch is a commercial product that is sold together with a proprietary software. With this proprietary software, there is no easy way to access the raw EEG data. Electrodes in Emotiv Epoch are embedded in a pre-assembled headset. So, using this device means being completely locked to the electrode locations of the pre-assembled headset. These locations are, according to the 10-20 system notation, the following ones: AF3, F7, F3, FC5, T7, P7, O1, O2,

Table 3.1: Technical specifications of the Emotiv Epoch EEG device [39] and of the OpenBCI Cyton board [4].

(a) Emotiv Epoch

Property	**Value**
Channels	Up to 14
Sampling rate	256 Hz
Resolution(bit)	14
Filtering	[0.2 - 43] Hz

(b) OpenBCI Cyton

Property	**Value**
Channels	Up to 16
Sampling rate	250 Hz
Resolution(bit)	24
Filtering	None

P8, T8, FC6, F4, F8, AF42. Since one of the most important components in detecting the P300 response is the electrodes locations, Emotiv Epoch wouldn't allow to choose other electrodes locations to be used.

For the above mentioned reasons, after this preliminary analysis we decided to use the OpenBCI Cyton board. The company which produces OpenBCI boards bases its philosophy on open-source technology. So, complete specifications of the board, including the firmware and the protocol used for Bluetooth communication, are available. The access to this information allowed us to read EEG raw data in an easy way. Then, it was also possible to develop a custom firmware for the board. Last but not least, we could decide which set of electrodes locations to use.

The OpenBCI board is based on the MicroChip PIC μcontroller model PIC32MX250F128B [40]. This microcontroller uses the ADS1299 [41] as the circuitry for analog to digital conversion. OpenBCI board transmits the data using Bluetooth communication. If the default communication protocol is used, a USB dongle plugged in into a computer is necessary in order to retrieve the data sent by the board.

The model of the OpenBCI board that we used is the Cyton board version 3. The board is shown in Figure 3.1. One channel of the board is used as a reference for all the others, and another channel is used as the bias channel. The bias channel makes use of destructive interference waveform techniques to eliminate the common mode noise of all the active recording channels.

One of the questions that we wanted to answer was if the quality of the signals recorded with the OpenBCI Cyton board can be compared to the one of a medical device, since the quality of the signals is a fundamental component of a BCI. *Frey* presented a study in order to

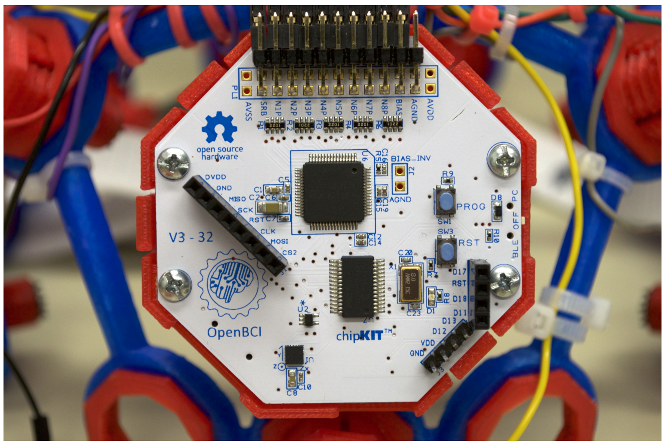


Figure 3.1: The OpenBCI Cyton board used as the EEG acquisition system in the device [4].

fully address this question [42] . In the study, the OpenBCI board was compared to a medical grade EEG system, the g.tec g.USBamp [38]. To perform the comparison, the EEG signals were contemporaneously recorded from the two devices in the same locations of the 10-20 systems. The main outcome of this study was that the OpenBCI board represents a good alternative to the most traditional EEG devices. Correlation computed both in time and frequency domain showed that the signals acquired by the g.USBamp and by the OpenBCI board were very closely related. The Pearson R score was higher than 0.99 for all the channels tested.

It is important to underline here that the OpenBCI board is not a medical device, and is not intended for that use: in fact, it has no certification.

Another important aspect is that the OpenBCI board uses a battery as a power source. Using a battery, hazards caused by the power supply can be avoided. Establishing a connection between the human body and the power lines requires protection and isolation circuits, fulfilling international standards for safety. The battery that we used to power the OpenBCI board is a 3.7V, 500mAh lithium battery.

3.1.1 EEG Headset

In order to record the EEG signal with the OpenBCI board, electrodes need to be precisely positioned on the scalp with an harness that limits motion artifacts. OpenBCI is an electronic board designed to record any type of biological signal (EEG, EKG, EMG,...) and therefore no EEG electrodes harness is supplied with the board.

For this reason, we had to determine which type of support to use. In order to do this, three possible options were identified:

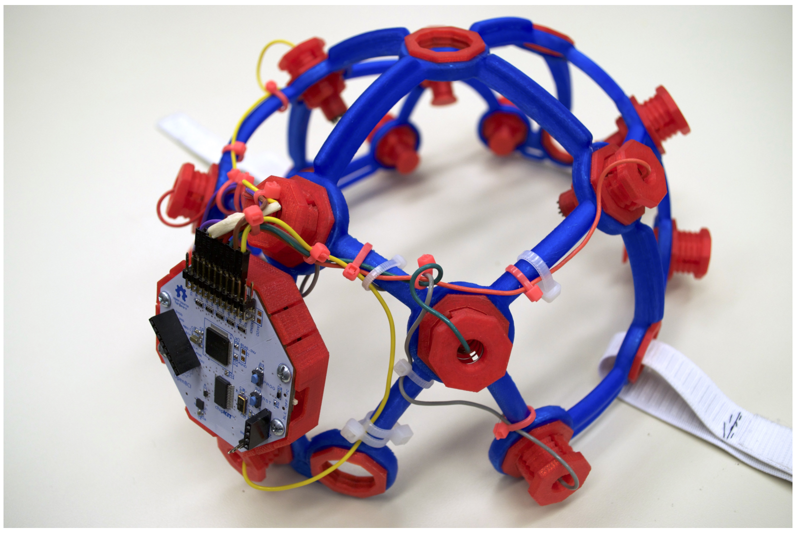


Figure 3.2: The Mark III 3D printed EEG headset that was used as a support for the EEG dry electrodes. The chin strap that we included to add more stability to the headset is visible.

- **Gold cup electrodes with harness**: OpenBCI comes with gold-cup electrodes. These electrodes typically have a low impedance, below 20 $kOhm$, and allow to obtain high quality EEG recordings. Conductive paste is required to be used with these electrodes. This paste has to be placed on the scalp, on the positions correspondent to the chosen electrodes locations. From the end-user point of view this is not a good solution, because it would require the constant presence of a care-giver who takes charge of the application of the paste and of the subsequent positioning of the electrodes. Furthermore, electrodes have to be fixed on the user's scalp using elastic bands. These elastic bands are not, in our opinion, suitable for long term recording. They may cause involuntary movements of the electrodes, decreasing the quality of the EEG signal;

- **Electro-caps**: these caps are head caps specifically designed for EEG recordings. They are used in hospital for the recording of EEG, and allow to obtain high quality signals over long-term recordings. There are three main problems with these caps when designing a cost effective device: first of all, the market price associated to them ranges from 250 $ to 400 $, according to the chosen model; they also require an adapter, which costs around 100$, in order to be used with the OpenBCI board; lastly, they also require the use of a conductive gel to be applied between the cap and the scalp;

- **3D printed headset**: the last option was to use a 3D printed headset, acting as a support for the electrodes. Files for 3D printing are released by the OpenBCI company

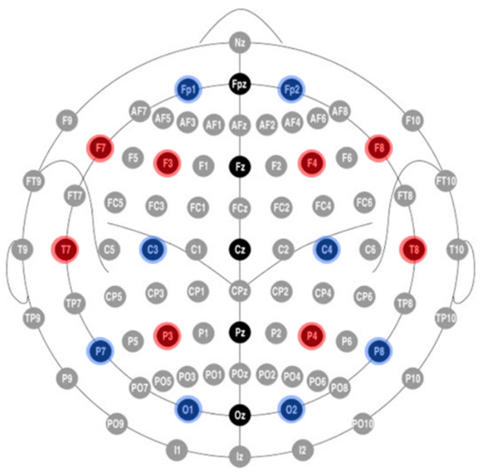


Figure 3.3: The available electrodes locations of the Mark III 3D printed EEG headset. Image taken from [43].

[4]. This headset can be easily 3D printed with adjustable sizes according to the user's head circumference. This headset makes use of dry electrodes, that don't require any specific paste or gel to be applied on the scalp. Furthermore, printing and purchasing the material to build this headset costs around 100$, which is far less than the cost necessary to buy an electro-cap. Therefore, we used the 3D printed headset model Mark III [43] for our wearable and cost effective device.

An image of the headset that we built and used to test our device is shown in Figure 3.2. A map with all the electrodes locations that are available in the Mark III headset is shown in Figure 3.3. These locations reflect the 10-20 international standard for EEG recording.

The P300 response is mostly predominant on the parietal region and along the central line of the scalp [5]. For this reason, the default electrodes locations of the device are: C3, C4 on the "central" lobe; P7, P3, Pz, P4, P8 on the parietal lobe; Oz on the occipital lobe. As reference and ground, two ear clip electrodes placed on the right and left earlobes are employed.

3.1.2 Bluetooth Communication

As we previously explained, the default way with which the OpenBCI board communicates with a laptop is through a USB Dongle. While this solution is perfectly suitable when the OpenBCI board is used in conjunction with a laptop, it becomes impracticable if it has to be used with a mobile device, that can be either a smartphone or a tablet. Since one of the key features of the presented device is its being wearable, this aspect has to be taken into serious consideration.

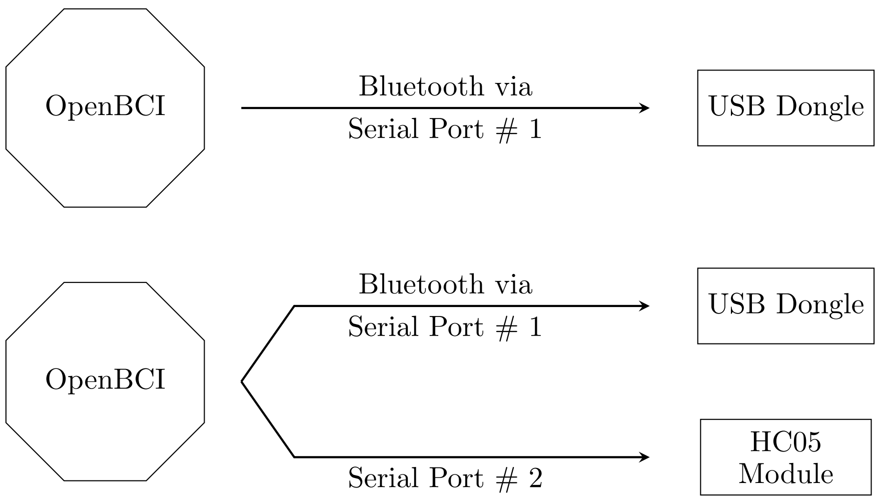


Figure 3.4: *Top:* the default communication protocol used by the OpenBCI board, with serial port #1 used to communicate with an USB Dongle. *Bottom:* the modified communication protocol, with serial port #1 used to send data to an USB Dongle, and serial port #2 used to send data through the HC05 module.

For this reason, we proceeded in modifying the communication protocol used by default by the OpenBCI board. The purpose of this modification was to enable the OpenBCI board to communicate directly via a Bluetooth module, avoiding the need of a USB Dongle. In particular, the Bluetooth module that we choose is the HC05 module, which is a low cost Bluetooth module.

Two main steps were required for the modification of the communication protocol:

- from the *hardware* point of view, there was the necessity to connect the HC05 Bluetooth module to the OpenBCI board. The OpenBCI board features two hardware serial ports. By default, the second serial port is used only to send debug messages. Thanks to the presence of the second serial port, it was possible to connect the HC05 module to the OpenBCI board pins used by the second serial port;

- from the *firmware* point of view, it was necessary to transmit data over both the serial ports. So, we proceeded in modifying the existing library and the firmware of the OpenBCI board, in such a way that data and debug messages are sent over both serial ports.

A schema representing the hardware and firmware modifications is shown in Figure 3.4. These modifications allowed us to developed a BCI mobile application, as it will be explained later in Chapter 4.

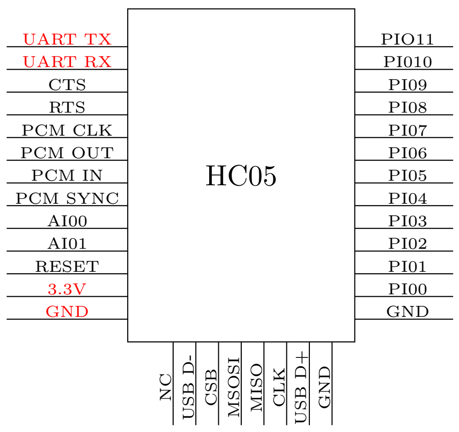


Figure 3.5: The schematic of the HC05 module. The pins that were used for the connection with the OpenBCI board are highlighted in red. The RX and TX pins were used for data transmission and receiving, while the GND and 3.3V pins were used for power supply.

HC05 Module

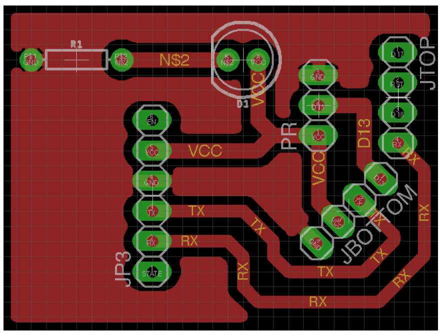
The HC05 is the module that we choose as a Bluetooth module to transmit data from the OpenBCI board to mobile devices, such as a smartphone or a tablet. This module is a low cost Bluetooth module, that can be used with a baud rate up to 1382400 bit/s. The schematic of the HC05 module is shown in Figure 3.5. Only 4 out of the total 34 pins available on the HC05 module were used. In particular, the RX and TX pins were used for data transmission and receiving, and two pins (GND and 3.3V) were used for power supply.

The default properties of the HC05 module set a baud rate of 9600 bit/s. Since the sampling frequency of the OpenBCI board is 250 Hz, this baud rate would be too slow for a reliable data transmission. For this reason, we changed the properties of the HC05 module, by setting a baud rate of 234000 bit/s. This allowed us to reliably transmit the data sent by the OpenBCI board, without experiencing any delay in data transmission.

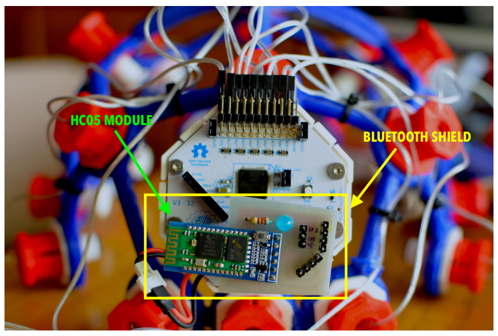
Bluetooth Shield

In order to have the HC05 Bluetooth module securely connected to the OpenBCI board, a PCB was designed and printed to act as a *Bluetooth shield* for the OpenBCI board. The design of the PCB, and the PCB, are shown in Figure 3.6.

As shown in Figure 3.5, only 4 pins are necessary to the HC05 module to function properly. Two pins are required for data transmission and receiving, while two other pins are required for power supply. The connections between the HC05 module and the OpenBCI board are the following ones:



(a) The design of the PCB.



(b) The PCB mounted on the OpenBCI board.

Figure 3.6: The PCB of the Bluetooth shield used to connect the HC05 module to the OpenBCI board.

- the RX pin of the HC05 module is connected to pin D11 of the OpenBCI board, which is connected to the internal TX pin;
- the TX pin of the HC05 module is connected to pin D12 of the OpenBCI board, which is connected to the internal RX pin;
- the 3.3V pin of the HC05 module is connected to pin VDD of the OpenBCI board;
- the GND pin of the HC05 module is connected to pin GND of the OpenBCI board

As it will be explained in Chapter 4, in the proposed BCI system a photoresistor was used to synchronize the EEG, read by the OpenBCI board, with visual stimuli provided to the user. In order to ease the connection between the photoresistor and the OpenBCI board when the Bluetooth shield is being used, a 3-pin connector is present on the PCB, so that the sensor can be power supplied and its value can be read by the OpenBCI board.

Furthermore, a blue LED is mounted on the PCB, so that it is possible to give a feedback to the user about the current status of the board. A 220Ω resistor is connected to the LED in order to reduce the amount current flowing into it. The HC05 module that we used in this project already has a built-in red LED that allows to obtain information about the current Bluetooth connection.

3.2 Robotic Arm

One application of our BCI device is to control a robotic arm. We choose to use a robotic arm produced by Lynxmotion: the AL5B arm [44]. The technical specifications of the robotic arm are described in Table 3.2. A picture of the complete setup for the robotic arm is shown in

Property	Value
Shoulder to Elbow	12.05 cm
Elbow to Wrist	12.70 cm
Wrist to Tip	8.50 cm
Height (reaching up)	40 cm
Weight	446.5 g

Table 3.2: Technical specifications of the Lynxmotion AL5B robotic arm.

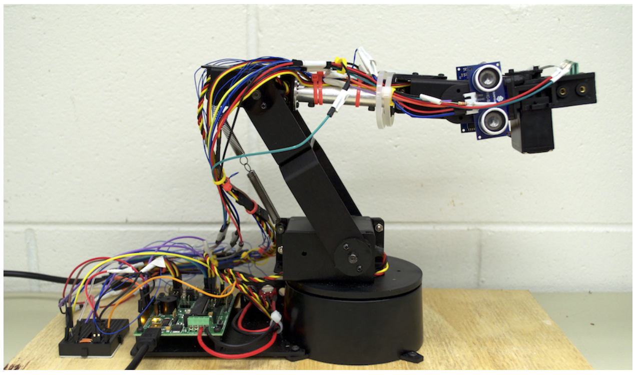


Figure 3.7: The setup of the Lynxmotion AL5B robotic arm.

Figure 3.7. The robotic arm has 4 degrees of freedom. One DoF is due to the rotation of the base, and the other three are due for three joints that act as shoulder, elbow and wrist. An additional DoF can be added with a servo motor for the rotation of the wrist.

The movement of each joint is obtained by setting the angle of the correspondent servo motor. In order to achieve this, the servo motors have to controlled with an electronic board. The electronic board that we used to control the servos is the BotBoarduino microcontroller [45]. This board is based on the Arduino Duemilanove [46], and allows to control a set of servo motors in an easy way.

We powered the board directly from the USB port of a laptop, and we used a 6V power supply to power the 5 servo motors of the robotic arm. This distinction in the power supply was necessary because the servo motors could take away the current necessary for the electronic board to function properly.

In order to enhance the functionality of the robotic arm, we added ultrasonic and force sensors. Two ultrasonic sensors are used to avoid crashes of the robotic arm with close objects, while two force sensors are used to detect the width of the object being grabbed.

The type of ultrasonic sensor used is shown in Figure 3.8. Each ultrasonic sensor is placed on one side of the robotic arm, in a position close to the gripper. These sensors are used to detect the presence of any object in proximity of the robotic arm. The chosen proximity threshold is 10cm.

The principles behind ultrasonic sensors is very simple. An ultrasonic sensor sends high frequency sound pulses at a regular time interval. If the sound wave strikes an object, the pulse is reflected back. By determining the time difference between the sent and received signal, one

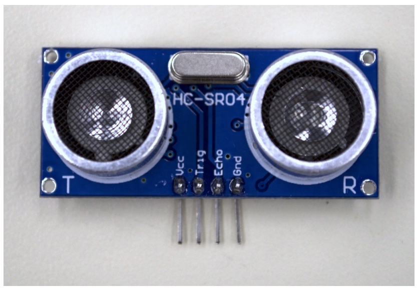


Figure 3.8: The HCSR04 ultrasonic sensor.

can compute the distance of the object, according to this equation:

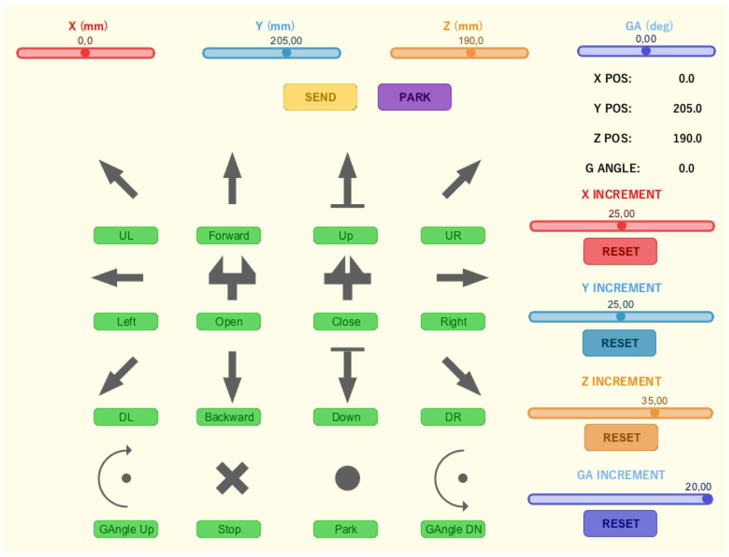
$$d = \frac{t \cdot v}{2} \tag{3.1}$$

where d is our unknown variable (distance), t is the measured time and v is sound speed in the air, which is approximately 767 mph. The 2 in the denominator is necessary because the sound wave has to travel back and forth in the air in order to be detected.

On the gripper of the robotic arm, we added two force sensing resistors. These sensors were necessary to detect the width of the object being grabbed, in order to avoid crashing the object. This technique prevents the servo motor to overheat, that could happen when the servo motor keeps moving while an object has already been grasped by the gripper.

A software was developed using the Processing programming language [47] to control the robotic arm. This software communicates via a serial port with the BotBoarduino board controlling the servo motors of the robotic arm. A screenshot of the software is visible in Figure 3.9. This software was designed with the following idea: each icon corresponds to a discrete movement of the robotic arm, such as *up*, *left*, ...

When a button is clicked, the updated X, Y, Z coordinates and the value of the wrist angle are sent to the robotic arm, and the arm would move accordingly. The current position of the robotic arm is always displayed and updated with the real current position of the robotic arm. The software also allows to control the incremental value for the discrete movement along the three axis and for the angle of the wrist. So, it was possible for us to identify the best set of values to be used in the BCI control of the robotic arm. In fact, a small incremental value would have resulted in the possibility of performing finer movements, but at the expense of a longer time. A large incremental value, instead, would have resulted in a smaller amount of time necessary to cover the same distance, but at the expense of movement resolution. Experiments were performed in order to identify the set of incremental values leading to the best trade off between time and movement resolution.


X (mm) 0,0
Y (mm) 205,00
Z (mm) 190,0
GA (deg) 0,00

X POS: 0.0
Y POS: 205.0
Z POS: 190.0
G ANGLE: 0.0

X INCREMENT 25,00
Y INCREMENT 25,00
Z INCREMENT 35,00
GA INCREMENT 20,00

Figure 3.9: The interface of the software developed for robotic arm control. Buttons to perform discrete movements, sliders for setting a specific position and sliders for setting increment values are visible. These tools can be used to tune the parameters for robotic arm control in an optimal way.

Some locations in the three-dimensional space may not be accessible to the robotic arm, since reaching that location may be obstructed by close objects. Therefore, it is important to read back the correct values of X,Y, Z coordinates and the wrist angle, which are sent by the BotBoarduino board.

Furthermore, the software we developed allowed us to directly specify the X, Y, Z coordinates and the wrist angle the robotic arm should reach. This resulted useful to us to determine the extreme locations that can be reached by the robotic arm.

Chapter 4

Software

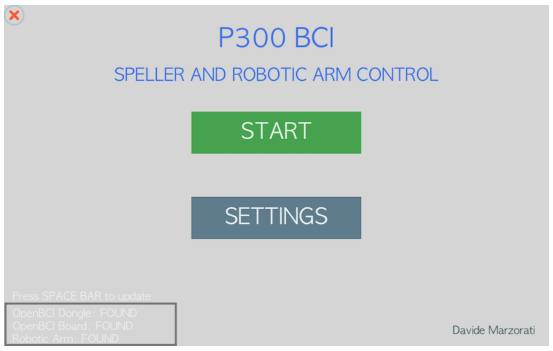
In this chapter, we describe all the software components that were developed in this work. The software consists of two main applications: a BCI software running on a computer and a mobile application running on Android smartphones and tablets.

4.1 Computer Based BCI

After setting up the EEG acquisition system and the robotic arm, we developed a BCI software running on a computer. The main goals of this software are the following ones:

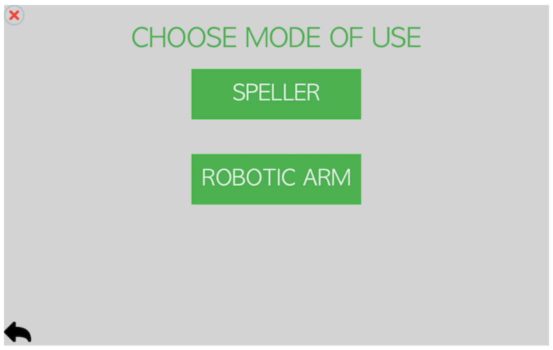
- present to the user visual stimuli using a flashing grid of options. When the stimulus representing the user's intent flashes, a P300 is elicited;
- acquire EEG signal streamed by the OpenBCI board;
- process the EEG signal. In particular, perform a classification on the EEG signal aimed at the identification of the P300 response;
- communicate with the robotic arm, sending commands to it and receive information back from it.

We analyzed several programming languages and software solutions that could be employed to develop our BCI application. After this analysis, we decided to use Processing [47], the same software that we used for the robotic arm control application. Processing is mainly used to develop graphic-based application, but it is also possible to easily use it to communicate with external devices through serial communication. The combination of these factors lead us into choosing Processing as the software to develop our BCI application.



Press SPACE BAR to update
OpenBCI Dongle : FOUND
OpenBCI Board : FOUND
Robotic Arm : FOUND

(a) Main menu of the computer based BCI.



(b) Possible modes of BCI use.

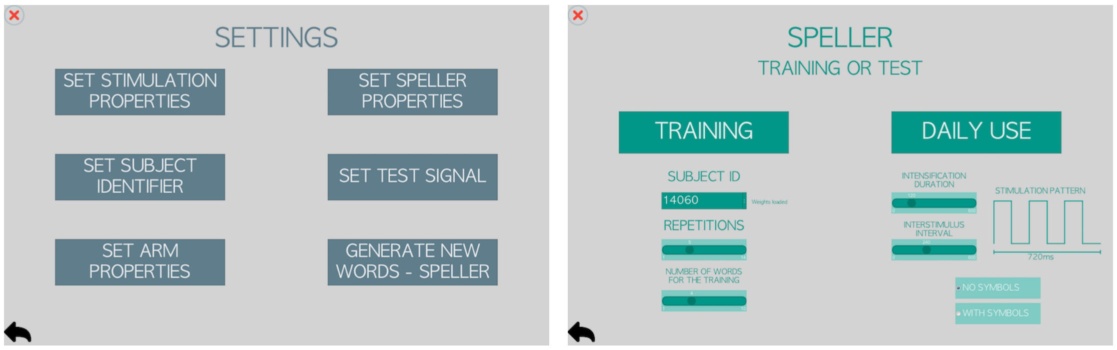
Figure 4.1: The main menu and the possible BCI modes of use of the computer based BCI.

Some screenshots of the developed BCI software are presented in Figure 4.1, 4.2 and 4.3. These windows are as follows:

A. Main Menu Window The main menu is shown in Figure 4.1a. This is the first screen presented to the user. The user can choose to start the system or to enter the menu to set some settings related to the parameters for the stimulation, to the robotic arm control, ... Furthermore, in the bottom left corner it is possible to check if the USB Dongle, the OpenBCI board and the robotic are connected or not.

B. Start Menu Window If the start button is chosen in the main menu, the screen visible in Figure 4.1b appears. This window presents to the user two modes of operation: *Speller* and *Robotic Arm Control*. If the *Speller mode* is chosen, the user can use a P300 speller similar to the one developed by *Farwell and Donchin* in [21]. The speller mode is also used for the training phase of the EEG classifier. This will be discussed in Chapter 5. The second possible option is the *Robotic Arm Control*. When chosen, the user gains access to a P300 based control of the robotic arm. So, according to the user's needs, it is possible to choose between the control of the robotic arm and the use of the speller.

C. Settings Window When entering the settings page, shown in Figure 4.2a, it is possible to define some options related to the BCI. For some of the provided options, a sub menu will be opened when the option is clicked. These options are related to: stimulation parameters, such as inter stimulus interval, intensification duration, number of repetitions; properties for the training of the classifier, such as the number of words to be used in the training session; the subject ID, necessary to save the EEG data; test signal on the board, which is useful to verify



SUBJECT ID 14060 Weights loaded
REPETITIONS 5
NUMBER OF WORDS FOR THE TRAINING 4
INTENSIFICATION DURATION 120
INTERSTIMULUS INTERVAL 240
[✓ NO SYMBOLS] WITH SYMBOLS

(a) Settings. **(b)** Settings for speller mode.

Figure 4.2: Main settings window and speller settings window of the computer based BCI..

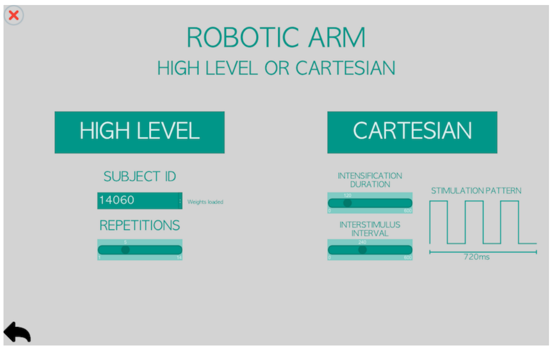
that the communication with the board is working; properties for the control of the robotic arm, such as the incremental values for the discrete movements.

D. Speller Mode Window The speller mode window is shown in Figure 4.2b. When entering this mode,it is possible to choose between the training mode and the daily use of the speller. In the training mode, the user is asked to spell predetermined letters. This is used in order to perform the training of the classifier, and will be explained in detail later in Chapter 5. While during the daily use of the speller mode, the user can spell any word he/she desires.

When entering the speller mode, the user can also change the default stimulation parameters. Among these parameters, there is the number of repetitions. Now, we explain what we mean with the term repetitions. In general, when dealing with evoked potentials, it is difficult to identify the presence of an evoked potential with only one presentation of the stimuli. Therefore, in order to identify the presence of an evoked potential with greater accuracy, the same stimuli are presented to the user multiple times. This way, by averaging the recorded responses, it is possible to identify the presence of the evoked potential with higher accuracy, avoiding the errors that could arise with just one presentation of the stimulus.

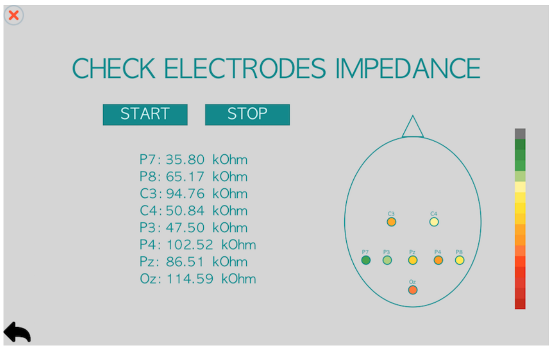
In our case, the term repetition refers to a complete set of flashes of a visual stimulation grid. If we consider the visual stimulation grid of *Farwell and Donchin*, that we have already shown in Figure 2.6, we have that a single repetition would be composed of 12 flashes, one for each column and each row.

E. Robotic Arm Mode Window The robotic arm mode window is shown in Figure 4.3a. When entering the robotic arm mode, the user is presented with the options to set some pa-



SUBJECT ID: 14060 Weights loaded
REPETITIONS: 5 (1–14)
INTENSIFICATION DURATION: 120 (0–600)
INTERSTIMULUS INTERVAL: 240 (0–600)

(a) Settings for robotic arm mode.



P7: 35.80 kOhm
P8: 65.17 kOhm
C3: 94.76 kOhm
C4: 50.84 kOhm
P3: 47.50 kOhm
P4: 102.52 kOhm
Pz: 86.51 kOhm
Oz: 114.59 kOhm

(b) Impedance Check.

Figure 4.3: The robotic arm settings and the impedance control windows in the computer based BCI.

rameters, such as number of repetitions and subject ID. Then, the user is also provided with an option to choose between high level control and cartesian control of the robotic arm. The difference between the two modes will be explained later in the chapter.

F. Impedance Control Window The last screenshot, visible in Figure 4.3b, shows the electrodes impedance check. It is very important to have good quality signals in order to achieve good accuracy in P300 detection. One way to check if the electrodes are placed correctly on the scalp is to measure the electrode-to-skin impedance. In order to measure the impedance, a small, and known, current is injected in the electrode, and the resulting voltage is measured. Since both voltage and current are known, it is then possible to compute the electrode-to-skin impedance as:

$$R = \frac{V}{I} \tag{4.1}$$

where V is the measured voltage, and I is the injected current. By using a color-based legend, the user can determine which electrodes have to be adjusted before starting to use the OpenBCI for recording.

4.1.1 Visual Stimulation

In this section, we describe in detail how we implemented the visual stimulation, both for speller and robotic arm mode. To elicit a P300 response, an oddball paradigm is frequently used. With this paradigm, various stimuli are presented to the user. From the user's point of view, one stimulus represents the *target*, i.e. the one the user is focused on. All the other stimuli are non-target. Considering all the stimuli presented to the user, the target stimulus occurs infrequently. When presented with this target stimulus, a P300 response is elicited [5].

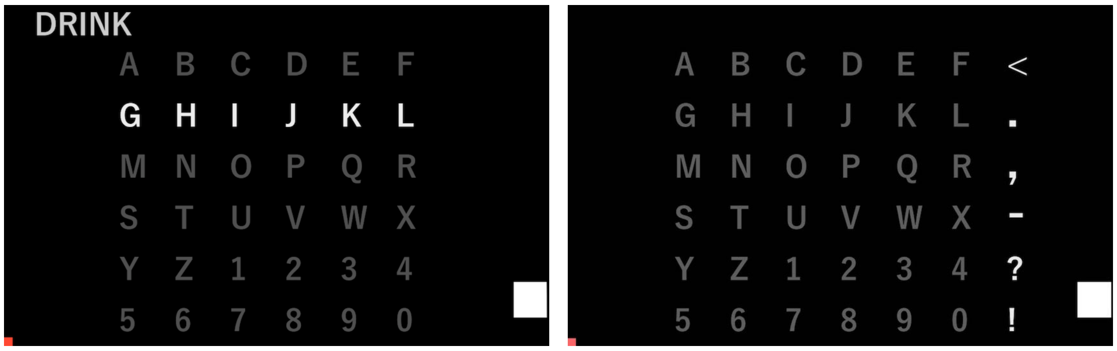


Figure 4.4: *Left*: speller stimulation grid without symbols. *Right*: speller stimulation grid with symbols.

For our wearable BCI device, we employed visual stimulation. Stimuli are arranged in a $n \times m$ grid. For speller mode and cartesian control of the robotic arm, we decided to flash columns and rows, instead of single cells, in order to reduce the time required to perform a complete stimulation. With this paradigm, the generation of a P300 response would be associated to the flashing of a row or of a column, rather than to the specific cell.

In the speller mode, it is possible to use two different types of grids: one grid is a 6×6 grid containing only letters and numbers; the other one is a 6×7, grid which, other than letters and numbers, also has 4 symbols (**.** **,** **?** **!**), a symbol to delete the last choice (<), and the space character (**-**). These two grids are shown in Figure 4.4.

For the robotic arm mode, two different control modes can be used, *cartesian control* and *high level control*. For the cartesian control, the grid is a 4×4 grid, in which 16 movements are represented. The 16 movements are the same as shown in the robotic arm control software shown in Section 3.2, and allow to perform discrete movements with the arm. A screenshot of the flashing grid is shown in Figure 4.5. Also in this case, columns and rows flash, and a single repetition is composed of 8 flashes.

In the high level control, instead, the subject is not asked to perform a series of discrete movements to complete a generic task. The idea behind the implementation of this mode is that the subject can directly perform a complete, and predefined, action with a single command. These commands are represented in Figure 4.5, and are as follows:

- Grab a glass of water: the robotic arm will move to a location, already defined in the firmware, in which a glass of water is expected to be present. After grabbing the glass, it will move towards the subject mouth;

- Grab food: as above, the robotic arm will perform a complete movement to a location

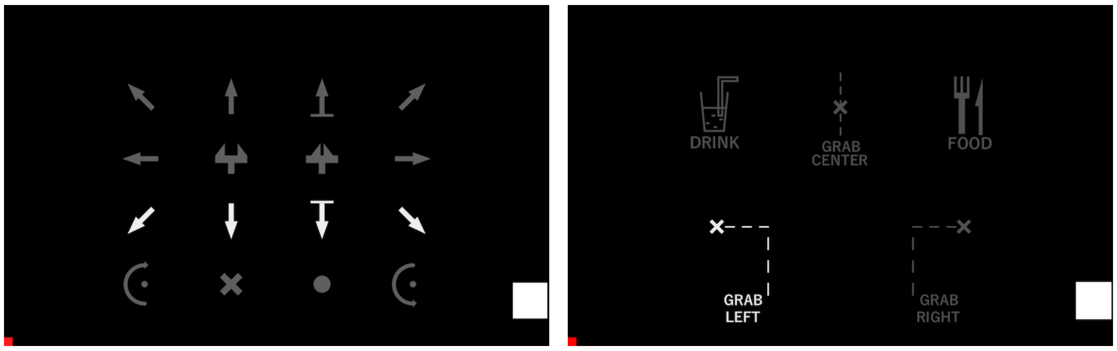


Figure 4.5: *Left*: robotic arm cartesian control stimulation grid. *Right*: robotic arm high level control stimulation grid.

where food is expected to be present. After picking up the food, the robotic arm will move towards the subject;

- Grab an object on the left: the idea behind this high level command is that if a person wants to give to the user an object, the robotic arm will move to a defined location on the left, and wait until an object is grabbed;
- Grab an object in central position: this movement is the same as above, but the robotic arm will move to a central location;
- Grab an object on the right: as above, but on the right.

Regarding the stimulation in high level robotic arm control, a single cell flashes, instead of rows and columns flashing as it occurs in the other BCI modes of use. So, a single repetition is composed of a single flash of each cell. While the robotic arm is performing one of these high level actions, the stimulation is stopped until the movement is complete.

With the current implementation of the robotic arm firmware, it is required that the objects are in a position already defined in the firmware.

As a default, stimuli are presented with this flashing pattern: a random column, row or cell intensifies with a white color for 120 ms. This intensification happens every 240 ms. These two parameters (intensification duration and inter stimulus interval) can be adjusted in the settings page . A study on the effect of the stimulus rate on P300 response was done by *McFarland et al.* [48]. They showed that the optimal inter stimulus interval lies between 62.5 and 250 ms.

Table 4.1: Description of the data packet sent by the OpenBCI board [4].

Byte(s)	Value	Byte(s)	Value
1	0xA0	18 - 20	Channel 6
2	1 - 255	21 - 23	Channel 7
3 - 5	Channel 1	24 - 26	Channel 8
6 - 8	Channel 2	27 - 28	Auxiliary
9 - 11	Channel 3	29 - 30	Auxiliary
12 - 14	Channel 4	31 - 32	Auxiliary
15 - 17	Channel 5	33	0xC1

4.1.2 EEG Data Retrieval

The 32 bit OpenBCI board that we used for our wearable BCI device makes use of bluetooth connection to communicate EEG data. A USB dongle has to be plugged in the laptop to retrieve the data. Then, in our software we implemented a serial communication with the USB dongle with a baud rate of 115200. Each data packet sent by the board is composed of 33 bytes. Table 4.1 provides the complete description of the data packet.

In order to parse the data contained in the data packet, the start and stop bytes, respectively 0xA0 and 0xC1, are used.

4.1.3 Synchronization Protocol

Synchronization between the recorded EEG and the stimulation time is a fundamental component for analyzing evoked potentials. If this condition is not met, then it would be impossible to correctly detect the evoked potential in the EEG signal, because time information would be misleading. As an example, with P300 responses we expect a positive deflection in the EEG signal at around 300 ms. It is easy to understand that, if the information about the stimulation time is incorrect, then it would be impossible to correctly identify the presence of a P300 response in the EEG trace.

Therefore, a synchronization protocol had to be set up. In our software, the synchronization protocol uses one of the available analog channels on the board to read the light activity generated by the flashing stimulation. In order to achieve this, we added a small square in the bottom right corner of the grid. This small square flashes at the same frequency and at the same time of the columns and rows. This square is visible both in Figure 4.4 and Figure 4.5.

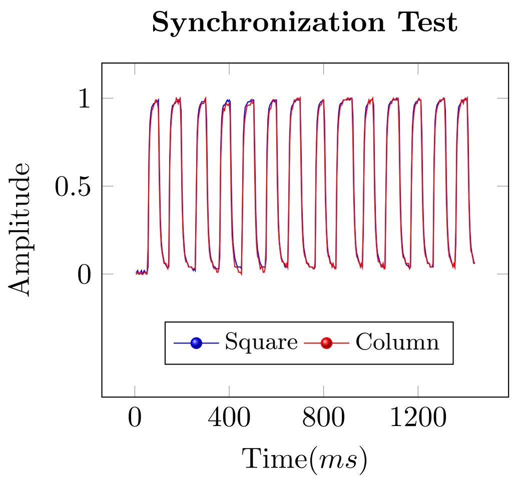


Figure 4.6: Signals recorded simultaneously from the flashing square and column. Signals recorded from the photoresistors are normalized in the range [0-1].

Using a simple photoresistor and an analog pin on the board, the photoresistor signal can be read directly with the OpenBCI Cyton board together with the EEG signal, and in this way a synchronization between the visual stimulation and the recorded EEG signal can be established.

In order to assure that the square flashed at the same frequency of the columns and rows, we performed the following test: we simultaneously recorded, using two photoresistors, the light signal of the square and of a flashing column of the matrix. The result is visible in Figure 4.6. The two signals overlap, meaning that the square and column are flashing exactly with the same frequency. So, this method represents a reliable technique to synchronize the EEG signal with the visual stimulation provided to the user.

4.2 Android Based BCI

In order for our device to be completely wearable, we developed a mobile application that can be used as a BCI software. The developed application expands the project “Progetto ON” [49]. This mobile application allows the BCI user to use the BCI assistive device wherever he/she is, without relying on the use of a laptop.

The operating system that we choose for our application is Android [50]. The mobile application reflects the functionality of the computer software described in Section 4.1, but provides additional features. The main functionality that is missing in the mobile application is the control of the robotic arm, that can be achieved only with the computer software.

In order for the OpenBCI board to communicate with an Android device, the hardware

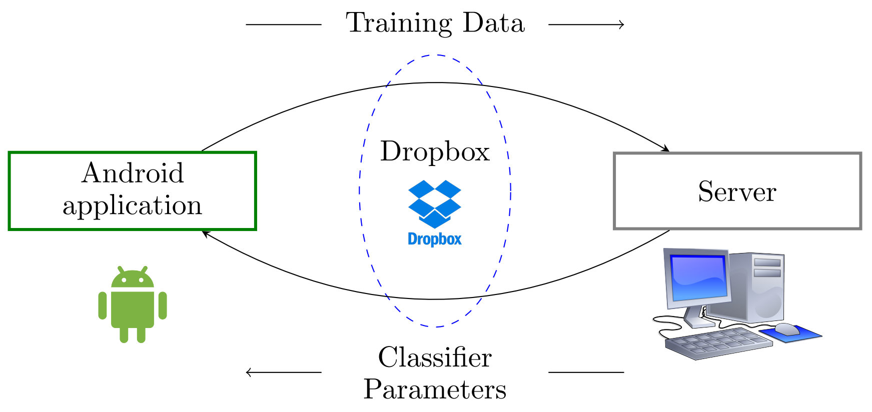


Figure 4.7: Interaction between the Android application and the remote server.

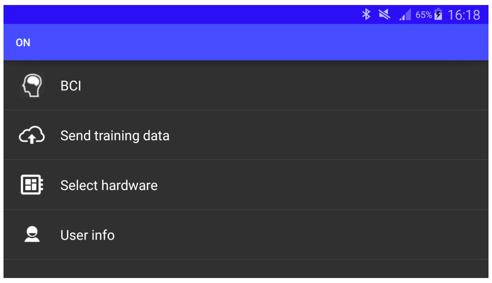
and firmware modifications described in Section 3.1.2 were required. As a reminder, with these modifications, not only does the OpenBCI board communicate with external devices through a USB dongle, but also through a HC05 Bluetooth module mounted on a PCB, which is shown in Figure 3.6.

In the same way of the computer software, the main features of the Android application are the following ones:

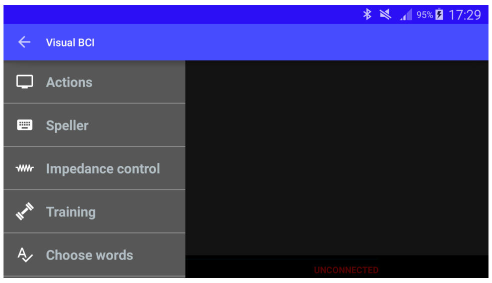
- acquisition of EEG signal streamed by the OpenBCI board through the HC05 Bluetooth module;
- presentation of visual stimuli in order to elicit a P300 response;
- processing of the recorded EEG signal and feedback to the user.

An additional feature that was implemented in the Android application is the communication with a remote server. This feature was required because the training process of the classifier used for P300 detection, that will be discussed in Chapter 5, can't be performed on a mobile device. Thus, the data saved during the training phase are sent to a remote server via Dropbox [51]. After the data have been received by the server, the classifier performs training on them. The parameters of the trained classifier are then uploaded on Dropbox, and can be retrieved on the mobile device. A schema representing this operation is shown in Figure 4.7.

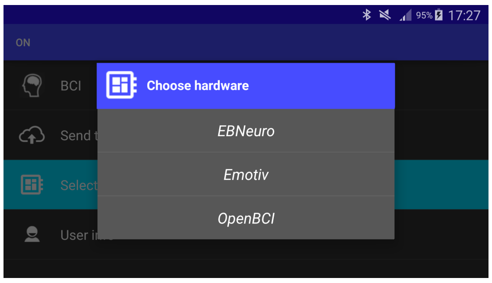
In Figure 4.8, some screenshots of the developed Android application are shown. Each of the shown windows is related to a main function of the application. Now, we list the most important ones.



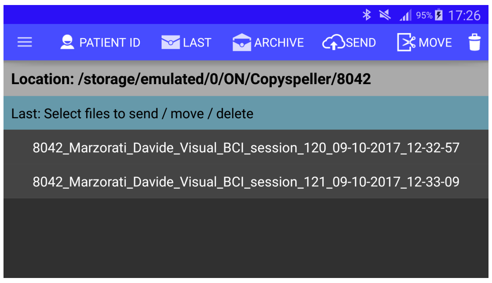
(a) Main menu of the application.



(b) BCI menu.



(c) Choose hardware dialog.



Location: /storage/emulated/0/ON/Copyspeller/8042

Last: Select files to send / move / delete

8042_Marzorati_Davide_Visual_BCI_session_120_09-10-2017_12-32-57

8042_Marzorati_Davide_Visual_BCI_session_121_09-10-2017_12-33-09

(d) Send training data menu.

Figure 4.8: Screenshots of the developed Android application.

A. Main menu In Figure 4.8a, the main menu of the application is shown. In this menu, it is possible to choose among four options: start the BCI, upload to the server the data saved during the training sessions; select the hardware that is being used for EEG recording, and set some general settings of the app.

B. BCI menu When the user selects the option *BCI* in the main manu, the window shown in Figure 4.8b is presented to the user. Through a navigation drawer positioned on the left side of the screen, the user can then enter modes related to different tasks . The modes among which the user can choose are: actions, speller, impedance control, training, choose words, view signals and settings. These options will be explained in detail later in this chapter.

C. Select hardware When the user selects the option *Select hardware*, the dialog shown in Figure 4.8c is presented to the user. Through this dialog, the user can select which type of hardware is being used for EEG recording. The available options are: EBNeuro [52], Emotiv [39] and OpenBCI [4]. This option was included so that, in the future, it will be possible to use the same Android application with different EEG acquisition systems.

At the time of writing, the fully supported EEG acquisition systems are two: EBNeuro [52], and the OpenBCI Cyton board [4]. Since the Emotiv Epoch device does not allow free access to the raw recorded EEG data, it is not supported by the developed Android application.

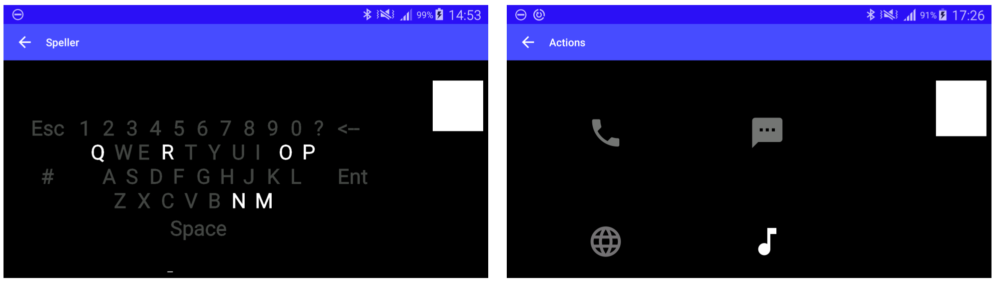
D. Send training data When the user selects the option *Send training data*, the window visible in Figure 4.8d is shown. In this window, after selecting the user for which the training data has to be uploaded to the server, it is possible to choose which sessions to upload to the server. In addition to the upload action, it is possible to perform several actions: see which sessions have been archived (i.e., those sessions that have been already uploaded to the server), delete sessions, and manually move sessions to the archive.

The main mode of use of the Android application is the one related to the BCI. As soon as the BCI mode is started, the application attempts to connect to the OpenBCI board (in particular, to the HC05 module connected to the board) via Bluetooth. On the bottom of the screen, the current status of the Bluetooth connection is constantly updated. If the connection attempt fails, every 5 seconds a new connection attempt is made. Then, through a navigation drawer positioned on the left side of the screen, it is possible to select other modes. Now, we explain each one of these modes in detail.

4.2.1 Online BCI

Two different main modes of online BCI are provided in the Android application. We named them *Speller* and *Actions*.

The *Speller* mode is shown in Figure 4.9a. This mode is exactly the same as the one implemented in the computer based BCI, which has already been described in Section 4.1.1. The *Speller* mode allows the user to compose words and sentences. The difference between the *Speller* mode implemented in the computer based BCI and the one implemented in the Android based BCI is related to the layout of the virtual keyboard. In particular, in the computer software, the grid is arranged in a matrix of dimensions equal to 6 rows and 6 or 7 columns (according to the chosen mode). Furthermore, the flashing is performed with rows and columns flashing. For the Android application, instead, the virtual keyboard is the one shown in Figure 4.9a, which is as much similar as possible to a real keyboard. The flashing is no longer performed with rows and columns flashing, but rather with a flashing of groups of letters/numbers. The groups are composed in such a way that they reflect the structure of the computer software speller matrix. As an example, in Figure 4.9a the letters *M,N,O,P,Q,R* are flashing. These letters compose the row number 3 of the speller grid of the computer based BCI, as visible in Figure 4.4.



(a) Speller mode. **(b)** Actions mode.

Figure 4.9: The two modes of use of the online BCI in the Android application.

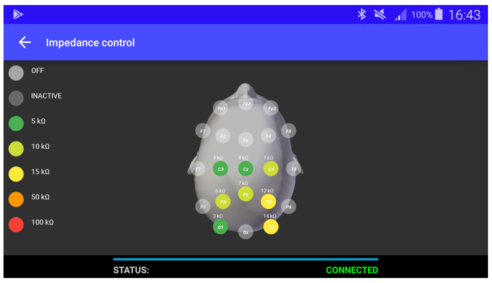
The *Actions* mode, instead, allows the user to perform high level actions. The idea behind this mode of use is that the user can control the smartphone through the P300 BCI. With the current version of the application, only four actions can be selected by the user.

As for the computer based BCI, synchronization between the EEG signal and the visual stimulation is fundamental. In order to achieve this synchronization, the same protocol used for the computer software has been employed in the Android application. During a P300 visual stimulation, a small square flashes with the same frequency of the symbols on the screen. A photoresistor connected to the OpenBCI board detects the light generated by the flashing of square, and allows to achieve the synchronization.

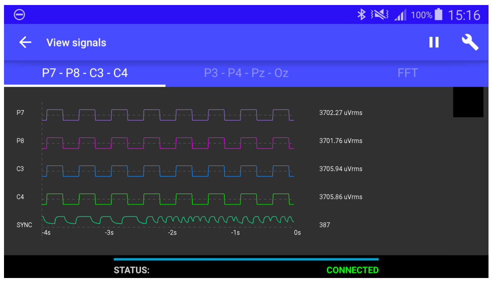
4.2.2 Visualization

In the Android application, three main modules of data visualization, shown in Figure 4.10, are provided.

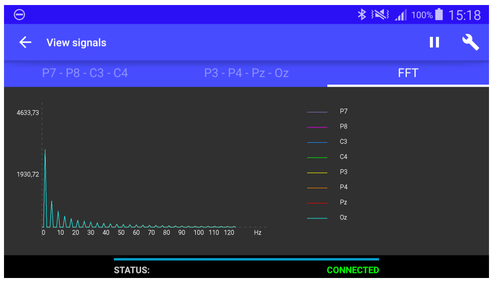
The first module is related to the impedance control of the electrodes. When the user enters the impedance control mode, the window shown in Figure 4.10a is presented. In this module, the impedance of each electrode is shown both with a numeric value and in a color-coded fashion. The way in which impedance is computed has been already explained in Section 4.1. Since the OpenBCI Cyton board has only 8 channels available for recording, it is necessary to map these 8 channels to 8 locations of the 10-20 system for EEG recording [13]. This mapping can be performed in the Settings menu of the app, that will be discussed later in Section 4.2.4. The EEG recording locations that are not mapped to any OpenBCI channel will be shown as *OFF* in the impedance control window. Furthermore, it is possible to set a mapped channel as active or inactive. The mapped channels that are not active are not taken into consideration in all the BCI modes, and are shown as *INACTIVE* in the impedance control window.



(a) Impedance control.



(b) View signals.



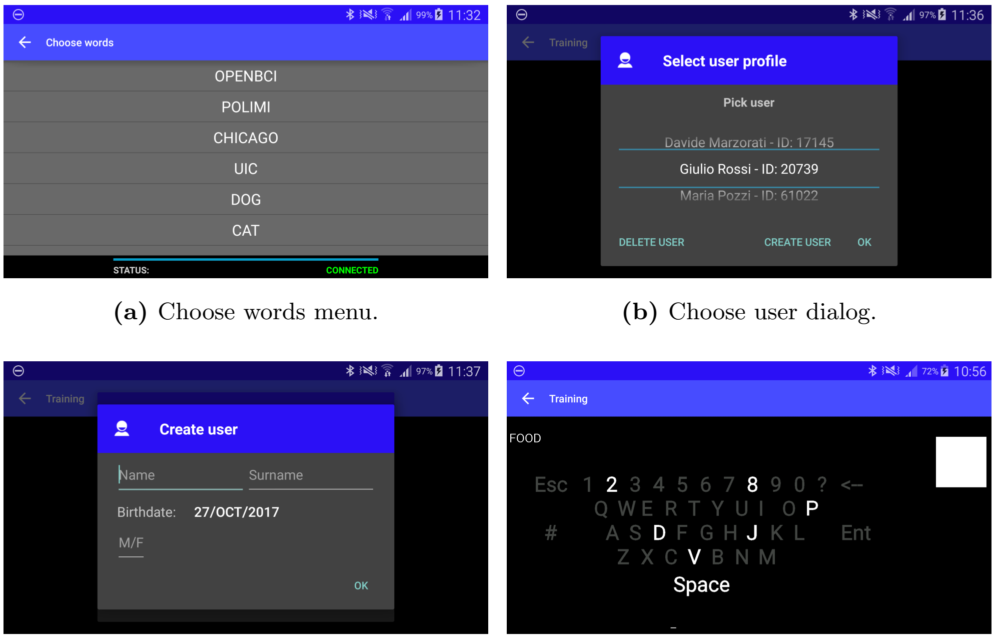
(c) Power spectrum.

Figure 4.10: The three modules of data visualization in the Android application. For visualization purposes, test signals were applied to the channels.

The second module is related to the visualization of the signals recorded by the OpenBCI board. When the user enters the view signals mode, the window shown in Figure 4.10b is presented. The window is composed by three different tabs. Each tab can be selected either by swiping (right to left and left to right) or by clicking on the tab title. The first two tabs show the data of the first and last four channels, respectively. The last tab, which represents the third module of the data visualization, shows the real-time power spectrum of the active channels.

In the top bar of the view signals window, two buttons are present. The actions that can be accessed by clicking on these buttons are:

- start/stop the streaming of data. If the board is not currently streaming data, a *play* button is shown. Otherwise, a *pause* button is shown;
- open a dialog, where the user can then select to:
  - apply test signals to the channels, so that it is possible to be sure that data transmission is working fine;
  - apply a notch and/or a band pass filter to the data. The available choices for the notch filter are 50 and 60 Hz, while for the band pass filter are 1 - 40, 5 - 40, 15 - 40, 7 - 13 and 0.5 - 30 Hz;



(a) Choose words menu. **(b)** Choose user dialog.

(c) Create user dialog. **(d)** Training stimulation.

Figure 4.11: Windows of the Android application related to the training session.

– select the maximum frequency to be displayed in the power spectrum plot.

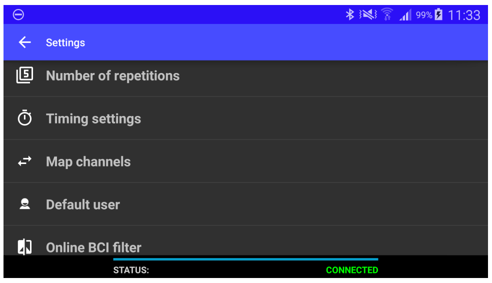
For each channel, other than the signal trace, more information is provided. In particular, the root mean square value is shown on the right of the trace, and a feedback is shown if the signal is railed or near railed.

With the impedance control and view signal modules, the user can understand if all the electrodes are placed correctly on the scalp. This way, it is possible to adjust their positioning before the beginning of a training session or of the online use of the BCI.

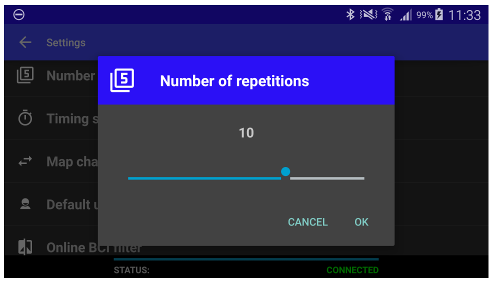
4.2.3 BCI Training

If the user selects the *training* option, a P300 training sessions starts. The procedure of the training session is equal to the one performed in the computer based BCI.

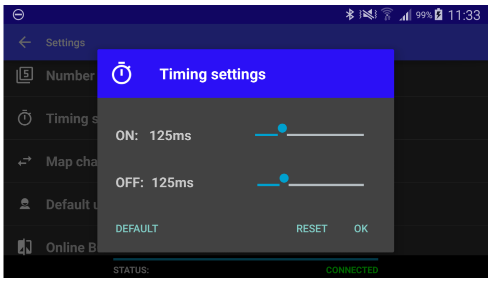
In the training session, the user is asked to spell predefined letters, that can be chosen in the *Choose Words* option of the menu, as visible in Figure 4.11a. The maximum number of words for a single training session is equal to 10. Before the beginning of a training sessions, the dialog shown in Figure 4.11b is presented to the user. This dialog allows to select for which user the training is being performed. If the current user has never undergone a training session, it is possible to create his/her profile by clicking on *Create new user* option. If this is the case, a



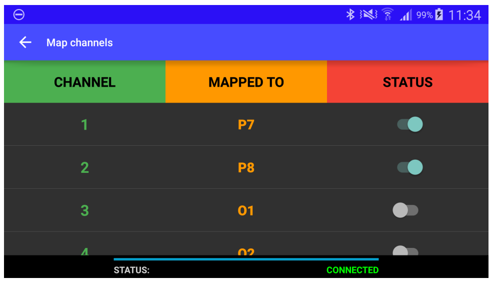
(a) Settings main page.



(b) Repetitions dialog.



(c) Timing dialog.



CHANNEL	MAPPED TO	STATUS
1	P7	●
2	P8	●
3	O1	○
4	O2	○

(d) Channels mapping menu.

Figure 4.12: Settings page and settings dialogs of the Android application.

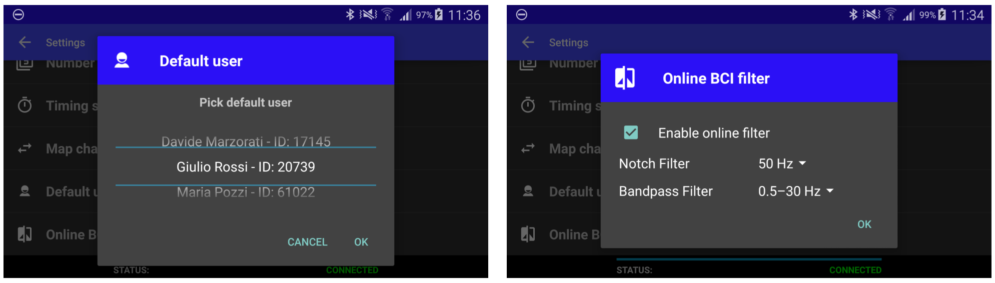
new dialog, shown in Figure 4.11c, is shown. Basic information about the user are asked. These information will then be included in the EDF file used to save the EEG data. EDF stands for European Data Format and it is a simple and flexible format for exchange and storage of multichannel biological and physical signals [53].

During the training session, the letter on which the subject has to focus is highlighted with a red circle.

At the end of the training session, an EDF file is saved. This file contains EEG data and the photoresistor signal, necessary to achieve the synchronization between the EEG and the visual stimuli. In addition to the EDF file, a text file is saved. This file contains information about the stimuli that were provided to the user. These two files are the ones that the classifier needs to perform the training, and they are the files that have to be sent to the server via Dropbox. For each user, the files are saved in a different folder on the mobile device.

4.2.4 Settings

If the *settings* option is chosen, the user can adjust several default settings related to the visual stimulation, to data processing and to the OpenBCI board. The settings main window is shown in Figure 4.12a. In particular, in the settings page the user can:


Settings

Default user

Pick default user

Davide Marzorati - ID: 17145

Giulio Rossi - ID: 20739

Maria Pozzi - ID: 61022

CANCEL OK

Online BCI filter

✓ Enable online filter

Notch Filter 50 Hz

Bandpass Filter 0.5–30 Hz

OK

STATUS: CONNECTED

(a) Default user dialog. **(b)** Real time filter dialog.

Figure 4.13: Default user and online BCI filter dialogs of the Android application.

- set the number of repetitions to be used in the visual stimulation, as shown in 4.12b . We remind here that the term *repetition* stands for a complete sequence of flashes. So, if the virtual keyboard used for the speller contains 36 items arranged in 6 columns and 6 rows, and 6 symbols at a time flash, a repetition would then be composed by 12 flashes (6 for the rows and 6 for the columns);

- set the stimulation pattern used for the visual stimulation. By default, a single flash lasts for 250 ms, with a 50% duty cycle, as shown in Figure **??**. The user, if he/she wants to, can change this pattern by modifying the total duration of the flash and the duty cycle. This is possible with the dialog shown in Figure 4.12c. In this dialog, an option is provided in order to restore the flashing pattern to the default one;

- set the mapping between the OpenBCI board channels and the 10-20 EEG locations. This is necessary because the OpenBCI board has only 8 channels available, while the 10-20 standard for EEG recording defines 21 locations on the scalp. So, the user can set the location of each channel: this information is then used in the impedance control mode and also for labeling purposes when saving EEG data. Furthermore, in the channels mapping window, shown in Figure 4.12d, it is possible to turn *ON* and *OFF* each channel;

- set the default user of the BCI. The default user is the one whose classification parameters are used in the online BCI mode. When clicking on this option, the dialog shown in Figure 4.13a appears, allowing to select, among the active users, the default one;

- adjust the settings for the filtering applied to the EEG signals during the online use of the BCI. When clicking on this option, the dialog represented in Figure 4.13b is presented to the user, so that it is possible to enable or disable the real-time filter. If the filter is

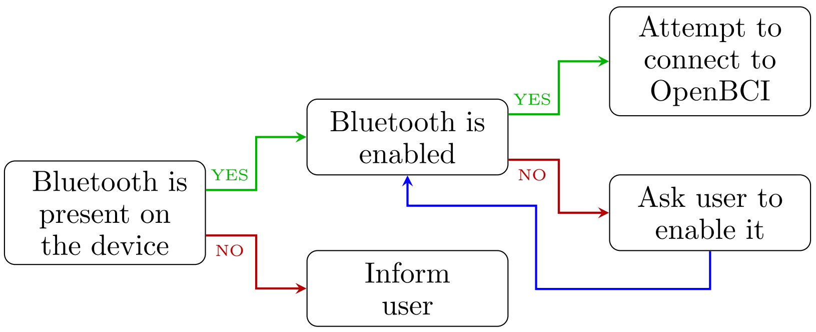

Bluetooth is present on the device
YES
NO
Bluetooth is enabled
Inform user
YES
NO
Attempt to connect to OpenBCI
Ask user to enable it

Figure 4.14: Bluetooth communication process in the Android based BCI. The first thing that is checked by the `CommunicationThread` is whether the device (smartphone or tablet) features a Bluetooth module. Then, it checks if Bluetooth is activated. Lastly, it attempts to connect to the OpenBCI Cyton board .

enabled, it is possible to select which kind of notch and band-pass filters have to be applied on the signals.

4.2.5 Main Modules

In this section, we are going to describe with more detail how the main components of the Android application were implemented.

Bluetooth Communication The main component of the application is a Class extending the Android base Class `Thread` [54]. This Class, that is called `CommunicationThread`, takes care of the communication with the OpenBCI board.

As soon as the user clicks on the BCI item of the main menu of the application, the `CommunicationThread` checks whether or not the Bluetooth module is available on the device. If this is the case, it checks if the Bluetooth functionality is activated or not. If the Bluetooth functionality is not activated on the device, it allows the user to directly activate it within the app. After checking that the device features a Bluetooth and that the Bluetooth functionality is activated, the `CommunicationThread` tries to establish a connection with the OpenBCI board, in particularly with the HC05 module connected to the board. A schema representing this process is shown in Figure 4.14.

The `CommunicationThread` keeps track of the status of the Bluetooth connection in a continuous way. For this reason, it is possible to show a feedback to the user regarding the status of the Bluetooth connection.

Data Buffer If the connection to the OpenBCI board is successful, the `Communication Thread` is ready to read the data sent by the OpenBCI board and also to send commands

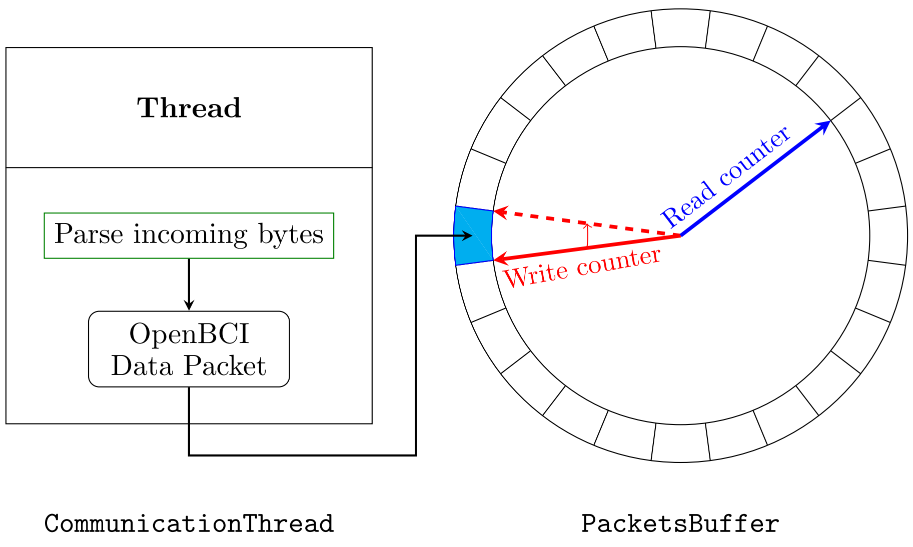


Figure 4.15: The circular data buffer implemented in the Android application. Each block of the circle contains a data packet, which have been inserted into the buffer by the `CommunicationThread`. When a new packet is received from the OpenBCI board, the write counter increments by one. When a packet is read from the buffer by any component of the application, the read counter increments by one.

to the board.

When the OpenBCI board starts streaming data, the `Communication Thread` takes care of parsing the received bytes into data packets, that have the structure shown in Table 4.1. When a data packet is complete, it is inserted in a circular `Buffer`, that is represented in Figure 4.15. We will refer to this `Buffer` as the `PacketsBuffer`. The size of the `PacketsBuffer` is equal to 500 packets, so it is able to store at maximum two seconds of data, since the sampling frequency of the OpenBCI board is equal to 250 Hz. All the other components of the application get access to the data through the `PacketsBuffer`, so it can be considered the most important component of the application.

The way in which a circular `Buffer` works is quite simple. The `Buffer` starts empty, and whenever a data is written into it, an index, that in Figure 4.15 is called the `WriteCounter`, is incremented by one. This increment is necessary in order to keep track of the last position of the buffer where a data was written. Another important index is the `ReadCounter`, that is incremented by one whenever a data is read from the buffer. This increment is necessary to keep track of the last position of the buffer where a data was read by a component of the app, so that when a new packet is requested, the correct one is provided.

A buffer overflow could occur when the `WriteCounter` goes beyond the `ReadCounter`. Therefore, the values of the `WriteCounter` and of the `ReadCounter` are always checked to determine

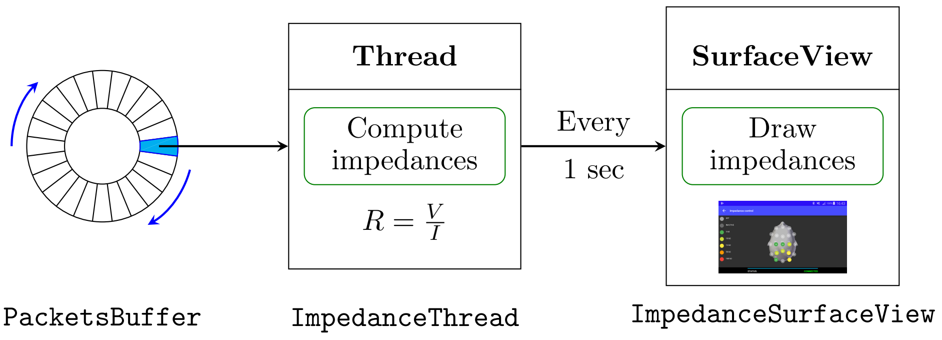


Figure 4.16: The impedance computation process in the Android application. The `ImpedanceThread` gets fresh new data from the `PacketsBuffer`, computes the impedances and sends them to the `ImpedanceSurfaceView` with an update rate of one second .

if an overflow of the buffer occurred.

The `PacketsBuffer` also performs filtering on the data, according to the settings specified by the user. The filters that are applied to the data are all 2nd order IIR Butterworth filters. Furthermore, during the training session, the `PacketsBuffer` takes care of saving the data sent by the OpenBCI board in an EDF file.

Data Visualization As we have already shown in Section 4.2.2, in the Android application it is possible to visualize EEG data in three different ways: impedance control, time series and real-time power spectrum. Each one of these visualization modes relies on two Classes: one Class extending the Android base Class `Thread` and another Class extending the Android base Class `SurfaceView` [54].

For the impedance control mode, the interactions between the `Packets Buffer`, `Impedance Thread` and `ImpedanceSurfaceView` are shown in Figure 4.16. The `ImpedanceThread` takes care of getting the OpenBCI data packets from the `PacketsBuffer`, computing the impedance values with Equation 4.1 and sending these values to the `ImpedanceSurfaceView`. The impedance values are updated on the screen every second, so after the `ImpedanceThread` have processed 250 data packets sent by the OpenBCI board.

The `ImpedanceSurfaceView` takes care of providing to the user the values of the electrodes impedance. In particular, the values of the electrodes impedance are shown in the correct locations of the 10-20 system with a color-coded legend and with the exact value. By performing a long click on the `ImpedanceSurfaceView`, it is also possible to turn the OpenBCI board channels on/off.

For the visualization of EEG data as time-series, the process is similar to the one explained

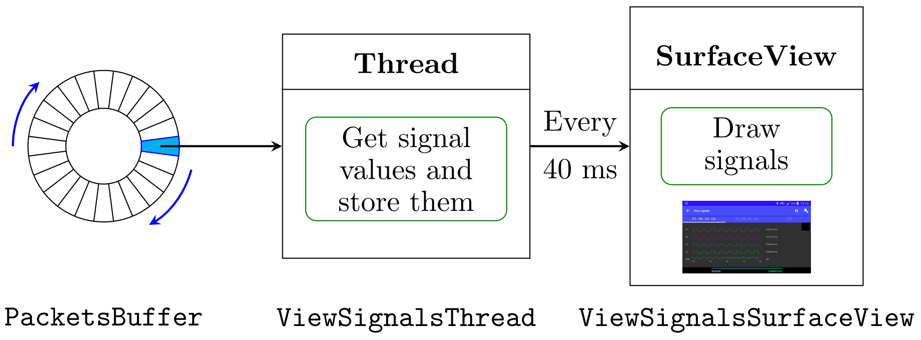


Figure 4.17: The signals visualization process in the Android application. The `ViewSignalsThread` gets the data from the `PacketsBuffer`, stores the values and sends them to the `ViewSignalsSurfaceView` every 40 ms.

for the impedance control mode. The `ViewSignalsThread` gets the OpenBCI data packets from the `PacketsBuffer`. In order not to refresh the screen with a very high frequency, the `ViewSignalsThread` sends the signals values to the `ViewSignalsSurfaceView` when 10 new packets have been read. This corresponds to a time interval of approximately 40 ms, since a new packet is received every 4 ms.

Furthermore, the `ViewSignalsSurfaceView` allows to visualize the real-time power spectrum of the EEG data. Upon receiving the new EEG data from the `ViewSignalsThread`, the `ViewSignalsSurfaceView` computes the power-spectrum of the active EEG channels and plots it on the screen. The maximum frequency that is displayed in the power-spectrum plot has a default value of 120 Hz. This value can be changed by entering the settings in the view signals mode. The available options of maximum frequency values are: 20, 40, 60, 80 and 120 Hz.

Visual Stimulation Another component of the Android application is the visual stimulation necessary for P300 elicitation. The visual stimulation is controlled by two Classes: one Class extending the Android base Class `Thread` and a Class extending the Android base Class `SurfaceView` [54].

As an example, for the BCI *Speller*, there is a `SpellerThread` that controls the precise timing required by the visual stimulation. This way, it is possible to correctly update what is shown on the screen by redrawing the `SpellerSurfaceView`, therefore correctly stimulating the user and achieving the best possible synchronization between the visual stimulation and the EEG recording.

During the visual stimulation, other processes are involved. In particular, if the user is undergoing a training session, the EEG data and the photoresistor data are saved in an EDF

Table 4.2: Structure of the text file saved during a generic training session.

Trial Number	Stimulation Number	Stimulation Code	Target	Time
1	1	7	0	7890
1	2	3	1	8132
...	...	...	...	...
1	60	8	1	22310
2	1	3	0	24210
...	...	...	...	...

file [53]. Other than this EDF file, a text file containing information about the stimulation is saved. This text file is necessary to label the EEG epochs as target or non-target epochs during the training of the classifier.

The structure of the text file saved during a training session is shown in Table 4.2. The numbers in the first column of the text file represent the number of the trial5. If a training session is composed by 8 letters, the trial number would range from 1 (first letter) to 8 (last letter). The second column contains the stimulation number. The stimulation number cycles for all the trials from 1 to $N_{REP} \times N_{FLASH}$, where N_{REP} is the number of repetitions chosen for the visual stimulation and N_{FLASH} is the maximum number of stimuli provided to the user. As an example, in the case of the *Speller* mode of use, we have that N_{FLASH} is equal to 12, since in a single repetition each one of the 6 columns and each one of the 6 rows flashes. So, if 5 repetitions were chosen, for each trial the stimulation number would range from 1 (first stimulation) to 60 (last stimulation). The third column contains the stimulation code, which is simply the number associated to the row or column that flashed. In the speller grid, the columns are ordered from 1 to 6, while the rows are ordered from 7 to 12. The fourth column is a boolean flag to determine if that stimulus has to be considered as target or non-target. The last column provides information about the time at which that stimulation occurred.

Considering the example shown in Table 4.2, it is possible to understand that, during trial number 1, the user had to focus on letter *I*, since this letter lies at the intersection between column number 3 (stimulation code: 3) and row number 2 (stimulation code: 8).

If the user is not undergoing a training session, but the visual stimulation is performed for an online use of the BCI, many other processes are involved. In brief, during the online use of the BCI, there is a `DataProcessingThread` that analyzes the photoresistor signal in order to detect if a stimulation occurred. Then, the EEG signal is segmented and a one second long epoch is extracted, and saved in a circular `Buffer` designed to store epochs data. Once the trial is completed, all the epochs are classified.

Chapter 5

EEG Classification

Detection of P300 event-related potentials with high accuracy is a fundamental component of a P300 based BCI. Therefore, classification of EEG signal is required to detect the P300 responses.

A classifier is a function that maps generic input data to a category. In our case, the input data are EEG signals, and the categories are presence or absence of P300 response. A *training* set is a set of data previously labeled into categories. Using the *training* set, the classification models generate predictive rules that would be used to determine the category of unclassified data [55]. This process is referred to as training of the classifier. After the training phase, the classifier is tested on a *validation* data set. The *validation* data set is composed of classified examples that were not used during the training of the classifier.

In mathematical terms, for a classification problem the training set is made up of m previously classified examples, $(\mathbf{x_i}, y_i), i \in M$. The elements of the vector $\mathbf{x_i} \in \Re^n$ represent the values of the n predictive attributes for the ith example; $y_i \in H$ defines the category correspondent to the vector $\mathbf{x_i}$. Let us define the *hypotheses* function F as the class of functions $f(\mathbf{x}) : \Re^n \mapsto H$. This class of functions F represents possible relationships between y_i and $\mathbf{x_i}$. In a generic classification problem, the aim is to define an appropriate hypothesis space F and an algorithm A_f, that, together, would allow to determine a function $f^* \in F$ that specifies the relationship between the value of the n predictive attributes and the corresponding categories [55].

If we define α as a parameter, or a vector of parameters, the function f is dependent on, the optimal classifier is the function $f^*(\mathbf{x}, \alpha_0)$, where α_0 is the parameter or the vector of parameters that minimizes the loss function $R(\alpha)$:

$$\begin{aligned} R(\alpha) = E[|y - f((\mathbf{x}, \alpha))|] &= \int |y - f(\mathbf{x}, \alpha)| dp(\mathbf{x}, y) \\ &= \int \int |y - f(\mathbf{x}, \alpha)| p(\mathbf{x}, y) d\mathbf{x} dy \end{aligned} \tag{5.1}$$

This loss function depends on the parameter or the vector of parameters α, but also on the distribution $p(x, y)$, which is a prior unknown. Therefore, the loss function is approximated with the expected mean value of the error on the training set. Equation (5.1) then becomes:

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} |y_i - f(\mathbf{x_i}, \alpha)| \tag{5.2}$$

We can see from Equation (5.1) and (5.2) that $R_{emp}(\alpha) \xrightarrow{N \to \infty} R_\alpha$, so it could be a good choice to approximate $R(\alpha)$ with $R_{emp}(\alpha)$. However, for a finite number N, the difference between the terms $R_{emp}(\alpha)$ and $R(\alpha)$ can be very high. This could lead to a phenomenon referred to as *overfitting*, that occurs when the chosen classifier has a perfect performance on the training set, but it is not able to generalize and correctly classify future examples of $\mathbf{x}$.

To avoid the problem of *overfitting*, k-fold cross validation is typically used. With k-fold cross validation, it is guaranteed that each observation $\mathbf{x_i}$ is used the same number of times in the training set and only one time in the test set. The complete dataset is divided into k disjoint subsets, and at the jth iteration of the procedure, one of the kth subsets is used as the *validation* set, while the union of the other subsets is used as the training set. At the end of the cross-validation procedure, the accuracy of the classifier is estimated with the average value of the accuracy on the k *validation* sets [56].

5.1 Logistic Classifier

In our application, we use logistic regression as the classification technique to detect P300 responses in the EEG signal. In logistic regression, the binary response y to a vector $\mathbf{x}$ (of n values) is explained using a Bernoulli distribution with $E(y) = P(y = 1) = p$. Considering the *ith* example, $E(y_i|X = \mathbf{x_i}) = P(y_i = 1|X = \mathbf{x_i}) = p_i$ is a number between 0 and 1. Therefore, considering a cumulative distribution function (CDF), F, it is possible to model p_i as $F(\beta_0 + \beta \mathbf{x_i})$. Any F can establish a relationship between p and $\mathbf{x_i}$. In the case of logistic regression, the used CDF is the logistic distribution. For logistic regression, the operator $log\frac{p}{1-p}$ is called the *logit* operator.

Therefore, for logistic regression the following equations are valid:

$$y_i \sim Ber(pi) \tag{5.3}$$

$$logit(p_i) = log\frac{p_i}{1-p_i} = \beta_0 + \beta\mathbf{x_i} \tag{5.4}$$

Upon estimating the parameters β_0 and β, the probability p_i can be computed as:

$$p_i = P(y=1|X=\mathbf{x_i}) = \frac{1}{1+e^{\beta_0+\beta\mathbf{x_i}}} \tag{5.5}$$

$$1-p_i = P(y=0|X=\mathbf{x_i}) = \frac{e^{\beta_0+\beta\mathbf{x_i}}}{1+e^{\beta_0+\beta\mathbf{x_i}}} \tag{5.6}$$

Therefore, it is possible to assign the vector $\mathbf{x_i}$ to category 0 or 1 according to the value of the computed probabilities.

The vector β can be found by maximizing its log likelihood:

$$L(\beta) = \sum_{i=1}^{N} logP(y_i|\mathbf{x_i}, \beta) \tag{5.7}$$

In practice, Equation (5.7) is rarely used, and a penalty term is added to avoid high values of β. Equation (5.7) then becomes:

$$L(\beta) = \sum_{i=1}^{N} logP(y_i|\mathbf{x_i}, \beta) - \lambda\|\beta\|^2 \tag{5.8}$$

5.2 Genetic Algorithms

The first step in applying classification is feature selection. Performing feature selection by hand can be sub-optimal, and cumbersome. For this reason, we used the genetic algorithm presented by *Dal Seno et al.* in [57] to perform automatic feature selection for P300 response detection.

In this section, we'll introduce genetic algorithms in a general terms, explaining the required terminology, and in the following section we'll describe the specific genetic algorithm that we have used for automatic feature extraction. In this description, we will refer to the generic structure of genetic algorithms, represented in Figure 5.1.

In nature, survival of the fittest happens as a result of the processes of selection, crossover, and mutation among the individuals of the population. Genetic algorithms, which belong to the family of evolutionary algorithms, mimic this natural process in order to identify an optimal solution to a presented problem starting form a population of candidate solutions.

In a genetic algorithm, a solution is represented as a fixed-length string of values, describing an individual in a population. Each value of the string is related to a specific characteristic of

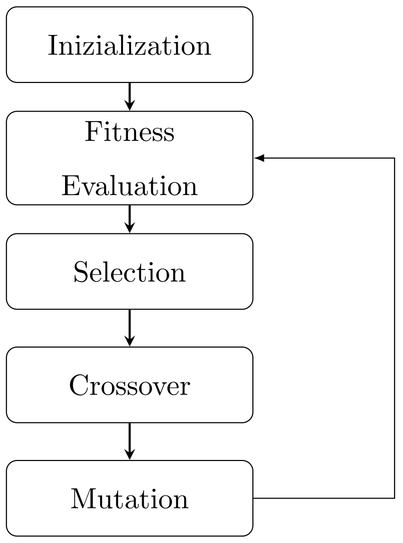


Figure 5.1: Generic structure of a genetic algorithm.

the individual, and the overall combination of the characteristics determines the solution to a problem. The key elements in a genetic algorithm are *individuals* and *populations*, that represent a collection of individuals.

Individuals

In nature, an individual (i.e., an organism) is characterized by a set of rules that describe completely the individual. These rules are encoded in the *genes* of the organism, and these genes are connected together to form the *chromosomes* of the organism. The genes are typically referred to as the *genotype* of the organism, while the physical expression of the genotype, which represents the organism, is referred to as the *phenotype* [58].

In a genetic algorithm, candidate solutions to the presented problem are represented by individuals. Each individual is characterized by two representations of the solution:

1. chromosome, that describe the solution in terms of bit or values;
2. phenotype, that represent the solution in the correct terms as defined by the problem. The phenotype space is linked to the solution space by a morphogenesis function, that associates the genotype to the phenotype.

Genes

The *genes* represent the basic building block in the development of a genetic algorithm. A gene describes a possible solution to the problem as an arbitrary length string of values. The structure of this string is defined in terms of parameters that are necessary to map the genotype to the phenotype.

Fitness

In genetic algorithms, *fitness* measure is used to evaluate each chromosome (i.e., each solution). To do so, a chromosome must be firstly decoded, and then an objective function has to be defined to compute the fitness value.

Population

A population is made up of individuals, that are tested in order to find the optimal solution to the presented problem. Two important characteristics of the population have to be taken into consideration: the initial population and the size of the population. The population size depends on the complexity of the problem. The larger the population is, the easier it is for the genetic algorithm to explore the solution space. But, having a very large population, even though it helps in reaching global optimum, results in more computational cost, memory and time. The typical value of population size is around 100 individuals [58]. Regarding the initial population, that is often randomly initialized at the beginning of the genetic algorithm as visible in Figure 5.1, it should present a large gene pool so that it is possible to explore the whole solution space. For this reason, the population is typically chosen using random techniques. If an heuristic method is used to seed the initial population, the resulting average fitness of the population is already high, and the genetic algorithm will converge to an optimal solution in a faster way.

Breeding

The breeding process represents the fundamental component of the genetic algorithm. During this process, new individuals, ideally with a better fitness, are created. The main steps for the breeding process are shown in Figure 5.1, and are:

- Selection of parents;
- Crossover of parents and creation of children individuals;
- Mutation of the children individuals;
- Replacement of the old individuals with the newly generated ones

We describe here in detail these steps:

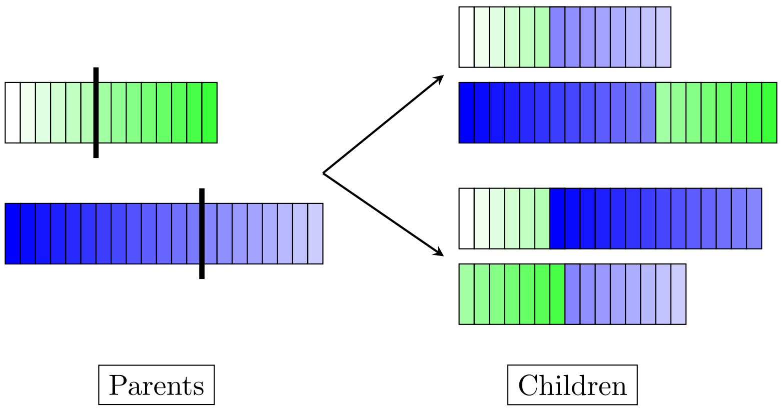


Figure 5.2: Crossover process in genetic algorithms.

1. **Selection**: it is the process of picking two parents from the old population in order to recombine them. The idea at the basis of selection is to choose individuals with a high fitness, with the hope that the generated children will have, in turn, an higher fitness than their parents. Typically, the higher the fitness of an individual, the higher the probability of that individual to be chosen. We can define as *selection pressure* the degree to which the fitter individuals are favored in the selection process. The selection pressure drives the convergence rate of the genetic algorithm: higher selection pressure results in higher convergence rate. A problem, that could result when choosing a high selection pressure, is that it is more probable that the genetic algorithm will converge to a local optimum [58]. Instead, if a low selection pressure is chosen, then the resulting convergence rate would be very low. *Elitism* is a process that takes part in the selection. It represents the practice of keeping the best chromosomes from the old generation, and transfer them to the new generation. In fact, if these individuals are not selected, or if crossover or mutation destroy them, they would get lost.

2. **Crossover**: after selection is performed, crossover takes place. Selection makes clones of the fitter individuals, but doesn't create new ones. With crossover, recombination of the individuals is performed in order to generate children with higher fitness. The simplest way to perform crossover is with single point crossover, which is shown in Figure 5.2. Three steps can be identified in this crossover operation: firstly, two individuals are chosen at random from the mating pool; secondly, two random points in the genes' strings are chosen; thirdly, the resulting parts are recombined together to form two children, by exchanging the sections after the cut point. An important parameter in the crossover operation is the probability that an individual would be selected to undergo crossover: if this value is

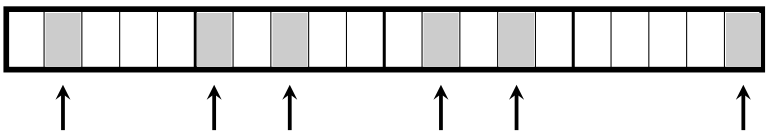


Figure 5.3: Mutation process in genetic algorithms.

0%, then the new population would result in a copy of the old generation; if it is equal to 100%, then all the old individuals undergo crossover. An intermediate value helps in having a good diversity in the population, by creating new individuals as well as keeping some of the old ones.

3. **Mutation**: after crossover is performed, mutation, shown in Figure 5.3, occurs. The purpose of mutation is to help the genetic algorithm to explore the solution space, by recovering lost genetic material and, also, to disturb the genetic information. Due to the process of mutation, the genetic algorithm is prevented from being stuck in a local minimum. Mutation works by changing the value of each gene with a predetermined, and typically small, probability.

4. **Replacement**: the last stage of the breeding process is the replacement. Replacement is necessary because, since the population size is constant, it's not possible that both parents and children take part in the new population. Therefore, a criterion to determine which individuals will proceed to the new generation has to be identified. The two main techniques that can be used in the replacement process are generational updates and steady state updates. In the case of generational updates, N children are produced from a population of size N, and so the new population completely replaces the old one. There are also derived forms of generational updates, that may consider the fittest individuals from both the children and the parents. In the case of steady state updates new individuals enter the new population as soon as they are created. This is different from the generational update, since in that case an entire new generation is produced at each generational step. In order to perform a steady state update, it is necessary to determine which individual has to be deleted. The individual to be deleted can be chosen as the worst or the oldest one, but these methods are typically drastic. For this reason, a tournament method is typically set up. A tournament between a fixed number of individuals takes place, and the individual with the highest fitness is chosen as the one who will proceed to the next generation.

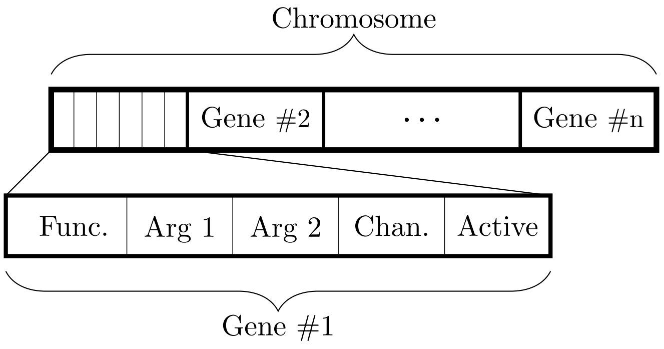


Figure 5.4: Structure of a chromosome in the genetic algorithm for automatic feature extraction [57] . © *[2011] IEEE*

Stop Criterion An important parameter that has to be defined for genetic algorithms is the stop criterion. There are several possibilities that can be employed to determine when the algorithm will stop:

- maximum generations: the evolution of the algorithm will stop when the defined maximum number of generations is reached;
- no change in fitness: if no change in the population's fitness values is observed, the algorithm will stop;
- maximum time: the algorithm will finish its search for the best solution if the defined maximum time is reached.

5.3 Genetic Algorithm for Automatic Feature Extraction

As already underlined at the beginning of the previous section, we used the genetic algorithm described by *Dal Seno et al.* in [57] to perform automatic feature extraction for P300 identification. In this approach, a genetic algorithm operates on simple feature extractors, and through the evolution process it allows to find the optimal set of feature extractors to be used in logistic regression classification. The only preprocessing that is done on the signal is bandpass filtering with cutoff frequencies of 0.5 and 30 Hz. In their work, the authors took inspiration from previous studies, where genetic algorithms were used to find the best combination between features and classifier for motor-imagery task [59] or to find suitable features for P300 detection [60].

Feature encoding

In this genetic algorithm, the structure of a chromosome is shown in Figure 5.4. A chromosome represents a possible solution to the defined problem. In this case, the problem is finding the best set of features to be extracted from the signal to perform classification. A chromosome encodes, through its genes, a set of features. A chromosome is made up of a certain number of genes, and each gene has the same identical structure with five elements.

As shown in Figure 5.4, the first three elements of a gene allow to define a function that acts as a feature extractor. The first integer in the gene (*Func.*) defines one function for feature extraction, while the following two arguments (*Arg.1* and *Arg.2*) are two parameters for that function, that lie in the range $[0, 1)$. The fourth element of a gene determines the EEG channel from which that feature has to be extracted from. The last element of the gene determines if that gene is active or inactive. Inactive genes are not considered for fitness computation. The role of inactive genes is of genetic reserve, since they can be turned on later during the evolution. The feature extractors all share the same basic structure. For each feature extractor, the input signal is cross-correlated with a weight function. The operation performed by the feature extractor is the following:

$$x = \sum_{t=1}^{T} u(t)s(t) \tag{5.9}$$

where x is the resulting feature, $u(t)$ is the weight function and $s(t)$ is the EEG signal at the input. So, the resulting feature can be seen as the correlation between the input signal $s(t)$ and the weight function $u(t)$. For our implementation of the genetic algorithm, an epoch is considered 1 second long. Since the sampling frequency of the OpenBCI board is equal to 250 Hz, an epoch is made up of 250 samples. Every epoch is extracted in the time interval around the stimulus onset. In particular, an epoch goes from 200 ms prior to the stimulation to 800ms after the stimulation.

The four weights function that we used are represented in Figure 5.5. The top left weight function computes the average value of the input signal on the interval determined by parameters A_1 and A_2, while the top right weight function computes the average of the signal by considering the values in the central portion of the interval with a greater weight. The bottom functions compute the differences in the signal of two adjacent intervals.

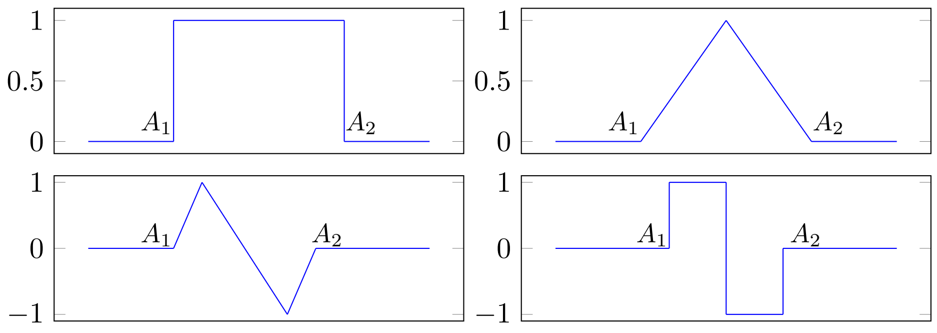


Figure 5.5: Weight functions used in the genetic algorithm.

Fitness

The fitness measure of each chromosome is chosen as the performance of a logistic classifier using the set of features encoded by the genes of the chromosome. To consider a better measure of the performance, a cross validation scheme using a number of folds equal to 4 is employed. The resulting performance on the 4 *validation* sets is used to compute the fitness measure. In order to perform the training of the logistic classifier, a training set has to be built. For our BCI system, the training was done using the speller mode. The subject, during the training phase, is provided with a series of words to spell. Before the stimulation starts, the letter or number the subject has to focus on is highlighted in red. This process, for both the computer based BCI and the Android based BCI, is shown in Figure 5.6.

The performance measure is not considered on the single epoch classification, but rather on the number of letters correctly predicted, with a bonus for those letters that can be predicted with a number of repetitions that is less than the maximum. If we call l the number of correctly predicted letters out of the total number of letters n, I the set of correctly predicted letters. R the maximum number of repetitions for the session, and $r_i, i = 1, \ldots, n$, the number of repetitions

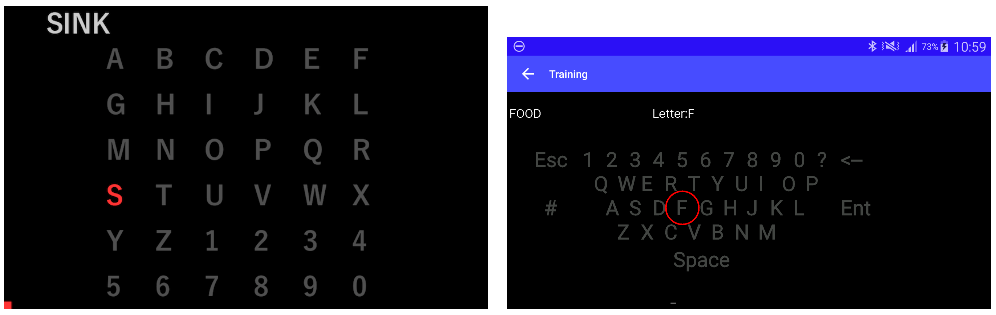


Figure 5.6: The training setup in the computer software and in the Android application.

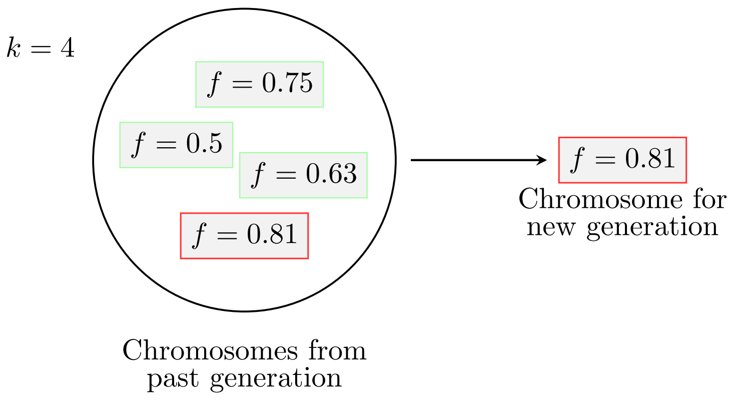

$k = 4$
$f = 0.75$
$f = 0.5$
$f = 0.63$
$f = 0.81$
$f = 0.81$
Chromosome for new generation
Chromosomes from past generation

Figure 5.7: Process of tournament selection employed in the genetic algorithm.

needed to correctly predict letter i, the fitness measure is defined as:

$$f = \frac{1}{n}\left(l + \frac{1}{l}\sum_{i\in I}\frac{R - r_i}{R+1}\right) \tag{5.10}$$

The second term of the parentheses represents the bonus term. Considering only the set of correctly predicted letters, it computes an index that is inversely proportional to r_i. This index is always strictly less than 1, so its contribution to the measure of fitness is lower than that of the one obtained with an additional correctly predicted letter. For this reason, having a higher number of correctly predicted letters is always better than having a lower number of repetitions necessary to perform a correct prediction.

In Equation (5.10), r_i is considered as the number of repetitions such that, if a letter is correctly predicted after r_i repetitions, then it has to be predicted in a correct way also for the following repetitions, i.e. for $r_i + 1, \dots, R$. repetitions. For example, if a letter is correctly predicted after only 2 repetitions, wrongly after 5, correctly when using 6 or more repetitions, then the value of r_i would be 6 and not 2.

Selection

Once the fitness of each chromosome is known, it is used to determine which chromosomes should proceed to the next generation. The employed selection mechanism is the tournament selection with elitism.

In tournament selection, represented in Figure 5.7, each new chromosome is chosen by setting up a competition between chromosomes of the old generation: in this competition, the winner is the chromosome with the highest fitness. The number of individuals considered for the tournament is typically small, in order not to favor the fittest individuals. In this implementation, we choose 4 as the number of individuals participating in the tournament. Elitism is another common practice in genetic algorithms. When employed, the fittest chromosome or chromosomes are kept in the new generation even if selection discarded them, or if crossover or mutation modified them. In our case, the number of chromosomes considered for elitism is equal to 1.

Crossover

Once selection is performed, the selected individuals undergo the crossover operation. This operator allows to create new individuals starting from a pair of parents chromosomes. For each chromosome, crossover is applied with a probability of 0.7. Two chromosomes are divided in a random manner at a gene boundary, and then the resulting four sections are recombined together, as shown in Figure 5.2. As shown in the figure, one section of a chromosome can be coupled to one of the two section of the other chromosome, and the way to do this is decided randomly each time. Since the crossover operation can be applied to chromosomes which share a common ancestor, in the children chromosomes duplicate genes may be present. These duplicate genes are not considered for fitness measure.

Mutation

After selection and crossover, the mutation operator, shown in Figure 5.3, is applied on the chromosomes. As already explained, mutation operator works on gene elements, rather than on chromosomes. For each element of each gene, mutation is applied with a probability of 0.005. For discrete elements in the gene (function for feature extraction, channel and boolean flag), another admissible value is chosen. For continuous elements (the parameters for the feature extraction function), a perturbation is added with a Gaussian distribution. Since it may happen that the new value is outside of the admissible range of $[0, 1)$, the new value is wrapped around.

Population Size

The number of chromosomes constituting the population is constant during the evolution of the genetic algorithm. For our implementation of the genetic algorithm, the population is composed of 100 chromosomes. The initial population is randomly initialized. In particular, a geometric distribution ($\mu = 20$) is used to determine the number of features (i.e, genes) for each chromosome of the population. The values of each gene element are chosen from uniform distribution over the whole range of admissible values for that element.

Stop Criterion

The criterion used to stop the evolution of the algorithm is the maximum number of generations. We used a number of generations equal to 15, decreasing it to 12 for the runs that were very high time-consuming. A check on the growth of the fitness value is done after each run, so that the evolution can be stopped if no growth of the fitness is present.

Feature Interpretation

Due to the fitness measure, which depends on the performance of the logistic classifier, feature extraction and training of the logistic classifier are tightly linked. Therefore, a deeper interpretation of the selected features is possible.

A single feature x_j is extracted as:

$$x_j = \sum_{t=1}^{T} u_j(t)s(t) \tag{5.11}$$

where j stands for the feature encoded in the $j-th$ gene of the chromosome. In this equation, T is the number of samples of the epoch, u_j is the specific weight function encoded by the parameters of the $j-th$ gene, $s(\cdot)$ is the EEG epoch extracted from a single channel.

A logistic classifier estimates the probability of $s(\cdot)$ belonging to the target class with the following formula:

$$P(y=+1|\mathbf{x}) = \frac{1}{1+e^{w_0+\sum_{j=1}^{n} w_j x_j}} \tag{5.12}$$

The exponent in the denominator of Equation (5.12) can be rearranged as:

$$\begin{aligned} w_0 + \sum_{j=1}^{n} w_j x_j &= w_0 + \sum_{j=1}^{n} w_j \left(\sum_{t=1}^{T} u_j(t)s(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) \right) s(t) \end{aligned} \tag{5.13}$$

Vector $v(t)$ is defined as:

$$v(t) = \sum_{j=1}^{n} w_j u_j(t) \tag{5.14}$$

As shown in Equation (5.14), the vector $v(t)$ depends only on the features set and on the classifier. Since $\mathbf{u}(\cdot)$ and $\mathbf{w}$ don't depend on the value of the signal itself, vector $v(t)$ is the same for all the considered epochs. Equation (5.12) can be rewritten as:

$$\begin{aligned} P(y=+1|s(\cdot)) &= \frac{1}{1+e^{w_0+\sum_{t=1}^{T} v(t)s(t)}} \\ &= \frac{1}{1+e^{w_0+\langle \mathbf{v},\mathbf{s} \rangle}} \end{aligned} \tag{5.15}$$

If all channels are considered, the precedent equations can be readjusted as follows:

$$x_j = \sum_{t=1}^{T} u_j(t) s_{c_{(j)}}(t) \tag{5.16}$$

where C is the number of channels, $s_c(\cdot)$ is the EEG signal recorded on channel c, $c_{(j)}$ it the channel from which the $j-th$ epoch has to be extracted from. Equation (5.13) becomes:

$$\begin{aligned} w_0 + \sum_{j=1}^{n} w_j x_j &= w_0 + \sum_{j=1}^{n} w_j \left(\sum_{t=1}^{T} u_j(t) s_{c_{(j)}}(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) s_{c_{(j)}}(t) \right) \end{aligned} \tag{5.17}$$

Then, we group together features related to the same channels, and Equation (5.17) becomes:

$$\begin{aligned} & w_0 + \sum_{t=1}^{T} \left(\sum_{j=1}^{n} w_j u_j(t) s_{c(j)}(t) \right) \\ &= w_0 + \sum_{t=1}^{T} \left(\sum_{c=1}^{C} \sum_{j:c(j)=c} w_j u_j(t) s_c(t) \right) \\ &= w_0 + \sum_{c=1}^{C} \sum_{t=1}^{T} \left(\sum_{j:c(j)=c} w_j u_j(t) \right) s_c(t) \end{aligned} \tag{5.18}$$

The term

$$v_c(t) = \sum_{j:c(j)=c} w_j u_j(t) \tag{5.19}$$

is the same for all the epochs, and depends only on the considered channel.

Considering all the above equations, the final formula for computing the probability of the signal $s(\cdot)$ pertaining to class $y = 1$ is:

$$P(y = 1|\mathbf{s_1}, \dots, \mathbf{s_c}) = \frac{1}{1 + e^{w_0 + \sum_{c=1}^{C} \langle \mathbf{v}_c, \mathbf{s}_c \rangle}} \tag{5.20}$$

In conclusion, Equation (5.20) allows to compute the probability for the target class by using directly the signals $\mathbf{s_1}, \dots, \mathbf{s_c}$. The dot product results in a correlation between vector $\mathbf{v}_c$ and the signal. Therefore this classification is based on the similarity between the epoch signals and channel templates, expressed by the vectors $\mathbf{v}_c$.

5.4 Online Classification

Now that we have explained the topics of classification with logistic regression and of automatic feature extraction performed with a genetic algorithm, in this section we cover the topic of online classification. Online classification works in the same way for both the computer based BCI and the Android based BCI. For the computer based BCI, there is no difference in online classification for three possible BCI modes of use (speller mode, cartesian control of the robotic arm, high level control of the robotic arm), with the only difference that in the high level control of the robotic arm single cells flash, rather than columns and rows.

Both in the computer based BCI and in the Android based BCI, the number of repetitions and some parameters related to the stimulation (intensification duration, inter stimulus interval,. . .) can be set prior to the beginning of visual stimulation.

During the visual stimulation, the EEG signal is segmented into epochs using the photoresistor signal. In particular, the sample at which the stimulus is provided is found with a moving window method. Basically, the time sample T at which the stimulus is provided is determined as the one for which the maximum value of the window $[T, T + 10]$ is higher than the threshold Th, and the minimum value of the window $[T - 11, T - 1]$ is lower than Th. The threshold value of the photoresistor signal pr is found as:

$$Th = \frac{\text{MAX}(pr) + \text{MIN}(pr)}{2} \tag{5.21}$$

After the epochs are extracted from the signals, they are filtered with a sixth-order Butterworth bandpass filter with cut off frequencies of 0.5 and 30 Hz.

At the conclusion of the visual stimulation for each trial, the classification process begins.We implemented two different classification modalities, to compare the results obtained with them, and we will refer to them as the *probability averaging* method and the *epochs averaging* method.

5.4.1 Probability Averaging Method

In the *probability averaging* method, data are averaged in the classification score space. For each extracted epoch, the probability of the presence of a P300 is computed with the Equation (5.20). Since information about which stimuli are presented is saved during the stimulation process, it is possible to determine which row and column elicited a P300 response with the highest probability.

If we consider, as an example, the $i - th$ row of the matrix, for each repetition of the stimulation, row i flashes one time, and the probability of the presence of the P300 in the epoch extracted around the time at which row i flashed is calculated. At the end of the stimulation, the average probability is computed as:

$$\overline{P}_i = \frac{1}{R} \sum_{j=1}^{R} p_{i,j} \tag{5.22}$$

where $p_{i,j}$ is the probability of the P300 response being present in the epoch associated to row i at the $j-th$ repetition, and R is the maximum number of repetitions. This process is done for all the rows and for all the columns.

So, at the conclusion of the probability computation, we will have the following vectors:

$$\overline{P}_{rows} = \left[\begin{array}{ccccc} \overline{P}_1 & \overline{P}_2 & \overline{P}_3 & \dots & \overline{P}_{rows_{MAX}} \end{array} \right]$$
$$\overline{P}_{cols} = \left[\begin{array}{ccccc} \overline{P}_1 & \overline{P}_2 & \overline{P}_3 & \dots & \overline{P}_{cols_{MAX}} \end{array} \right]$$

where $rows_{MAX}$ is the maximum number of rows in the grid, and $cols_{MAX}$ is the maximum number of columns in the grid. The $i-th$ element of the array $\overline{P}_{rows}$ contains the average probability of the $i-th$ row, while the $i-th$ element of the array $\overline{P}_{cols}$ contains the average probability of the $i-th$ column. These two vectors are then used to determine the symbol on which the subject was focusing on.

The target symbol is found by determining the intersection between the row and the column for which the average value of probability is maximum:

$$\text{row}_{P300} = \max_i \overline{P}_{rows,i}$$

$$\text{column}_{P300} = \max_i \overline{P}_{cols,i}$$

This classification process is the same for speller and cartesian robotic arm control. The only difference between these two modalities is the dimension of the stimulation matrix. For the high level control of the robotic arm, the cell with the highest probability is directly determined.

5.4.2 Epochs Averaging Method

In the *epochs averaging* method for the classification, EEG epochs are averaged in the data space. The basic difference in respect to the *probability averaging* method is that, instead of computing the average probability, we compute the average epoch associated to the flashing of each row and column. Then, we compute the probability of this average signal.

If we consider, as an example, the $i-th$ row of the matrix, for each repetition of the stimulation, row i flashes one time. For each repetition, the epoch around the time at which row i flashed is extracted. At the end of the stimulation, the average epoch is computed as:

$$\overline{S}_i = \frac{1}{R} \sum_{j=1}^{R} s_{i,j} \tag{5.23}$$

where $s_{i,j}$ is the epoch associated to the flashing of row i at the $j-th$ repetition, and R is the maximum number of repetitions. After computing the averages for all rows and columns, the following vectors are defined:

$$\overline{S}_{rows} = \left[\begin{array}{cccc} \overline{S}_1 & \overline{S}_2 & \ldots & \overline{S}_{rows_{MAX}} \end{array} \right]$$
$$\overline{S}_{cols} = \left[\begin{array}{cccc} \overline{S}_1 & \overline{S}_2 & \ldots & \overline{S}_{cols_{MAX}} \end{array} \right]$$

where $rows_{MAX}$ is the maximum number of rows in the grid, and $cols_{MAX}$ is the maximum number of columns in the grid. The $i-th$ element of the array $\overline{S}_{rows}$ contains the average epoch of the $i-th$ row, while the $i-th$ element of the array $\overline{S}_{cols}$ contains the average epoch of the $i-th$ column.

Then, the probability associated to each of these average signals is computed, and the following vectors are obtained:

$$P_{rows} = \left[\begin{array}{ccccc} P_1 & P_2 & P_3 & \ldots & P_{rows_{MAX}} \end{array} \right]$$
$$P_{cols} = \left[\begin{array}{ccccc} P_1 & P_2 & P_3 & \ldots & P_{cols_{MAX}} \end{array} \right]$$

Once the probability values are known for each average epoch, the target symbol is determined by computing the row and column for which probability is the highest. Similar to the *probability averaging* method, we have that the predicted row and column are:

$$\text{row}_{P300} = \max_i P_{rows,i}$$
$$\text{column}_{P300} = \max_i P_{cols,i}$$

In the high level control of the robotic arm an individual cell, instead of a column and row, is identified as the one which generated the P300 response.

5.4.3 Summary

In Figure 5.8, we show a graphical summary of the classification component of the BCI system. In this block diagram, all the different phases of the classification process are shown. On the left side of the image, the steps required to perform the training of the classifier are represented. While on the right side of the image, the steps required to perform online classification are summarized.

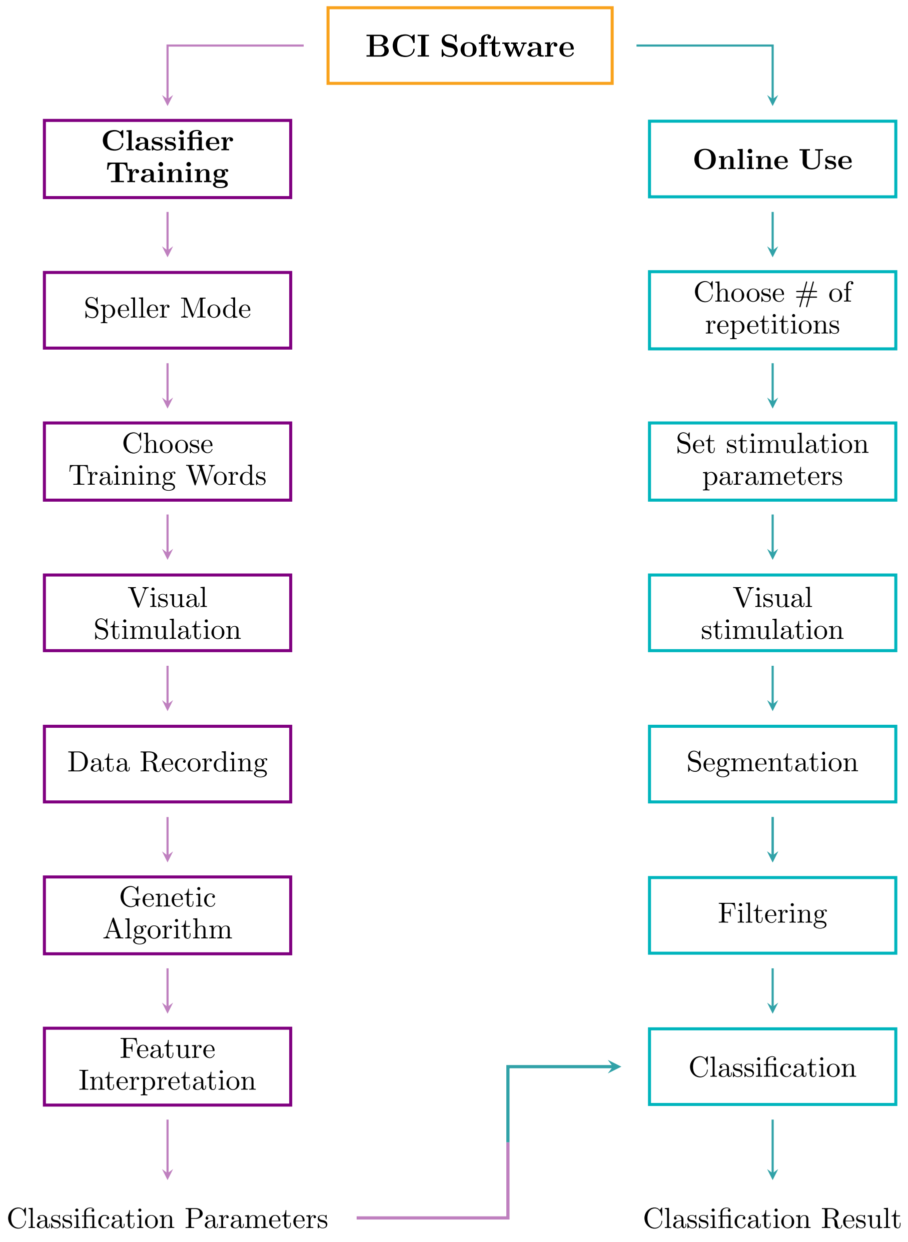


Figure 5.8: Block diagram of the classification process. On the left side of the image, the blocks related to the training of the classifier are shown. On the right side of the image, the steps required to perform online classification are represented.

Chapter 6

Experimental Results

In this chapter, we will discuss the results obtained with two subjects participating in evaluating the device. Firstly, the experimental design will be described. Then, data obtained with the computer based BCI will be presented. In particular, data related to the genetic algorithm and to the classifier validation will be provided, and online performance of the device will be evaluated. Lastly, data obtained with the Android based BCI will be presented.

6.1 Experimental Design

Two healthy subjects (Table 6.1) participated in evaluating the device. Each subject was required to participate in three EEG recording sessions. In session one, P300 elicitation was valuated and data to train the classifier were recorded. In the second session experimental data for classifier validation were collected. In the last session, tests on online use of the BCI system were carried out. Subject S1 participated in tests with both the computer based BCI and the Android based BCI. Subject S2 participated in tests with only the computer based BCI.

Table 6.1: Subjects participating in the experimental study.

Subject ID	**Age**	**BCI Experience**	**BCI Tested**	
			Computer	Android
S1	23	No	X	X
S2	20	No	X	

6.2 Computer Based BCI

In this section, data obtained with the computer based BCI will be presented. Both subject S1 and subject S2 participated in the tests carried out with the computer based BCI. The locations on the scalp used for EEG recording were C3, C4, Pz, P3, P4, P7, P8 and Oz, according to the 10-20 international standard for EEG recording [13].

6.2.1 Classifier Training Phase

The first session subject S1 and subject S2 participated in was a training session. As we already showed in Figure 5.6, during the training session the subjects were asked to spell a series of words. Before each trial, the letter the subject had to focus on was highlighted in red. During each trial, flashing of the row or the column that contained the letter of interest elicited a P300 response. The spelling process lasted until the sequence of chosen letters was completed. Each epoch was then be labeled as target, if extracted around a target stimuli, or non-target.

The training set of target and non-target epochs was then used to perform automatic feature extraction and training of the logistic classifier for each subject. Both of these tasks were accomplished using the genetic algorithm described in Section 5.3. Information about the training set used for each subject is shown in Table 6.2. Subject S2 preferred 5 repetitions instead of 10, finding it easier to focus on shorter length sessions.

Fitness values of the population of 100 chromosomes across the generations of the genetic algorithm, computed using Equation (5.10), are shown in Figure 6.1a. The value of fitness for each chromosome is reported, together with the mean value of fitness for each generation. Fitness values, as expected, increase as the evolution proceeds. As selection, crossover, and mutation modify the population of chromosomes, the set of features encoded by the chromosomes improve. Therefore, the average performance of the logistic classifier on the 4 folds used for cross validation increases.

Table 6.2: Characteristics of the training set of subject S1 and S2 obtained with the computer based BCI.

Subject	**Letters**	**Epochs**	**Hours**	**Repetitions**
S1	555	66600	5	10
S2	127	7620	1	5

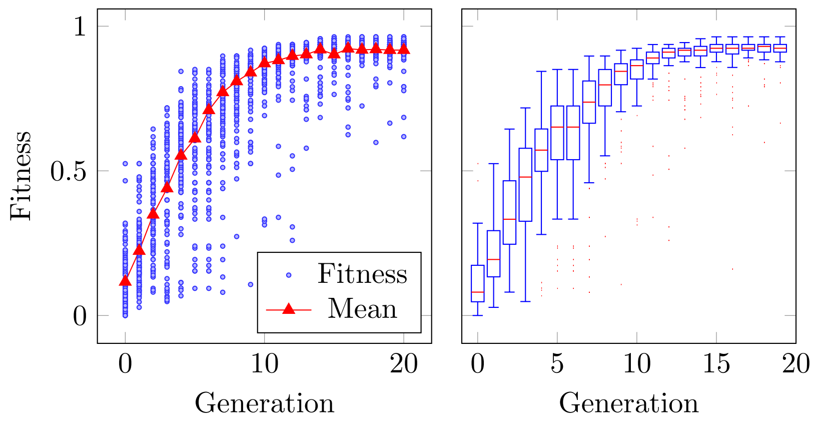


Figure 6.1: *Left:* Fitness value of each of the 100 chromosomes at each generation is shown in blue. Mean fitness value is shown in red. *Right:* Box plots of fitness values as a function of genetic algorithm generation. The data shown in the figure were obtained with the training set of subject S2.

In Figure 6.1b, the same information about fitness evolution is represented trough box plots. As expected, the mean fitness values converges to high values, close to 1, and the variation in fitness across chromosomes reduces. A plateau in the fitness values is typically reached after approximately 12 generations. Therefore, we limited the maximum number of generations to 15. For the biggest training sets, we limited it to 12.

In Figure 6.2 we evaluate the fitness values and the number of features encoded by only active and unique genes across generations. Greater number of features encoded by the chromosome results in a higher fitness value. The plot reaches a plateau at ~ 200 features, close to generation 12. As shown in the plot, even if chromosomes increase in size, and encode more features for classification, there is minimal to no improvement in the fitness value after the 12^{th} generation.

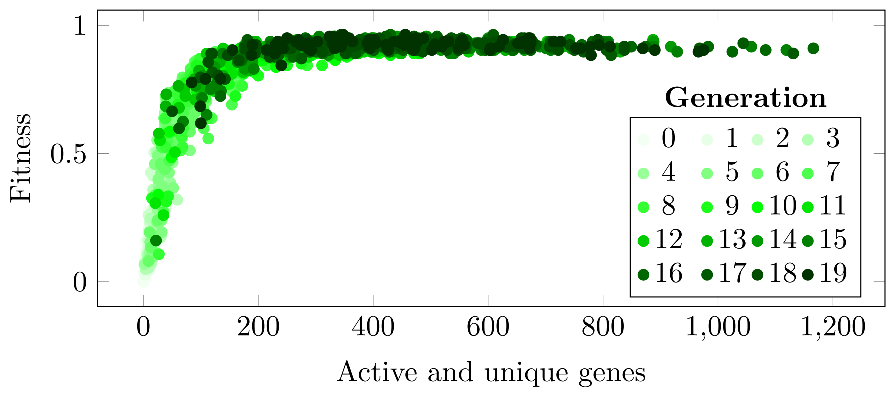


Figure 6.2: Fitness values as a function of number of features and of genetic algorithm generation. The data shown in figure were obtained with the training set of subject S2.

Table 6.3: Characteristics of the validation set of the subject S1 and S2, obtained with the computer based BCI.

Subject ID	Sessions	Letters	Epochs	Maximum Rep.
S1	6	62	7440	10
S2	4	57	3420	5

6.2.2 Classifier Validation Phase

After performing automatic feature extraction and training the logistic regression classifier, the obtained classifier was validated on a validation data set never used before by the genetic algorithm. For both subjects, the validation set was composed of data recorded on various days. The characteristics of the validation sets are shown in Table 6.3. It is important to evaluate the classifier using data recorded on various sessions, to assure performance independence from external variations that could be present in recording sessions.

Later in the chapter, we will refer to a measure of accuracy. Accuracy is computed in terms of correctly predicted symbols. If l represents the number of correctly predicted symbols in a session, and n the total number of symbols spelled, accuracy is computed as:

$$ACC = \frac{l}{n} \tag{6.1}$$

Four analysis were performed on the validation set. First analysis, standard measures of classifier performance. Second analysis, accuracy as a function of the number of repetitions. Third analysis, accuracy as a function of the size of the training set. Fourth analysis, inter - subjects dependence.

Classifier Performance

In order to assess the classifier performance, we computed standard statistical measures of binary classification. The measures that we took into account were:

- recall and specificity, defined as the probability that a target or non-target epoch is correctly detected as such;
- precision, defined as the probability that a detected target epoch is a true target epoch;
- area under the ROC (Receiving Operating Characteristic) curve.

Table 6.4: Classifier performance statistical measures computed on the validation sets of subject S1 and S1, obtained with the computer based BCI.

Subject ID	TP	TN	FP	FN	Recall	Precision
S1	853	4506	1694	387	0.69	0.33
S2	415	2216	634	155	0.73	0.40

The results of this analysis, together with measures of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), are shown in Table 6.4. The ROC curve, and the related measure of area under the ROC curve, is shown in Figure 6.3.

Accuracy vs Repetitions

In this section we evaluate accuracy as a function of the number of repetitions. This evaluation is done with both classification methods: *probability averaging* and *epochs averaging* method.

In Figure 6.4, accuracy measured with *probability averaging* and *epochs averaging* methods as a function of the number of repetitions is represented.

As expected, for both classification methods accuracy increases when the number of repetitions is increased. Regarding the two different classification methods, *probability averaging* and *epochs averaging*, they both perform well. Throughout our experiments,we noticed that for a noisy signal, the *epochs averaging* method perform better than the *probability averaging* method. Since, by averaging the epochs, the amount of noise present in the signal is reduced. This is not true for the probability averaging method, since the averaging is performed on the classification

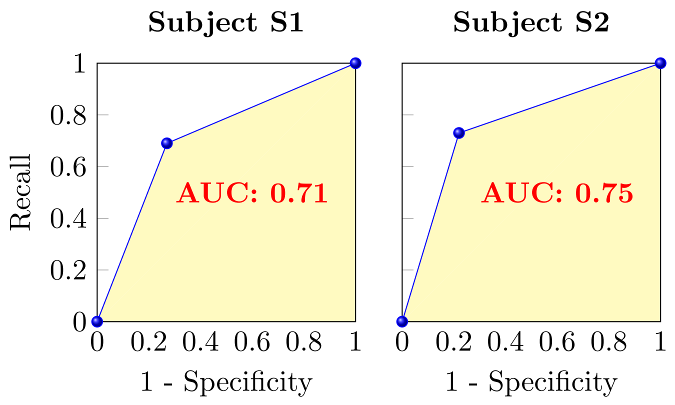


Figure 6.3: The ROC curve of the classifier computed on the validation sets of subject S1 and S2, obtained with the computer based BCI. The value of the area under the ROC curve is also shown.

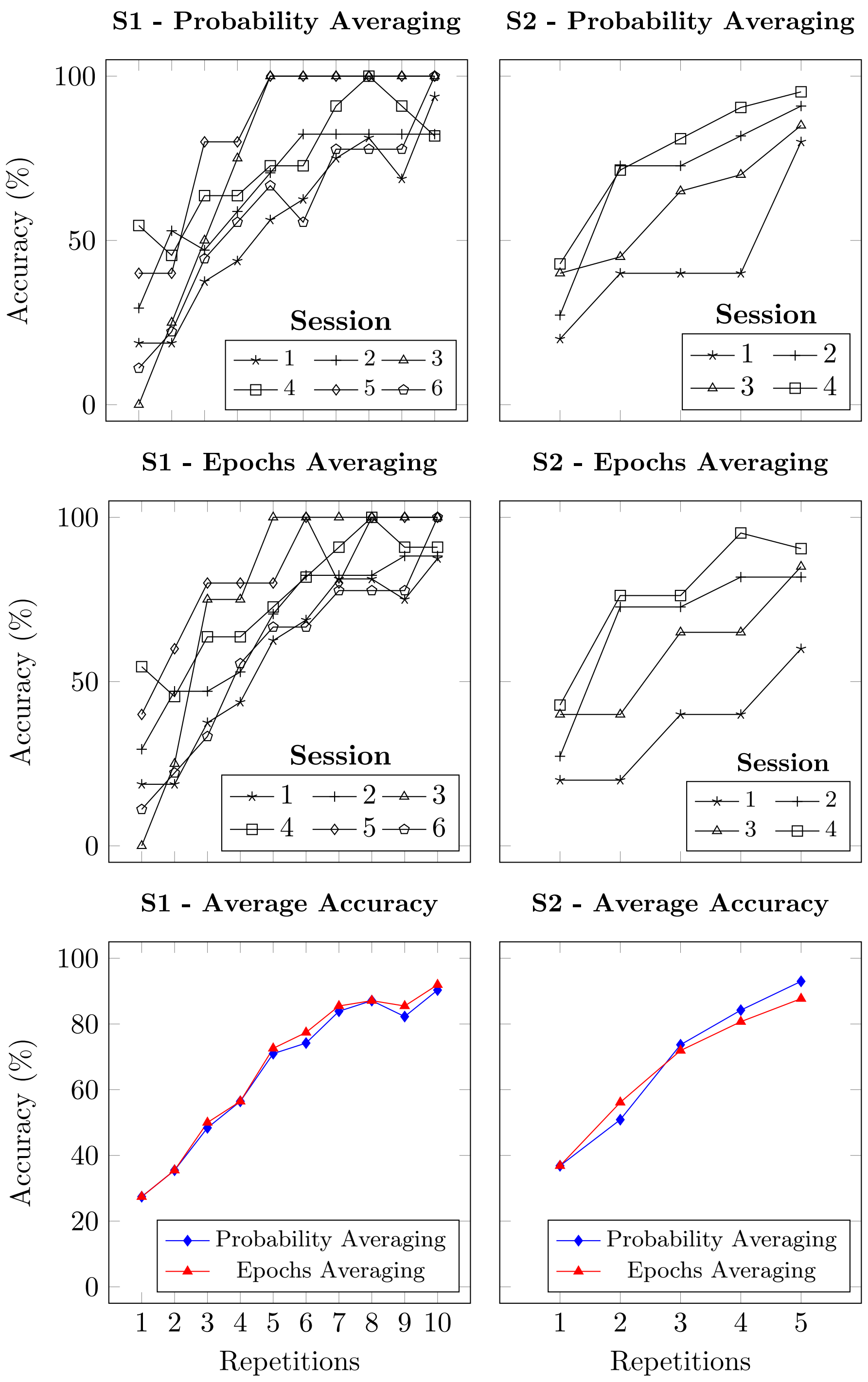


Figure 6.4: Accuracy as a function of the number of repetitions for the validation set.

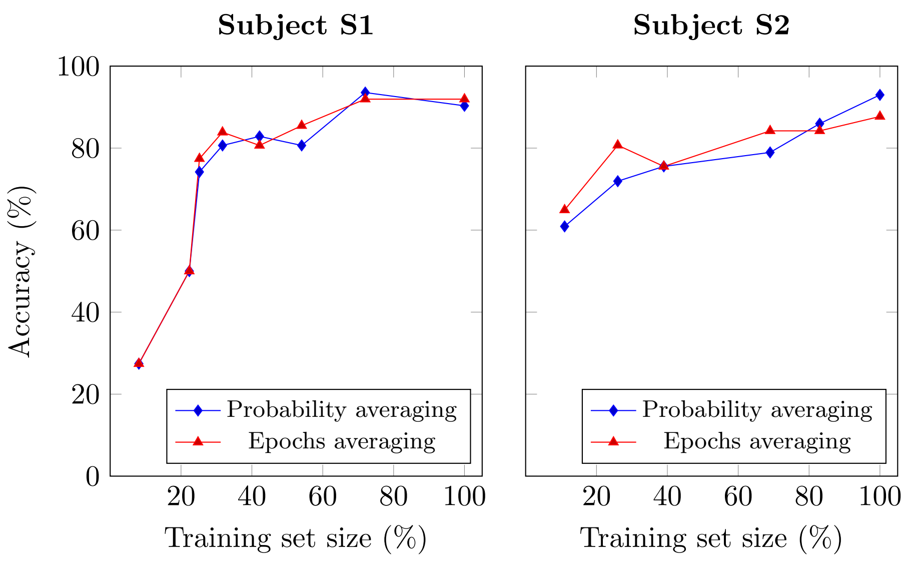


Figure 6.5: Average accuracy on the validation set as a function of the training set size. The maximum number of repetitions was used to compute the accuracy.

score space and not on the data space.

For subject S1, accuracy greater than 80% is reached when using 7 repetitions (83.87% with probability averaging method, 85.48% with epochs averaging method). For subject S2, accuracy greater than 80% is reached when using 4 repetitions (84.21% with probability averaging method, 80.7% with epochs averaging method).

Accuracy vs Training Set Size

Another analysis we performed was related to how accuracy changes as a function of the size of the training set. The training set, as already explained, is used to perform automatic feature extraction and training of the logistic regression classifier. Therefore, an increase in the accuracy is expected when increasing the size of the training set. To perform this analysis, the accuracy on the same test set was computed by gradually increasing the size of the training set. In Figure 6.5, the result of this analysis is shown. The training set size is shown as a percentage of the total training set size for each subject.

For subject S1, accuracy reaches above 80% around 32% of the total training set size. This corresponds to a training set composed of 180 letters, equal to 21600 epochs when using 10 repetitions per trial. For subject S2, accuracy reaches above 80% around 70% of the total training set size. This corresponds to a training set composed of 89 letters, equal to 5340 epochs when using 5 repetitions per trial.

Table 6.5: Inter-subject dependence. The accuracy was computed using the maximum number of repetitions of the validation set.

(a) Probability Averaging

	P300 Templates	
	S1	S2
S1	90.32	1.61
S2	12.28	92.98

(b) Epochs Averaging

	P300 Templates	
	S1	S2
S1	91.35	1.61
S2	10.35	87.71

Inter-Subjects Dependence

The last analysis that we performed on the validation set is related to inter-subjects dependence. In Figure 6.6 the average target and non-target epochs of the validation set for subject S1 and S2 are shown. Visually comparing the two signals, a clear difference is visible among the two subjects. Therefore, emphasizing the fact that subject specific features have to be extracted with the genetic algorithm for each subject. To underline again the importance of subject specific training, we performed classification on the validation set of one subject using P300 templates extracted from the training set of the other subject. Results are shown in Table 6.5. In this table, the value of each cell represents the accuracy on the validation set of one subject (row) when tested with the templates of the other subject (column).

As expected, classification performance is low when not using the subject specific templates, emphasizing that the fact that even though P300 responses are innate response of the brain, they are different among individuals. Therefore, each subject has to participate in a training session.

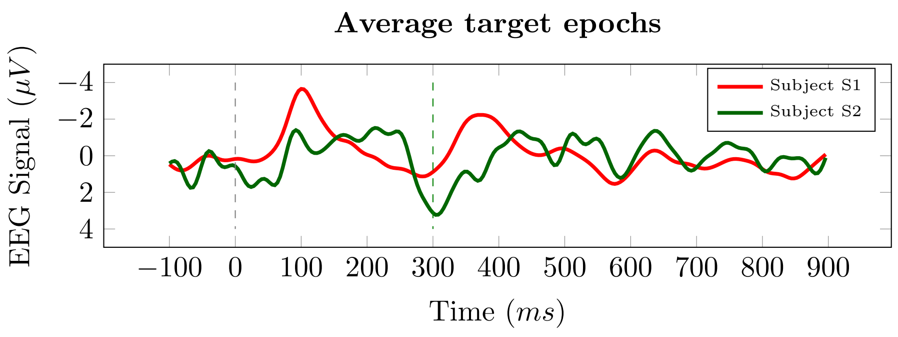


Figure 6.6: Average target epochs of the validation set for subject S1 and S2.

Table 6.6: Number of choices, N, and of flashes, F, for the three possible BCI modes of use.

Mode of use	N	F
Speller	36	12
Cartesian robotic arm control	16	8
High level robotic arm control	5	5

6.2.3 Online Classification

After performing classifier training and validation for each subject, we tested the online classification accuracy for the three modes of use: speller, cartesian control of the robotic arm, and high level control of the robotic arm. All the experimental tests carried out to assess the performance of the online classification were performed on different days than the ones in which the training set and the validation set were recorded. This was done in order to assure the validity of the results.

Other than accuracy, that is computed using Equation (6.1), a common measure of performance in the BCI field is the bit rate. The bit rate can be expressed in terms of $bits/min$ [2], and is commonly called Information Transfer Rate (ITR). ITR contains information about communication speed and accuracy.

The formula used to compute bit rate B is:

$$B = \log_2 N + P \log_2 P + (1 - P) \log_2 \left(\frac{1-P}{N-1} \right) \tag{6.2}$$

where N represents the possible choices per trial (values of N for the different modes are shown in Table 6.6), and P represents the probability that the choice on which the subject is focusing on is selected. Therefore, P represents the accuracy of the system. The assumed hypothesis is that each choice has the same probability of being selected. ITR can be computed as follows:

$$ITR = B \cdot \frac{N}{min} = B \cdot \frac{60}{T_{tot}} = \left[\log_2 N + P \log_2 P + (1-P) \log_2 \left(\frac{1-P}{N-1} \right) \right] \cdot \frac{60}{T_{tot}} \tag{6.3}$$

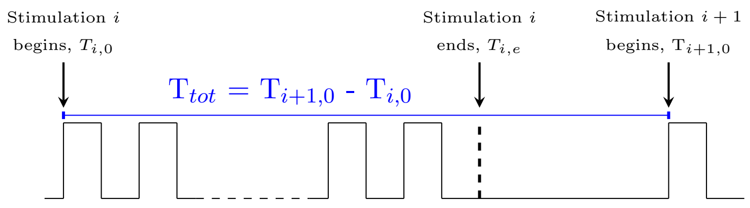

Stimulation i begins, $T_{i,0}$
Stimulation i ends, $T_{i,e}$
Stimulation $i+1$ begins, $T_{i+1,0}$
$T_{tot} = T_{i+1,0} - T_{i,0}$

Figure 6.7: Graphical description of the protocol used to compute total stimulation time, T_{tot}.

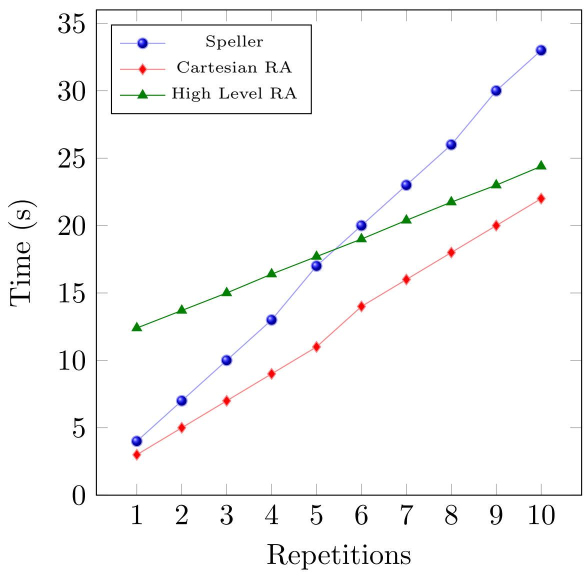


Figure 6.8: Time to complete a trial, T$_{tot}$, as a function of the number of repetitions in the computer based BCI.

To compute the ITR, it is necessary to determine the total number of trials per minute, N/min. Therefore, we computed the time required by the software to complete a trial, T_{tot}. A graphical representation on how the computation of this value of time is performed is shown in Figure 6.7. The measurement of time starts from the beginning of the stimulation, and ends when the following stimulation starts. The results of the time computation are shown in Figure 6.8.

Speller

In order to compute the online classification accuracy in the speller mode, the subjects informed us on the letters they were focusing on during the online test session.

Results of the online classification for both subjects are shown in Figure 6.9. For subject S1 online performance is comparable to the validation set results. For subject S2, up until five repetitions, online performance is lower than the validation test results.

After computing the online classification accuracy, we computed the performance of the speller mode in terms of *bits/min*. The values of *bits/min* as a function of the number of repetitions are shown in Figure 6.10. Bit rate takes into consideration both speed and accuracy of the system. Therefore, in order to increase communication speed, a lower number of repetitions is favorable. Versus, high number of repetitions results in higher classification accuracy. Therefore, the optimal number of repetitions should be chosen in order to maximize speed and accuracy.

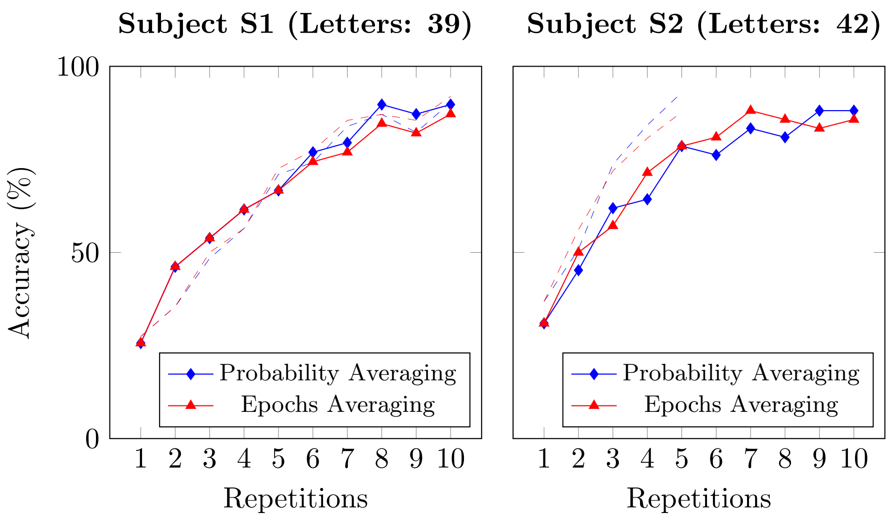


Figure 6.9: Online classification accuracy as a function of the number of repetitions for the speller mode. Accuracy on the validation set is shown with dashed lines.

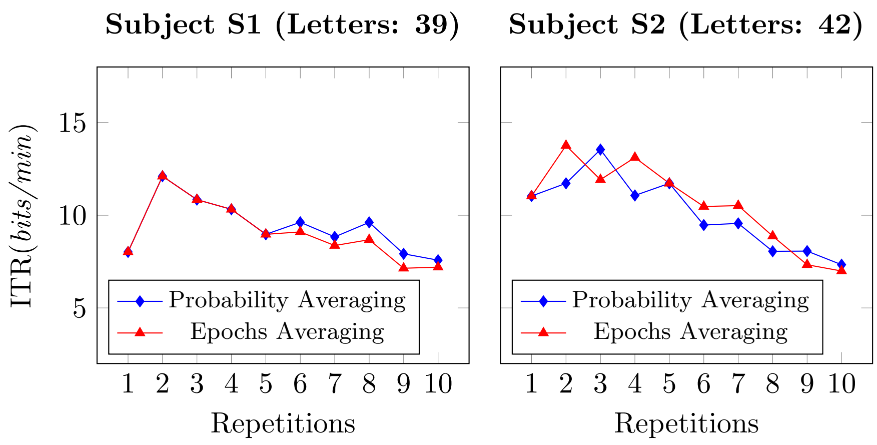


Figure 6.10: Information transfer rate as a function of the number of repetitions for the speller mode.

For subject S1, an accuracy of 79.49% is reached after 7 repetitions using probability averaging method. When using epochs averaging method, 8 repetitions are necessary to achieve an accuracy of 84.62%. The correspondent ITR are 8.83 and 8.68 *bits/min*. The maximum value of ITR for subject S1 is 12.11 *bits/min*, reached using 2 repetitions for both probability averaging and epochs averaging.

For subject S2, when using 7 repetitions and the probability averaging method, an accuracy of 83.33% is reached. With epochs averaging method, 6 repetitions are necessary to reach an accuracy of 80.95%. The correspondent ITR are 9.56 and 10.47 *bits/min*. Maximum values of ITR for subject S2 are 13.54 *bits/min* (3 repetitions, probability averaging) and 13.76 *bits/min* (2 repetitions, epochs averaging).

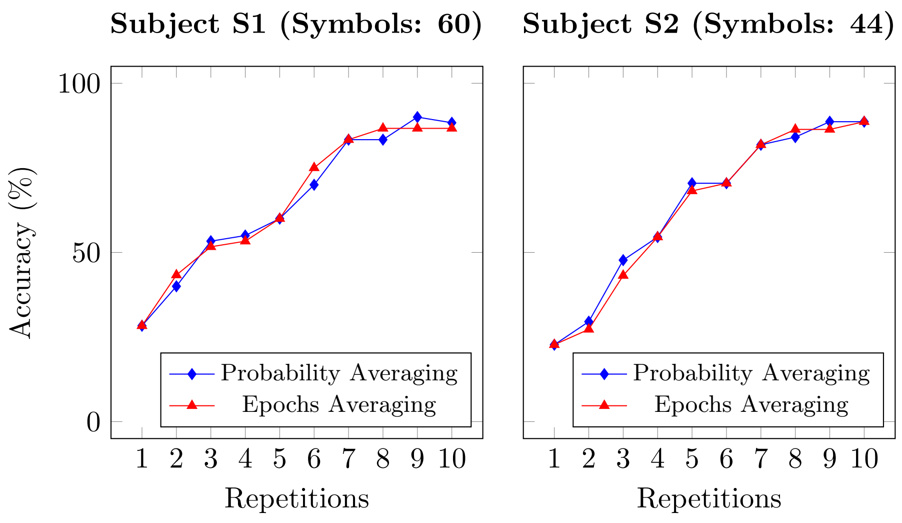


Figure 6.11: Online classification accuracy as a function of the number of repetitions for the cartesian control of the robotic arm.

Robotic Arm Cartesian Control

Online classification accuracy for the robotic arm in cartesian control mode was tested. Similar to the speller mode, the subject had to focus on symbols of the stimulation rather than on letters or numbers. The subject would then communicate the target symbols he focused on during the session. As a feedback to the user, the detected symbol is shown in green on the top of the screen. The additional feedback provided to the user is represented by the movement of the robotic arm. Results for the cartesian control of the robotic arm in terms of accuracy and ITR are shown in Figure 6.11 and Figure 6.12, respectively.

For subject S1, 7 repetitions were necessary to obtain an accuracy above 80% (precisely, 83.3%) with both *probability averaging* and *epochs averaging* method. The resulting ITR is 10.12 *bits/min*, that is also the highest value of ITR reached by subject S1. The highest value of accuracy, 90%, was reached using 9 repetitions and the probability averaging method.

For subject S2, 7 repetitions were necessary to achieve an accuracy of 81.82%, with both *probability averaging* and *epochs averaging* method. The correspondent ITR is 9.77 *bits/min*. The highest value of ITR was reached using 5 repetitions: 10.75 *bits/min*. This ITR corresponds to an accuracy of 75%. The highest value of accuracy, 88.64%, was reached with *probability averaging* method after 9 repetitions, and at 10 repetitions by the *epochs averaging* method.

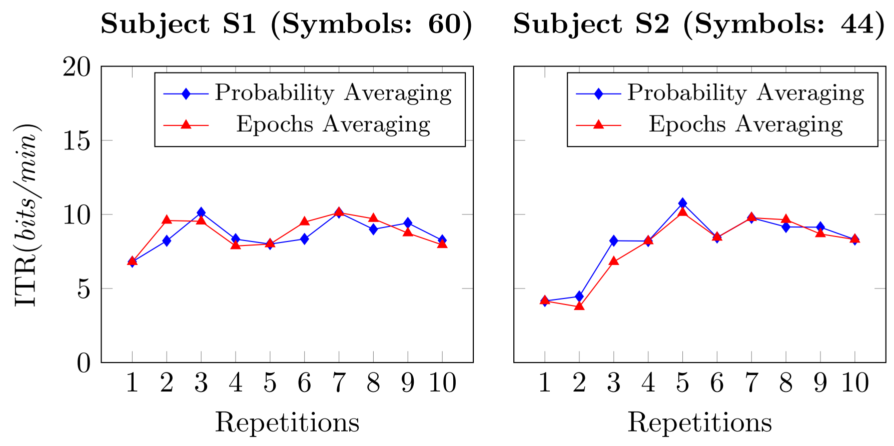


Figure 6.12: Information transfer rate as a function of the number of repetitions for the cartesian control of the robotic arm.

Robotic Arm High Level Control

The final test was performed with the high level control of the robotic arm. To compute online classification accuracy, the subjects would communicate the target symbols they focused on during the trial. We remind in this section that, for the high level control of the robotic arm, in a single repetition one cell flashes one time. For the speller and the cartesian control of the robotic arm, that employ row/column flashing, in a single repetition each cell flashes two times (one when the correspondent row flashes, and one when the correspondent column flashes).

Results for the high level control of the robotic arm in terms of accuracy and ITR are shown in Figure 6.13 and Figure 6.14, respectively.

For subject S1, 9 repetitions are necessary to reach an accuracy of 83.08% when using probability averaging or epochs averaging method. The resulting ITR is 3.46 *bits/min*, which represents also the maximum value of ITR achieved by subject S1. The maximum value of accuracy achieved by the subject is 84.62%, reached at 10 repetitions and employing the *epochs averaging* method.

For subject S2, lower accuracy, in comparison to speller and cartesian control of the robotic arm, was obtained. 6 repetitions were necessary to achieve an accuracy of 72%, both with *probability averaging* method and *epochs averaging* method. The correspondent ITR is 2.85 *bits/min.* Maximum value of accuracy, 76%, was achieved with 10 repetitions and *probability averaging method.* Maximum ITR, 3.11 *bits/min*, was reached with 3 repetitions and *epochs averaging method.*

Values of ITR in the high level control of the robotic arm may seem low in comparison to the

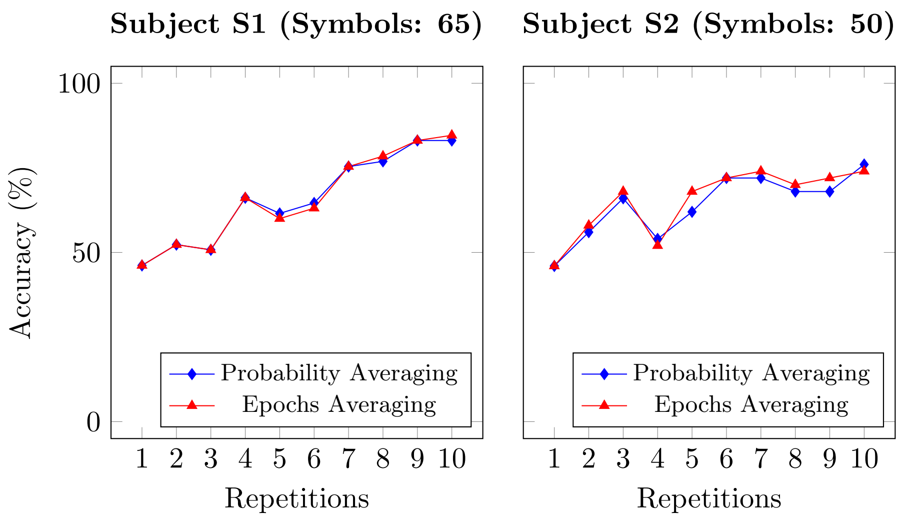


Figure 6.13: Online classification accuracy as a function of the number of repetitions for the high level control of the robotic arm.

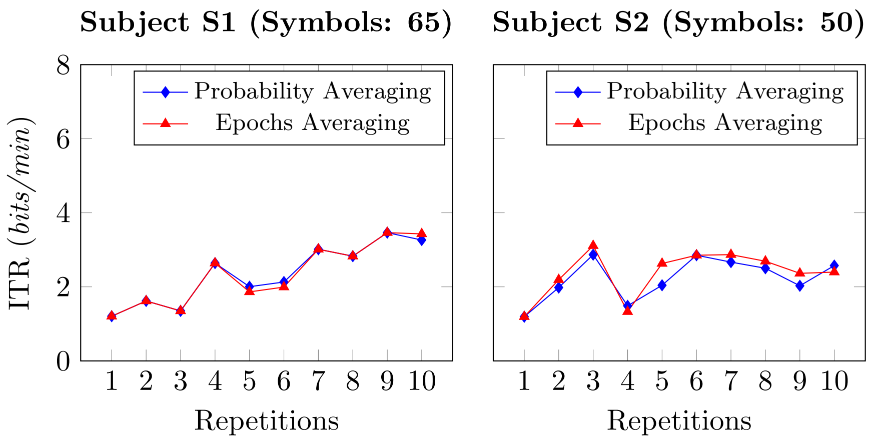


Figure 6.14: High level control of the robotic arm: information transfer rate as a function of the number of repetitions.

ones achieved with both speller and cartesian control of the robotic arm. What has to be taken into consideration is that, during high level control of the robotic arm, no stimulation happens when the robotic arm is moving, since the high level action is not immediate. Therefore, as shown in Figure 6.8, time required to complete a trial for the high level control of the robotic arm is higher. As a consequence, the ITR is lower.

Classification methods comparison

A further analysis that we performed was related to the comparison between the two classification methods employed in EEG classification for P300 detection: the *probability averaging* method and the *epochs averaging* method. The ratio between the online classification accuracy of the

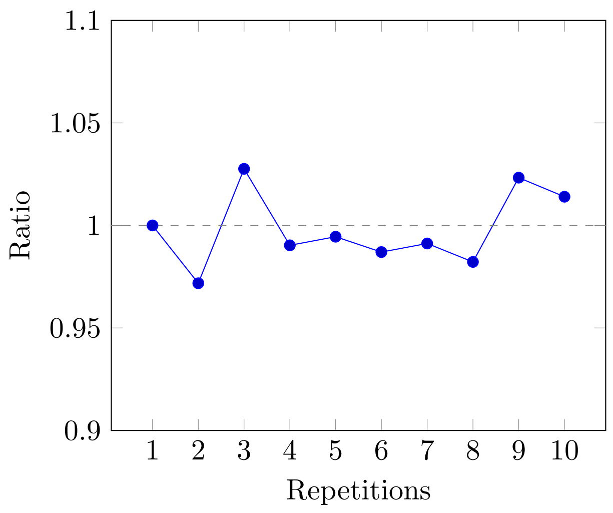


Figure 6.15: Classification accuracy ratio between probability averaging method and epochs averaging method as a function of the number of repetitions.

probability averaging method and the *epochs averaging* method as a function of the number of repetitions is shown in Figure 6.15. Classification accuracy for both subject S1 and subject S2 and for all the BCI modes (speller, cartesian control of the robotic arm, high level control of the robotic arm) was taken into consideration in this analysis. The values of the ratio are very close to 1 ($\mu = 0.998$, $\sigma = 0.018$). Therefore, it is possible to say that there is no difference between the two classification methods. It is important to note that the classifier is trained using *probability averaging* method, therefore this method is be more suitable for real-time classification. However, if EEG signals are highly corrupted by noise, *epochs averaging* method could result in better classification due to noise reduction.

6.3 Android Based BCI

In this section, we report the data obtained in the tests carried out with the Android based BCI. The subject that participated in testing the Android based BCI was subject S1. The locations on the scalp used for EEG recording were Pz, P3, P4, P7, P8 and Oz, according to the 10-20 international standard for EEG recording [13]. As for computer based BCI, the subject was required to undergo to three sessions.

The Android device that was used during the experiments was a smartphone, model Samsung S4. The operating system running on this device was Android 5.0.1 (Lollipop). The device screen diagonal had a length of 5”.

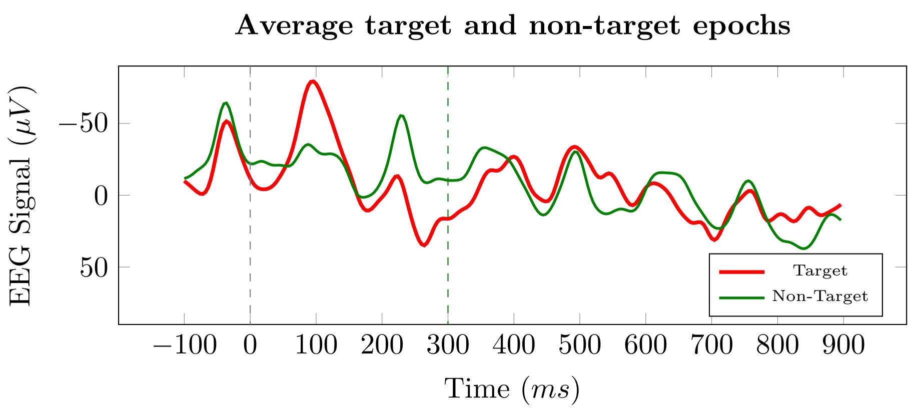


Figure 6.16: Average target and non-target epochs of the training set for subject S1. The training set was obtained using the Android based BCI.

6.3.1 Classifier Training Phase

The first session subject S1 participated in was a training session performed using the Android based BCI. As we already described, during the training session the subject was asked to spell a series of words. In the training sessions, the number of repetitions used was equal to 5. Before the beginning of each stimulation, the letter on which the subject had to focus was highlighted for one second.

In Figure 6.16, we report the average target and non-target epochs computed in the training set for subject S1 In the target epoch, it is possible to notice a peak at about 300 ms after the stimulus onset. As expected, this peak is not present in the non-target epoch, since no P300 should be elicited when a non-target stimulus is provided.

6.3.2 Classifier Validation Phase

As for the computer based BCI, after performing automatic feature extraction and training the logistic regression classifier, the obtained classifier was validated on a validation data set never used before by the genetic algorithm.

In Table 6.8, we report the characteristics of the validation set. In order to assess the

Table 6.7: Characteristics of the training set of subject S1. The training set was obtained with the Android based BCI.

Subject	**Letters**	**Epochs**	**Repetitions**
S1	100	6000	5

Table 6.8: Characteristics of the validation set of subject S1 obtained with the Android based BCI.

Subject ID	Sessions	Letters	Epochs	Maximum Rep.
S1	4	33	3960	10

Table 6.9: Classifier performance statistical measures computed on the validation set obtained with the Android based BCI. *TP:* true positive, *TN:* true negative, *FP:* false positive, *FN:* false negative, *AUC:* area under the ROC curve. The total number of epochs is equal to 3960.

Subject ID	TP	TN	FP	FN	Recall	Precision	AUC
S1	421	2158	1142	239	0.64	0.27	0.65

classifier performance, we firstly computed standard statistical measures of binary classification. The numbers of true positive, true negative, false positive and false negative epochs, together with measures of precision, recall and area under the ROC curve, are reported in Table 6.9.

Then, we evaluated the accuracy of letter spelling as a function of the number of repetitions. The evaluation was done with both classification methods: *probability averaging* and *epochs averaging* method. The results of this analysis are shown in Figure 6.17. For subject S1, an accuracy higher than 80% was reached using 10 repetitions.

6.3.3 Online Classification

After performing classifier training and validation for each subject, we tested the online classification accuracy for the speller mode of the Android based BCI.

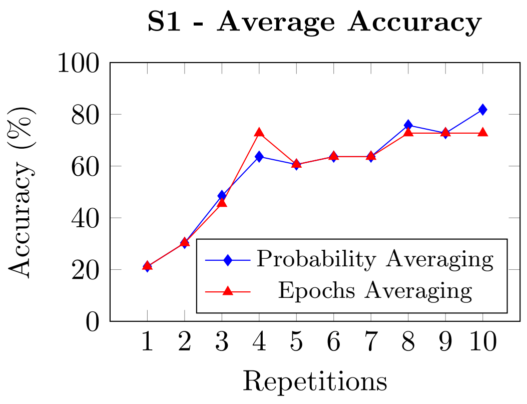


Figure 6.17: Average accuracy as a function of the number of repetitions for the validation set of subject S1. The validation set was obtained using the Android based BCI.

Table 6.10: Online classification accuracy in the speller mode of the Android based BCI.

Subject ID	**Accuracy**	**ITR**	**Number of letters**
S1	82.14%	7.52	28

Subject S1 was asked to use the online speller mode of the Android based BCI, and to spell a sequence of letters. The classification process was performed directly by the application. The number of repetitions used for this experimental test was fixed, and equal to 10.

The results of this analysis are shown in Table 6.10. Using 10 repetitions, subject S1 reached an accuracy of 82.14% during online classification. This corresponds to an ITR of 7.52 *bits/min*.

Chapter 7

Conclusions

In this work, a wearable and cost-effective BCI assistive device was presented. This assistive device is based on the P300 response of the human brain. Since the P300 response is an innate response of the brain, no training for the subject is required to use the developed P300 based BCI.

In the proposed device, EEG signal is recorded using the OpenBCI Cyton board [4]. This board is a low-cost electronic board designed for biological signal acquisition. The user wears a 3D printed headset that houses the electrodes. Due to the portability of the system, the user can use the device wherever he/she desires. Furthermore, the hardware components of the system are cost effective, bringing the total cost of the device to \$ 900. Therefore, the developed device is both wearable and cost-effective, and it could be used by many disabled individuals as an assistive technology device, helping to improve their quality of life by augmenting communication capabilities and allowing for the control of external devices.

The BCI developed in this work can be used with both a computer software and an Android application. Thanks to the Android application, the user can take the BCI assistive device wherever he/she wants to, allowing for the device to be completely portable.

Flashing symbols on a screen, that can be either a computer screen or a mobile device screen, are employed as stimuli to elicit a P300 response. By analyzing the P300 responses in the recorded EEG signal, the device allows the disabled to type words on a computer screen and to control a robotic arm. Regarding the control of a robotic arm, two modalities can be used. One modality (*cartesian* control) consists of controlling the robotic arm with discrete movements. The other modality (*high level* control) consists of sending high level commands to the robotic arm, which will move autonomously according to the selected action.

EEG classification aimed at P300 detection is a fundamental component of any P300 based

Table 7.1: Online classification results in terms of accuracy and information transfer rate. The results shown in the tables were obtained using probability averaging method.

(a) Computer based BCI

	Accuracy		**ITR (*bits/min*)**	
Repetitions	5	10	5	10
Speller	72.62%	88.92%	10.35	7.45
Cartesian robotic arm control	65.23%	88.48%	9.37	8.28
High level robotic arm control	61.77%	79.54%	2.02	2.92

(b) Android based BCI

	Accuracy	**ITR (*bits/min*)**
Repetitions	10	10
Speller	82.14%	7.52

BCI. In this work, the genetic algorithm presented by *Dal Seno et al.* [57] is used to perform automatic feature extraction for P300 detection. Logistic regression is employed for online detection of the P300 response. Two classification methods, both based on logistic regression, were tested: *probability averaging* and *epochs averaging* method. Theprobability averaging method bases its functionality on an average performed in the classification score space. In contrast, the *epochs averaging* method is based on an average performed in the data space, rather than in the classification score space.

The device was validated on two male healthy subjects (20 and 24 years old). Results in terms of accuracy and information transfer rate (ITR) for the computer based BCI, when using 5 and 10 repetitions, are shown in Table 7.1a. For the Android based BCI, the results, in terms of accuracy and ITR, when using 10 repetitions, are shown in Table 7.1b. Amongst the two subjects, the average maximum ITR for the speller mode was 12.19 *bits/min*, for the cartesian control of the robotic arm was 9.95 *bits/min*, and for the high level control of the robotic arm was 4.09 *bits/min*.

The achieved results are comparable to other BCI studies. The comparison we performed was limited to the online speller performance, since no other studies were done with the same

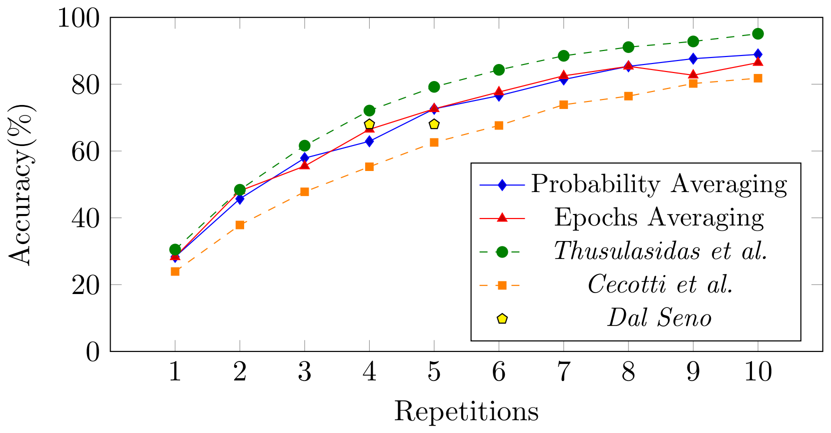


Figure 7.1: Comparison between the present study and *Thusulasidas et al.* [62], *Cecotti et al.* [63], *Dal Seno* [61] for online classification accuracy as a function of the number of repetitions in the speller mode.

setup we used for the control of the robotic arm. The comparison was done with *Dal Seno et al.*, because the same genetic algorithm employed in our system was used, and with *Thulasidas* and *Cecotti et al.* because exact values of accuracy as a function of the number of repetitions were reported in the studies [61, 62, 63]. The result of this comparison is shown in Figure 7.1 In our study, amongst the two subjects the average values of accuracy were 72.62% when using 5 repetitions and 88.92 % when using 10 repetitions. *Dal Seno* reported the results of a P300 speller used online by two subjects [61]. EEG signal was acquired with *BE Light* by EBNeuro [52]. The software used for EEG signal acquisition and processing was BCI2000 [33]. One subject achieved 68% online accuracy when using 4 repetitions. The other subject achieved 68% online performance with 5 repetitions. *Thulasidas et al.* presented a robust classification method for P300 detection [62]. For this study, EEG signal was acquired using Neuroscan SynAmps2 [64]. The software for the P300 speller was developed by the authors. Average accuracy across 9 subjects when using 5 repetitions was 79.2%. *Cecotti et al.* developed seven convolutional neural network based classifiers for P300 detection [63]. To test the classifiers, dataset from the third BCI competition was employed [65]. This data set contains a record of P300 evoked potentials from two subjects. The signals were recorded in five sessions with the BCI2000 software [33]. Average accuracy across the 7 classifiers between the two subjects, when using 5 repetitions, was 62.57%. In this study, data about ITR were also reported. When using 5 repetitions, average ITR amongst the two subjects is in the range of 8 - 13 *bits/min*, according to the chosen classifier. *Kronegg et al.* reported the average value of ITR for several BCI systems, which is equal to 11 *bits/min* [66].

In all the above studies, clinical instrumentation for EEG acquisition was used. Our results on two subjects show that it is possible to obtain comparable results in terms of accuracy and ITR when using cost-effective hardware for EEG acquisition. More advanced techniques for classification could be employed to improve the functionality of the device by increasing the ITR. As an example, *Speier et al.* [67] implemented techniques of natural language processing and dynamic classification for a P300 speller. The use of these methods lead to a great increase in ITR when compared to a standard classification technique, SWLDA (33.15 *bits/min* and 27.69 *bits/min* vs 22.07 *bits/min*).

According to the Amyotrophic Laterals Sclerosis Association, only one BCI device, the Intendix system, is commercially available at the moment [68, 69]. As stated by *Kübler*, low cost EEG acquisition devices render translational studies increasingly feasible [70]. Therefore, we believe that the wearable and cost-effective BCI presented in this work will help in the transition from BCI use in laboratory settings to in-home use, allowing more and more disabled individuals to improve their quality of life.

7.1 Future Improvements

Improvements of the device were identified during the device validation phase, and are as follows:

1. **Improved EEG headset:** the current 3D printed EEG headset may not be comfortable for long term-use. A new design is necessary to improve the comfort of the headset. The new design should also take into account the high electrode-to-skin impedance values that were sometimes obtained with the current version.

2. **Flashing of menu items:** since selection of menu items has to be performed with a mouse in the computer software, and with touch inputs in the Android application, the BCI cannot be initialized by the disabled without the help of a caregiver. Employing automatic flashing menu items, an updated version of the computer software and of the Android application would greatly improve the accessibility of the BCI.

3. **Acoustic BCI**: for subjects severely affected by moto neuron diseases, a BCI system can't be based on visual stimuli, due to reduction in ocular muscles movements and visual attention. Therefore, a P300 based BCI employing acoustic stimuli is more suitable [71]. Among the features that the Android application could offer in the future, there is a P300 based acoustic BCI working with the OpenBCI Cyton board.

4. **High level control of the robotic arm:** with the current firmware of the robotic arm,

when using the high level control the robotic arm expects objects to be in predefined locations. An updated version of both firmware and hardware of the arm would allow the robotic arm to automatically recognize where objects are located, thus improving the functionality of the system.

5. **Validation by disabled users:** for this work, test on healthy subjects were performed. Test on disabled individuals are necessary to discover possible key issues present when the device is used by disabled individuals.

Bibliography

[1] World Health Organization. *Fact Sheet n.384*. World Health Organization, 2013.

[2] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M Vaughan. Brain–Computer Interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791, 2002.

[3] Hans Berger. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570, 1929.

[4] OpenBCI. http://www.openbci.com [Online; Accessed: 05/11/2017].

[5] Terence W Picton. The P300 wave of the human event-related potential. *Journal of clinical neurophysiology*, 9(4):456–479, 1992.

[6] R.I. Damper, J.W. Burnett, P.W. Gray, L.P. Straus, and Rachel A. Symes. Hand-held text-to-speech device for the non-vocal disabled. *Journal of Biomedical Engineering*, 9(4):332–340, 1987.

[7] Masaya Kubota, Yoichi Sakakihara, Yoshiaki Uchiyama;, Atsushi Nara;, Tsutomu Nagata, Hiroshi Nitta, Koh Ishimoto;, Akira Oka, Keizo Horio, and Masayoshi Yanagisawa. New ocular movement detector system as a communication tool in ventilator-assisted werdnig-hoffmann disease. *Developmental Medicine and Child Neurology*, 42(1):61–64, 2000.

[8] Rudi Kobetic, Curtis S To, John R Schnellenberger, Thomas C Bulea, Richard Gaudio CO, and Gilles Pinault. Development of hybrid orthosis for standing, walking, and stair climbing after spinal cord injury. *Journal of rehabilitation research and development*, 46(3):447, 2009.

[9] Ujwal Chaudhary, Bin Xia, Stefano Silvoni, Leonardo G Cohen, and Niels Birbaumer. Brain–Computer Interface–Based Communication in the Completely Locked-In State. *PLoS Biology*, 15(1):e1002593, 2017.

[10] Z. Luo, S. Han, and F. Duan. The development of a smart house system based on brain-computer interface. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1012–1017, 2015.

[11] John E Hall. *Guyton and Hall textbook of medical physiology.* Elsevier Health Sciences, 2015.

[12] Korbinian Brodmann. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues.* Barth, 1909.

[13] Herbert H Jasper. The ten twenty electrode system of the international federation. *Electroencephalography and clinical neurophysiology*, 10:371–375, 1958.

[14] Tonio Ball, Markus Kern, Isabella Mutschler, Ad Aertsen, and Andreas Schulze-Bonhage. Signal quality of simultaneously recorded invasive and non-invasive EEG. *NeuroImage*, 46(3):708–716, 2009.

[15] David Cohen et al. Magnetoencephalography: evidence of magnetic fields produced by alpha-rhythm currents. *Science*, 161(3843):784–786, 1968.

[16] Ranganatha Sitaram, Andrea Caria, Ralf Veit, Tilman Gaber, Giuseppina Rota, Andrea Kuebler, and Niels Birbaumer. fMRI brain-computer interface: a tool for neuroscientific research and treatment. *Computational intelligence and neuroscience*, 2007.

[17] N. Birbaumer. Slow cortical potentials: their origin, meaning, and clinical use. *Brain and Behavior Past, Present, and Future*, pages 25–39, 1997.

[18] G. Pfurtscheller and C. Neuper. Motor imagery and direct Brain-Computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[19] Connie C Duncan-Johnson and Emanuel Donchin. On quantifying surprise: The variation of event-related potentials with subjective probability. *Psychophysiology*, 14(5):456–467, 1977.

[20] Daniel S Ruchkin, Samuel Sutton, Mitchell L Kietzman, and Kenneth Silver. Slow wave and P300 in signal detection. *Electroencephalography and Clinical Neurophysiology*, 50(1):35–47, 1980.

[21] L.A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.

[22] Eric W Sellers, Dean J Krusienski, Dennis J McFarland, Theresa M Vaughan, and Jonathan R Wolpaw. A P300 event-related potential brain–computer interface (BCI): the effects of matrix size and inter stimulus interval on performance. *Biological psychology*, 73(3):242–252, 2006.

[23] Emanuel Donchin, Kevin M Spencer, and Ranjith Wijesinghe. The mental prosthesis: assessing the speed of a P300-based Brain–Computer Interface. *IEEE Transactions on Rehabilitation Engineering*, 8(2):174–179, 2000.

[24] D. J. Krusienski, E. W. Sellers, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw. Toward enhanced P300 speller performance. *Journal of Neuroscience Methods*, 167(1):15–21, 2008.

[25] I. Iturrate, J. M. Antelis, A. Kubler, and J. Minguez. A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation. *IEEE Transactions on Robotics*, 25(3):614–627, 2009.

[26] B. Rebsamen, E. Burdet, Cuntai Guan, Haihong Zhang, Chee Leong Teo, Qiang Zeng, M. Ang, and C. Laugier. A brain-controlled wheelchair based on P300 and path guidance. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, pages 1101–1106, 2006.

[27] Marco Congedo, M. Goyat, N. Tarrin, L. Varnet., B. Rivet, G. Ionescu, N. Jrad, R. Phlypo, M. Acquadro, and C. Jutten. "Brain Invaders": a prototype of an open-source P300-based video game working with the OpenViBE platform. *5th International BCI Conference, Graz, Austria, 280-283*, 2011(Bci):1–6, 2011.

[28] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer. OpenViBE: An Open-Source Software Platform to Design, Test, and Use Brain-Computer Interfaces in Real and Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 19(1):35–53, 2010.

[29] Andrea Finke, Alexander Lenhardt, and Helge Ritter. The MindGame: A P300-based Brain-Computer Interface game. *Neural Networks*, 22(9):1329–1333, 2009.

[30] Mayur Palankar, Kathryn J. De Laurentis, Redwan Alqasemi, Eduardo Veras, Rajiv Dubey, Yael Arbel, and Emanuel Donchin. Control of a 9-DoF wheelchair-mounted robotic arm system using a P300 Brain-Computer Interface: Initial experiments. *2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008*, pages 348–353, 2008.

[31] Kinova Robotics. Kinova Jaco http://www.kinovarobotics.com/service-robotics/build-prices/ [Online; Accessed 03/11/2017].

[32] National Institute of Biomedical Imaging and Bioengineering. https://www.nibib.nih.gov/sites/default/files/R-ARM.pdf, [Online; Accessed 18/02/2017], 2013.

[33] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004.

[34] C J Bell, P Shenoy, R Chalodhorn, and R P Rao. Control of a humanoid robot by a noninvasive Brain-Computer Interface in humans. *Journal of neural engineering*, 5(2 PG - 214-220):214–220, 2008.

[35] Biosemi. https://www.biosemi.com/products.htm [Online, Accessed 01/41/2017].

[36] R. Ortner, B. Z. Allison, G. Korisek, H. Gaggl, and G. Pfurtscheller. An SSVEP BCI to control a hand orthosis for persons with tetraplegia. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(1):1–5, 2011.

[37] Ignas Martisius and Robertas Damasevicius. A prototype SSVEP based real time BCI gaming system. *Computational Intelligence and Neuroscience*, 2016, 2016.

[38] gTech. g.USBamp http://www.gtec.at/products/ [Online; Accessed 11/03/2017].

[39] Emotiv. Emotiv Epoch https://www.emotiv.com [Online; Accessed: 03/05/2017].

[40] Microchip. PIC32MX250F128B Datasheet, http://www.microchip.com/ [Online; Accessed 18/03/2017].

[41] Texas Instrument. ADS1299 Low-Noise, 8-Channel, 24-Bit Analog-to-Digital Converter http://www.ti.com/lit/ds/symlink/ads1299.pdf [Online; Accessed 18/03/2017].

[42] Jérémy Frey. Comparison of an open-hardware electroencephalography amplifier with medical grade device in Brain-Computer Interface applications. *CoRR*, abs/1606.02438, 2016.

[43] OpenBCI. Ultracortex Mark III headset, http://docs.openbci.com/ [Online; Accessed 05/04/2017].

[44] Lynxomotion. AL5B Robotic Arm, http://www.lynxmotion.com/c-126-al5b.aspx [Online; Accessed 18/03/2017].

[45] Lynxomotion. BotBoarduino http://www.lynxmotion.com/ [Online; Accessed 18/03/2017].

[46] Arduino Duemilanove https://www.arduino.cc/en [Online; Acessed 18/03/2017].

[47] Processing Foundation. Processing, version 3.0, build 0253. http://www.processing.org [Online; Accessed 03/03/2017].

[48] Dennis J. McFarland, William A. Sarnacki, George Townsend, Theresa Vaughan, and Jonathan R. Wolpaw. The P300-based brain computer interface (BCI): Effects of stimulus rate. *Clinical Neurophysiology*, 122(4):731–737, 2011.

[49] Fabio Veronese, Hassan Saidinejad, Matteo Matteucci, and Fabio Salice. Communication oriented brain computer interface in a remote monitoring system for amyotrophic lateral sclerosis. In *Assisitive Technology: From Research To Practice: AAATE 2013*, 2013.

[50] Android. https://www.android.com [Online, Accessed 09/10/2017].

[51] Dropbox SDK. https://www.dropbox.com/developers [Online, Accessed 09/10/2017].

[52] EBNeuro. http://www.ebneuro.biz/ [Online; Accessed 18/03/2017].

[53] Bob Kemp and Jesus Olivan. European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9):1755–1761, 2003.

[54] Android API Guide. http://developer.android.com/guide/index.html [Online, Accessed 20/09/2017].

[55] Carlo Vercellis. *Business intelligence: data mining and optimization for decision making.* John Wiley & Sons, 2011.

[56] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[57] Bernardo Dal Seno, Matteo Matteucci, and Luca Mainardi. A genetic algorithm for automatic feature extraction in P300 detection. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3145–3152, 2008.

[58] SN Sivanandam and SN Deepa. *Introduction to genetic algorithms.* Springer Science & Business Media, 2007.

[59] R. Boostani, B. Graimann, M. H. Moradi, and G. Pfurtscheller. A comparison approach toward finding the best feature and classifier in cue-based BCI. *Medical and Biological Engineering and Computing*, 45(4):403–412, 2007.

[60] Luca Citi, Riccardo Poli, Caterina Cinel, and Francisco Sepulveda. Feature Selection and Classification in Brain Computer Interfaces by a Genetic Algorithm. *Late breaking papers of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 2004.

[61] Bernardo Dal Seno. Toward an integrated P300 and ErrP based Brain-Computer Interface. *Politecnico di Milano*, 2009.

[62] M Thulasidas, Guan Cuntai, and Wu Jiankang. Robust classification of EEG signal for Brain-Computer Interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(1):24–29, 2006.

[63] Hubert Cecotti and Axel Graser. Convolutional neural networks for P300 detection with application to Brain-Computer Interfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):433–445, 2011.

[64] Neuroscan. Neuroscan synamps http://compumedicsneuroscan.com/tag/synamps/ [Online; Accessed 18/03/2017].

[65] Benjamin Blankertz, K-R Muller, Gabriel Curio, Theresa M Vaughan, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, et al. The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials. *IEEE transactions on biomedical engineering*, 51(6):1044–1051, 2004.

[66] Voloshynovskiy S Pun T Kronegg J. Analysis of bit-rate definitions for Brain-Computer Interfaces. *Proceedings of the 2005 International Conference on Human-Computer Interaction, HCI 05*, pages 40–46, 2005.

[67] William Speier, Corey Arnold, Jessica Lu, Ricky K Taira, and Nader Pouratian. Natural language processing with dynamic classification improves P300 speller accuracy and bit rate. *Journal of neural engineering*, 9(1), 2012.

[68] ALS. http://www.alsa.org/ [Online, Accessed 01/04/2017].

[69] Intendix. http://www.intendix.com/ [Online, Accessed 01/04/2017].

[70] A. Kübler. Quo vadis P300 BCI? In *2017 5th International Winter Conference on Brain-Computer Interface (BCI)*, pages 36–39, 2017.

[71] Emanuele Opna De Bernardi. Development of a portable auditory P300-based Brain-Computer Interface for yes-no communication to amyotrophic lateral sclerosis. 2015.