POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

On the Engineering and Control of a Robotic Arm via Brain Computer Interfaces

Supervisors

Prof. Francesco P. ANDRIULLI

Candidate

Gijon KANJIRATHUNKAL JOY

ACADEMIC YEAR 2020-2021

Summary

The brain, together with the other components of the nervous system, uses electrical signals to control most of the activities of our body. Lot of studies in literature show that it is possible to use these signals to make the brain interact with computers. In this work, by relying on external measurement of brain signals obtained with electroencephalography (EEG), we used a Brain-Computer Interface (BCI) to control a robotic arm that mimic human movements.

The BCI can be driven with EEG via different kinds of brain signals, so that we can classify BCIs into two main groups:

- endogenous BCIs, in which the signals are generated by the own user, without the usage of external stimuli;
- exogenous BCIs, in which stimuli from the environment are adopted to elicit the brain activity of the user. This creates an evoked signal with particular features, that can be used to classify them after the recording.

In this work we focused on the latter typology, and in particular on Steady-State Visual Evoked Potentials (SSVEPs), which can be induced in the brain by showing to the user flickering visual stimuli at predefined frequencies.

The objective of this thesis was to use the BCI technology to induce the movement of mechanical arm, so that it would be able to perform a simple task like picking and re-positioning an object from a cell of a chessboard to another location, both chosen by the user. In particular, a subject was posed in front of different visual stimuli, patterns flickering at various frequencies, that created a particular brain activity. The activity went monitored through EEG, which is a non-invasive method that allows the recording of electric potentials coming from electrodes placed on the scalp of the user. The EEG has a good time resolution despite its poor spatial resolution. Therefore, it was feasible to monitor the evolution in time of the signal we measured, allowing as to differentiate the visual stimuli based on their flickering frequencies. By using different flickering frequencies in different parts of a display, we could detect in which part the user was focusing in.

In our implementation, the collected measurements were pre-processed so that noise components that usually affect them were reduced. The pre-processed data

were then analyzed so that particular features could be extracted, allowing their classification via machine learning techniques. The output of the classification was interpreted and used as input for the shield controlling the mechanical arm, specifying the cells from/to which move the object.

The personal contribution in the work consisted in the realization of the code that controls the movement of the mechanical arm, in the modifications done to the mechanical arm in order to improve the accuracy of the movement, in the connection of the mechanical arm with the BCI system, in the collection of data for the training (with the tuning of most relevant parameters), and in the modification done in the Unity application in order to reproduce a chessboard game application.

As results of the work, we were able to control through brain signals the mechanical arm and with it move an object around cells of a 6x6 chessboard. The working framework has been recorded in a demonstration video.

Acknowledgements

I would like to thank my supervisor, Prof. Francesco Paolo Andriulli, for the opportunity to work on this project, eng. Davide Consoli and eng. Ermanno Citraro for the suggestions received and eng. Paolo Ricci and eng. Arturo Micheli for the help they gave me without which I would have not been able to reach concrete results.

I would also like to thank my family for the support and finally Edisu Piemonte, for the scholarships that helped me with books, transportation and so on during these years.

Table of Contents

List of Figures

Acronyms

BCI
Brain-Computer Interface

CPU
Central Processing Unit

EEG
Electroencephalography

EPSP
Excitatory Post-Synaptic Potential

GPU
Graphic Processing Unit

IPSP
Inhibitory Post-Synaptic Potential

PSD
Power Spectral Density

RAM
Random Access Memory

SSVEP
Steady-State Visual Evoked Potentials

SVM
Support Vector Machine

TCP

Transmission Control Protocol

Chapter 1

Introduction

The brain is an extremely complex organ but it bases its function on electrical signals.We have to say that it is not possible to identify with great accuracy when and what element was the cause of a certain change in the electrical quantity measurement we are looking at, but in the 70' it was discovered that looking towards a flickering light generates a signal that can be measured in the occipital area of the scalp which has, in the frequency domain, pick value at the frequency at which flickers the input light; this signal can be measured through EEG and can be used in order to give input to some other application: this is what we will try to do in this thesis.

In the following chapters we will discuss some general info about the brain, the mechanical arm, SVM and we will show the main setup required for the work both in the hardware and software side.

In chapter 4 we will discuss more in detail how was organized and modified the part related to the mechanical arm while in chapter 5 we will analyze the connection of the mechanical arm with the BCI and the pipeline that realize that.

The last chapters will be dedicated to show the results in both training and testing, analyze them and make reflection on what has been done.

Chapter 2

Background

In this chapter we will do an overview of the brain, exploiting the anatomy, the elements that compose it and some of its functionalities and we will give some information on the robotic arms. The general information contained in the part related to the brain was taken from [1] and [2].

2.1 Brain anatomy

The brain is mainly composed of 10^{10} to 10^{11} neurons, each of them is connected to thousands of other neurons creating the so called synapses that are about 10^{14}; in its anatomy it is divided in left and right hemispheres, each of them divided in four lobes in charge of different function and control of different parts of the body:

- Frontal lobes: related to emotional memory and speech other than controlling muscle and movement;
- Parietal lobes: related to somatosensory;
- Occipital lobe: the region we are interested most in this thesis because related to the visual;
- Temporal lobe: related to olfactory and auditory capability;

2.1.1 The neuron

The neuron is the basic functional unit of the brain. It is composed by:

- cell body that contains the nucleus and from which dendrites and axons branch;

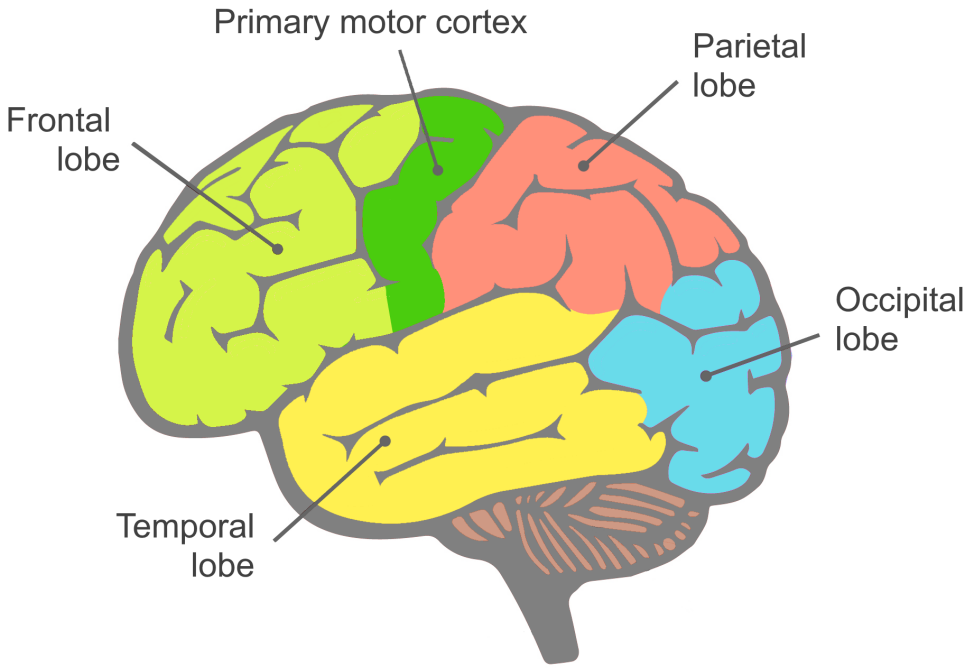


Figure 2.1: Functional areas of the brain. Image taken from [2].

- dendrites which are used by the neurons for receiving the electrical signal coming from the neighbour neuron;
- the axon: is the extension of the cellular body in which the electrical signals travels;

The neurons, thanks to this composition have the properties of excitability (they can generate electrical impulses) and conductivity (ability to propagate the electrical impulses). The electric signals are generated by electro-chemical processes inside the neurons produced by the so-called neurotransmitters. The excitability of the neurons depends on:

- the existence of a membrane electric potential which is a difference of electrical charge between the internal and the external side;
- the presence of ionic channel created by trans-membrane proteins that allows a flux of ions responsible for the difference electrical potential;

The neurotransmitter cause two different electrical phenomena:

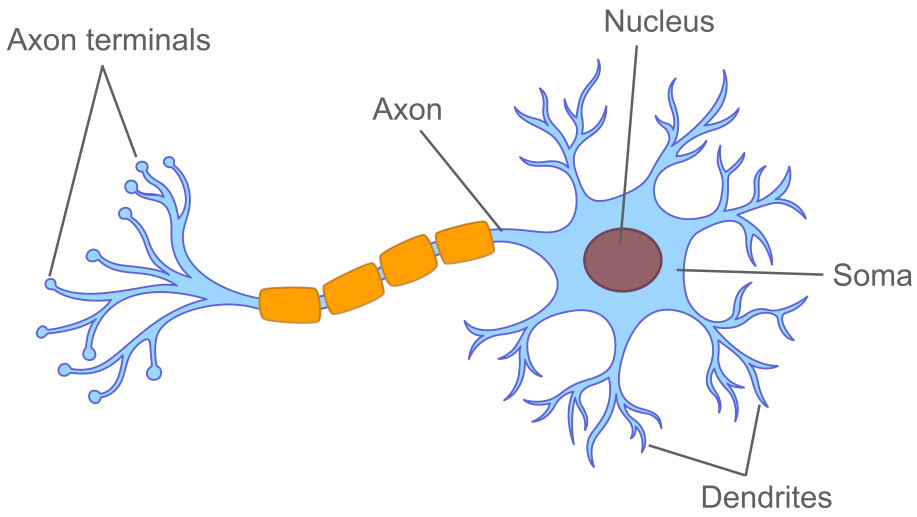


Figure 2.2: Structure of a neuron. Image taken from [2].

- Excitatory post-synaptic potential (EPSP) that is the depolarization of the post synaptic membrane potential due to incoming positive ions flux inside the membrane thanks to ionic channel;

- Inhibitory post synaptic potential (IPSP) that is the contrary of EPSP;

This two phenomena are summed together and if they overcome a certain threshold, an electrical signal is fired and propagates to the neighbour neuron cell; The information is not represented by these electrical signals themselves but by their timing and frequencies.

2.2 Brain activity monitoring

As said before the information transmitted by the neurons through electro-chemical signal can be observed from the surface of the head. One of the most used methods for reading them is the use of electroencephalography (EEG). It is based on the reading of the potential difference measured from electrodes put over the scalp; These signals range from $2\mu V$ to $200\mu V$; the electrodes can be put on the head according to different schemes; in this thesis we are interested in putting electrodes (in number of three as in [3]) on the occipital area and measuring the difference of potential with respect to a reference.

2.2.1 EEG

The EEG measures the electrical activity of the brain; it is a non invasive technique that allows a good temporal resolution despite a not so good spatial resolution; some electrodes are put on the scalp according to the type of sign we want to measures; in this thesis we will use three electrodes put in the occipital area related to the vision, one ground electrode and a reference one put in contact with the ear of the user. There are two type of electrodes that can be used by the EEG:

- Wet electrodes: we insert a particular gel in the electrode in order to create better contact between the head and the electrodes;
- Dry electrodes: they base their function on spring-loaded element that keep in position the electrodes in the site; it is a more noisy method with respect to the wet electrodes;

When using the EEG the band frequencies we can see are related to different functions of the mind:

- <4Hz: detectable in babies and deep sleeping adults;
- 4-7Hz: sleep state and meditation;
- 8-12Hz: related to relaxation states and observable in the occipital area [1, 4];
- 12-30Hz: visible during movement and motor imagery;
- 30-100Hz: cannot be recorded by EEG and are related to sensory stimulation [1, 5];

2.3 Brain Computer Interface (BCI)

A Brain Computer Interface allows the control of element in response of electrical signal measured in the brain. There are three main systems that intervene in the BCI:

- Brain acquisition system: a system that allows to read the electrical signals of the brain: in this thesis we will use the EEG;
- Computational system: the signals coming from the brain acquisition system are processed in order to reduce noise, enhance particular characteristic and classify the data so as to obtain the correct command input to give to the application connected to the BCI (in this work the mechanical arm);
- Interactive system: the one that according to the previous system takes the corresponding output: in our case it is the mechanical arm and its Arduino controller;

2.3.1 SSVEP

In order to generate in the brain of the user an identifiable signal we will use the Steady-State Evoked Potential (SSVEP), a kind of visual evoked potential generated by letting the user look at a flickering light. When the eyes of the user look at the light at a given frequency in his brain (occipital area) it is generated a signal which has a peak (in the frequency domain) at the same frequency at which the light flickers. Consequently this phenomena can be used for making the user choose between different light at which look at in order to generate different brain stimulus according to which light he was seeing.

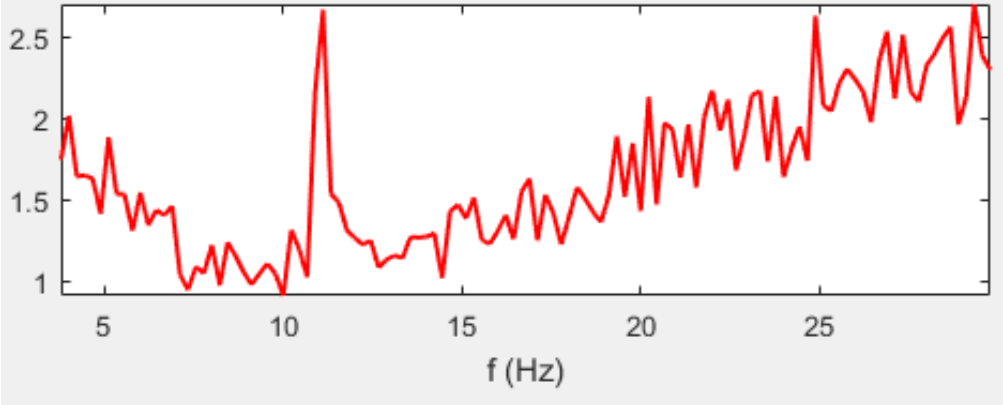


Figure 2.3: SSVEP signal example with distinguishable peak at 11Hz.

2.4 Robotic arm

There are several types of robotic arms. Here we describe some of characteristic and properties. The main information of this part is taken from the material of the course Robotics held by professor Rizzo. A robotic arm is an instrument able to manipulate objects in a work space by means of its joints and links/arms. Inside the joints generally are positioned the motors that allows the movement of the mechanical arm and allow relative motion between contiguous links. We can find two type of joint:

- Revolute: joint allowing rotation between the links;
- Prismatic: joint allowing linear movement between links;

We have at this point to introduce some important concepts that will be exploited in the next pages:

- Work space: region described by the end-effector of the robotic arm when all the joints execute all possible motions; an example of work space is shown in fig. 2.4;

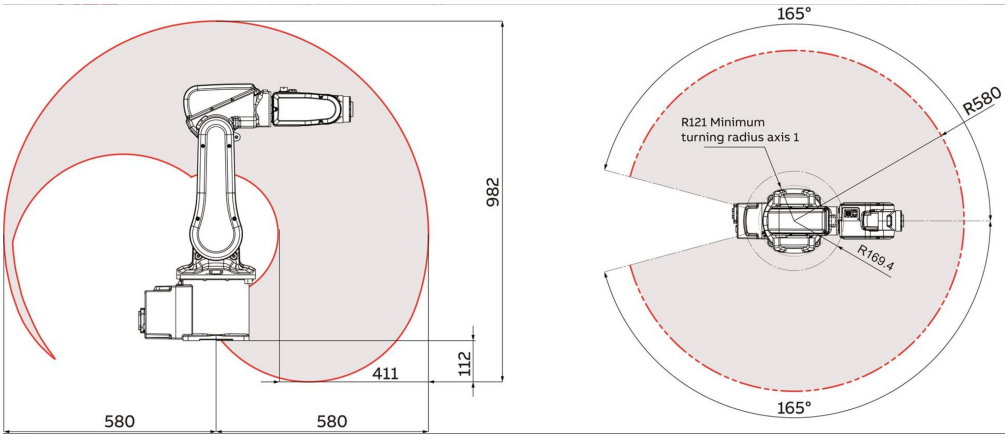


Figure 2.4: Example of a Work space for anthropomorphic robot; Image taken from `https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-data`

- Accuracy: discrepancy between the posture attained through the actuation of the joint variables and the one computed through the solution of the direct kinematic problem;
- Repeatability: ability of the manipulator to return to a previously reached position;

According to the arm configuration related to the type of the joint, as shown in fig. 2.5 we can classify a robot in:

- Cartesian: composed by three prismatic joints with a parallelepiped task space and guarantee good positioning but small dexterity;
- Cylindrical: composed by one rotational joint and two prismatic joints, with a cylindrical sector as task space and accuracy that decreases toward the arm end;
- Polar: composed by two rotational joints and a prismatic joint, with a spherical sector as task space; also in this case the accuracy decreases with toward the arm end;
- Scara: composed by two rotational joints and a prismatic joint; mainly used for small component manipulation;

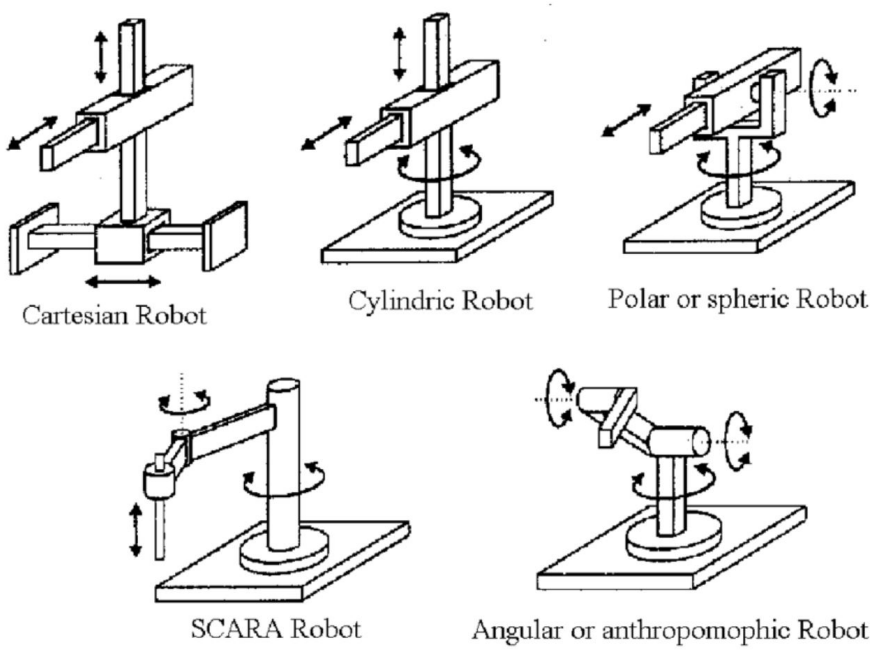


Figure 2.5: Different type of robot according to joint type; image taken from: `https://www.soulaxdo.top/ProductDetail.aspx?iid=506120716&pr=39.99`

- Articulated/Anthropomorphic: composed by three rotational joints with a sphere sector as task space; the accuracy is not constant all over it;

2.4.1 Inverse Kinematic solution for a planar manipulator

In the following lines we will show some calculations that will be re-used successively in the thesis. In particular, such results will be relevant in chapter 5 section 5.1 for solving an inverse kinematic problem,i.e. to calculate the variable associated to each joint of the robotic arm that allow to reach a certain position of the end-effector.

In particular we focus our attention on the case of a planar manipulator with 3 arms, shown in fig. 2.6. The known element in this problem are ϕ, p_x, p_y and the dimension of the arms a_1, a_2, a_3.

From the figure we can say that :

$$\phi = \theta_1 + \theta_2 + \theta_3$$

To make the notation clear, we indicate, for example, with c_1 the $cos\theta_1$, with s_1 the $sin\theta_1$ and with c_{12} the $cos(\theta_1 + \theta_2)$; knowing that and referring always to fig. 2.6 we can write:

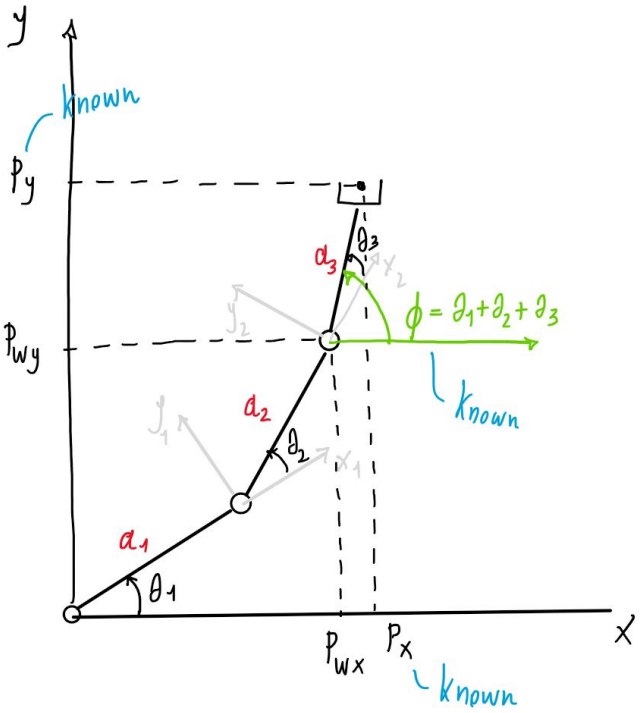


Figure 2.6: Planar manipulator with 3 arms

$$p_{Wx} = p_x - a_3 c_\phi = a_1 c_1 + a_2 c_{12} \tag{2.1}$$
$$p_{Wy} = p_y - a_3 s_\phi = a_1 s_1 + a_2 s_{12} \tag{2.2}$$

From the 2 formula above we obtain:

$$p_{Wx}^2 + p_{Wy}^2 = a_1^2 + a_2^2 + 2a_1 a_2 c_2$$

from where we can say that:

$$c_2 = \frac{p_{Wx}^2 + p_{Wy}^2 - a_1^2 - a_2^2}{2a_1 a_2}$$
$$s_2 = +\sqrt{1 - c_2^2}$$

so that we can find that:

$$\theta_2 = Atan2(s_2, c_2)$$

Knowing θ_2 it is possible to reuse (2.1) and (2.2) to obtain, as solution of a system of equation using Cramer method:

$$s_1 = \frac{(a_1 + a_2c_2)p_{Wy} - a_2s_2p_{Wx}}{p_{Wx}^2 + p_{Wy}^2}$$
$$c_1 = \frac{(a_1 + a_2c_2)p_{Wx} + a_2s_2p_{Wy}}{p_{Wx}^2 + p_{Wy}^2}$$

and consequently θ_1:

$$\theta_1 = Atan2(s_1, c_1)$$

Knowing θ_1, θ_2 and ϕ we can determine θ_3:

$$\theta_3 = \phi - \theta_2 - \theta_1$$

this three angles(θ_1, θ_2, θ_3) allows the end-effector of the robotic arm to reach the position p_x and p_y in a planar space; this calculation will be used in chapter 5 because it is at the basis of fcn1.

2.5 SVM Basis

The aim of this thesis is to control the mechanical arm by means of brain signals. Such signals, can be labeled accordingly with the goal of the user during a setup phase. This procedure allows to generate a dataset that is then utilized to train a classifier. We will now describe some of the principles that are at the basis of the classifier we will use in this work: a SVM (Support Vector Machine) classifier. There are several paper that threats the subject such as [6–11]. We used, in particular, some results from [12].

Lets consider to have a binary classification task with data set x_i $(i = 1, ..., m)$ and label to assign $y_i = \pm 1$. The classification function we are searching will have the following form:

$$f(x) = sign(wx - b)$$

Where w indicates the orientation of the plane that we will use to divide the data set into two class while b is the offset of that plane with respect to the origin, as shown in fig. 2.7. As we can see from the figure, there are a lot of planes that can be used for dividing the due classes of data. Although, we will choose the one that is "furthest" from both, as written in [12]. Referring to fig. 2.8 supposing that it is possible to say (for data labelled as class1):

$$wx_i - b \geq \kappa$$

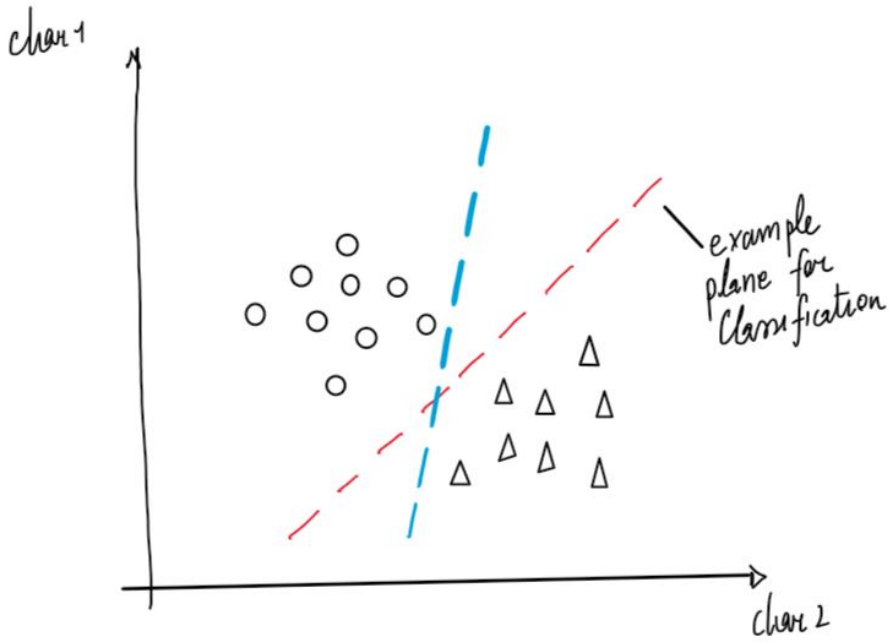


Figure 2.7: Two possible linear discriminant planes

(We have to put a $\leq$ instead of $\geq$ and at the left of the $\leq$ a minus sign for the data that belong to class labelled as -1). The above formula can be re-scaled and become:

$$wx_i - b \geq 1$$

The dotted lines represent the so called supporting plane and their equation is the following:

$$wx_i - b = 1$$
$$wx_i - b = -1$$

the margin is the distance between these two supporting plane and has form $\gamma = 2/\|w\|^2$.

In order to find such a plan that maximize the margin there is a Quadratic Problem to solve:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & wx_i \geq b + 1 \qquad y_i \in Class1 \\ & wx_i \geq b - 1 \qquad y_i \in Class - 1 \end{aligned} \tag{2.3}$$

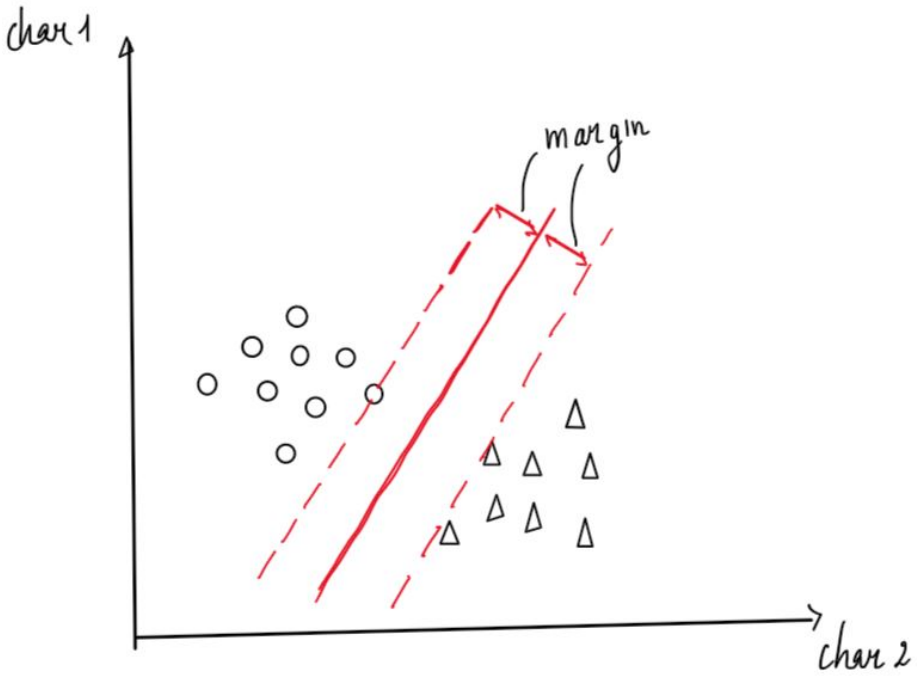


Figure 2.8: Plane that maximize the margin

that correspond to the following Lagrangian Dual problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j x_i x_j - \sum_{i=1}^{m} \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i \\ & \alpha_i \geq 0 \qquad\qquad i = 1, ..., m \end{aligned} \tag{2.4}$$

In the case we do not have two data sets linearly separable as it was in the case shown before, it is not possible to use the method described in order to obtain the plan that maximize the margin; instead, we have to modify the quadratic problem and the corresponding Lagrangian Dual in order to take consideration of points falling in the wrong side of the supporting planes, as shown in fig. 2.9. The

Quadratic Problem become the following:

$$\begin{aligned} \min_{w,b,z} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{t} z_i \\ \text{s.t.} \quad & y_i(wx_i - b) + z_i \geq 1 \\ & z_i \geq o \qquad\qquad i = 1, ..., m \end{aligned} \tag{2.5}$$

where z are non-negative slack or error variables; the corresponding Lagrangian Dual problem is the following:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j x_i x_j - \sum_{i=1}^{m} \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i \\ & C \geq \alpha_i \geq 0 \qquad\qquad i = 1, ..., m \end{aligned} \tag{2.6}$$

In conclusion we have to say that these were the basic principles above which the classifier that will be used in this thesis works. In the example above we have used only two classes data, but what we have discussed can be extended to multiple classes case, creating multiple model (the Classification function $f(x) = sign(wx - b)$), one for each class, using the One vs All setting. In this kind of setting a separate classifier for each class is trained in order to discriminate whether a sample is part or not of a specific class as written in [2].

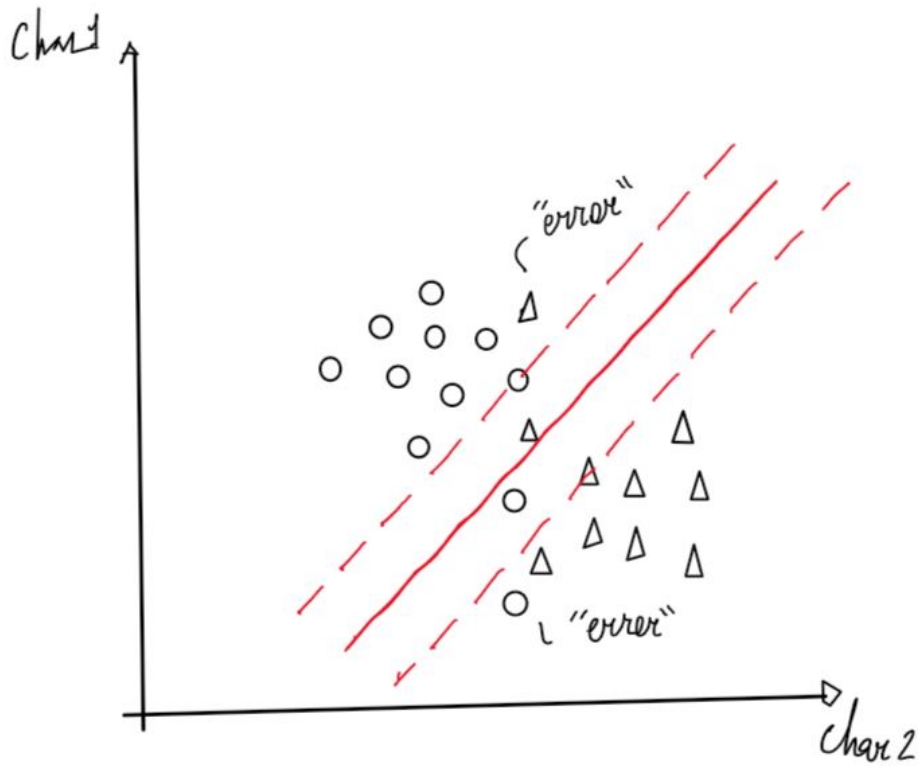


Figure 2.9: select plane to maximize margin and minimize error

Chapter 3

Project setup

In this chapter we will describe the main elements, both software and hardware that will be used in the work

3.1 Hardware

The main component are:

Laptop and workstation

The main laptop used in this thesis is the one of the CERL lab, where the most expensive computations are done, the data are collected and the controller of the mechanical arm attached; this computer has the following specification: a Intel Core i7 9700K @ 3.60GHz CPU, a Nvidia GeForce RTX 2080Ti GPU, and 64GB RAM; This computer uses Windows 10.

Elogoo Uno R3

It is a controller sold that is the corresponding of Arduino Uno (we have to remember that Arduino is an open source project); this is a controller that will be used in this work for controlling the mechanical arm; online there is a lot of documentation and forum for solving possible problems and it is compatible with the mechanical arm selected for the thesis: this is the reason why it was chosen for this application. In the fig. 3.1 it is shown the controller.

Tinkerkit Braccio

It is a robotic arm that can be controlled through an Arduino shield; the movement are allowed thanks to the presence of 6 servomotor in the junction; the Braccio is

Figure 3.1: Controller Elogoo Uno r3; image taken from `https://www.elegoo.com/products/elegoo-uno-project-super-starter-kit`

shown in fig. 4.1. Here we report some useful specification:

Figure 3.2: Tinkerkit Braccio. Image taken from `https://www.robotstore.it/Braccio-Robotico-Tinkerkit-Kit-di-montaggio?gclid=CjwKCAjwqeWKBhBFEiwABo_XBhPY-1O6lP8Ix70jofVX-7Wlfrg_Tm6D4dgE806lj9tLiw8-IjBl5RoCxFkQAvD_BwE`

- complex weight: 800g
- max operation distance: 80cm
- load capability: 150g at 32cm operational distance; 400g at minimum operational distance

Specification about the servo motors:

- Control Signal: PWM analog
- Max torque: 14,5kg-cm
- weight : 62g
- rotation range : 180deg

EEG

For the EEG we will use g.HiAmp that can handle till 32 electrodes and that can be used with wet measurement(with the gel that allows better accuracy); the signal acquired with that method can be handled through Simulink model. The Amplifier is shown in fig. 3.3 and the Helmet used during the test is shown in fig. 3.4

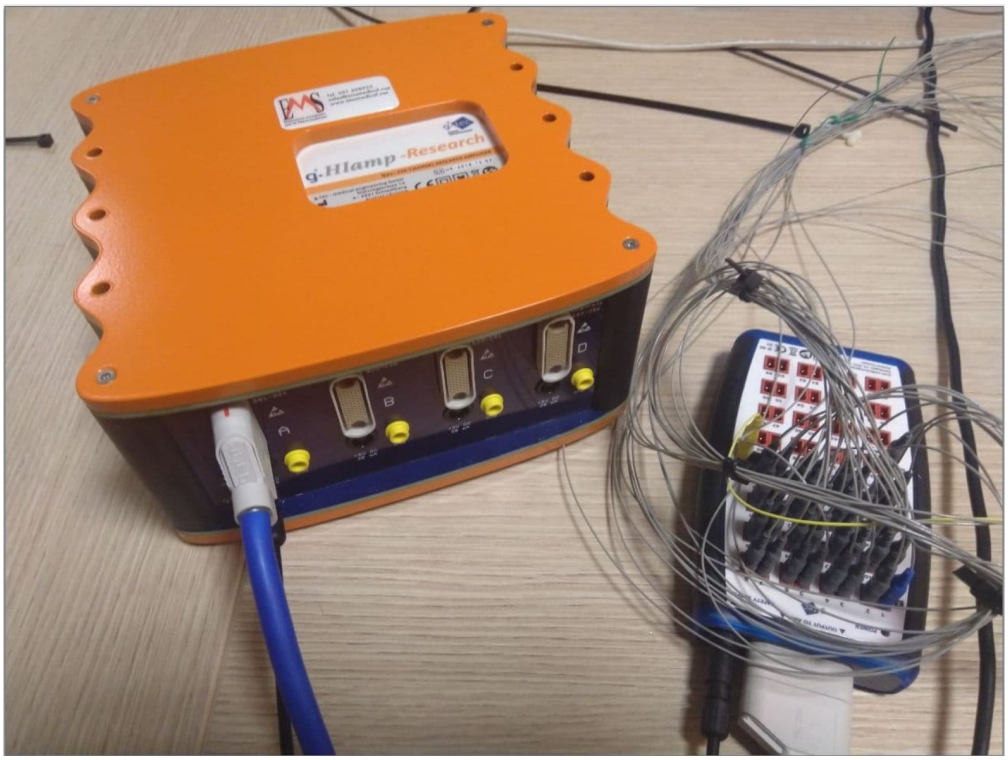


Figure 3.3: Amplifier used during the experiment and test.

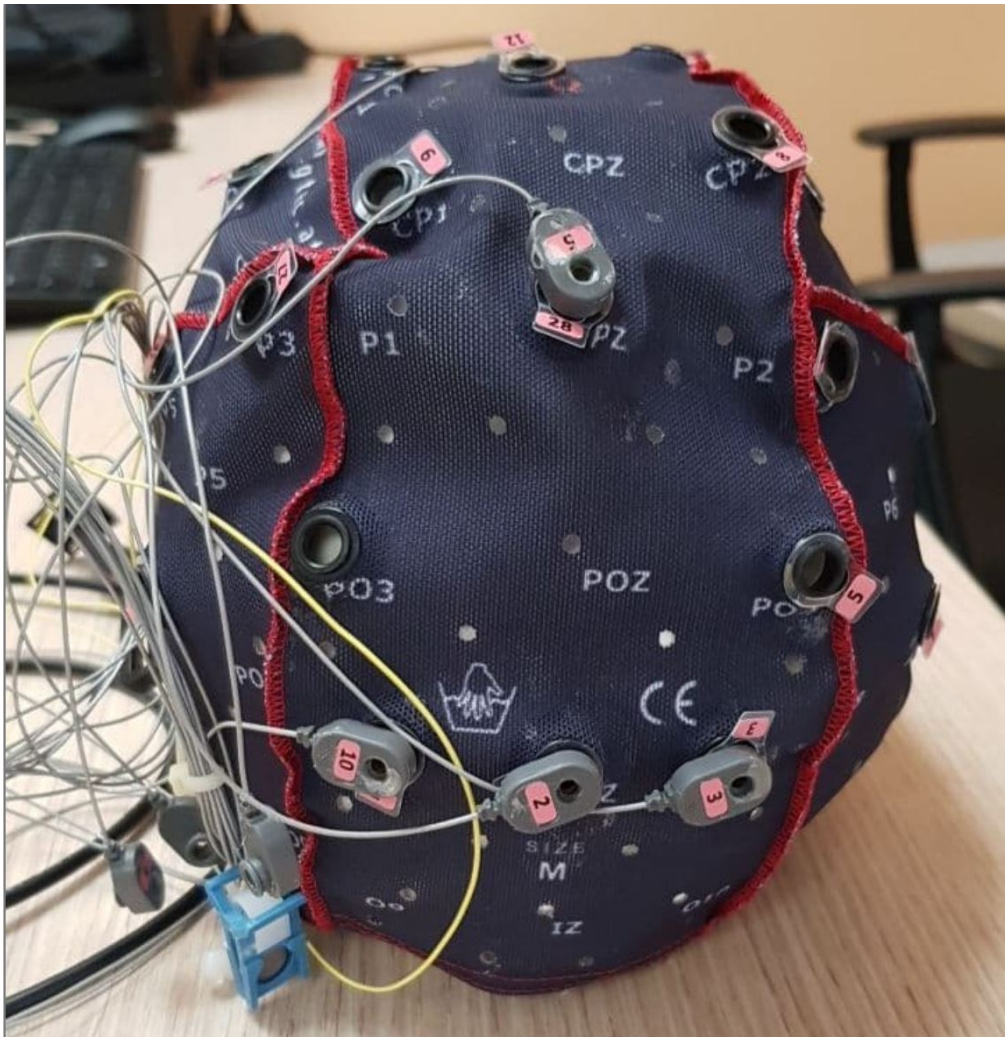


Figure 3.4: Helmet used during the experiment

3.2 Software

The software that were used for the realization of the pipeline, the realization of code for controlling and the acquisition of data are:

Unity Engine

As written in [2], Unity Engine is a cross-platform game engine with an associated editor. It is a software mainly used for the development of video games in 2D and 3D; the project in unity are organized in scenes and Games Object which can be connected to mesh or script. In this thesis this software will be used for providing

to the user different flickering stimulus; connecting Unity with the Simulink model dedicated to the acquisition and processing of data through a socket, it is possible to coordinate the flickering in order to have it in the correct timing; in this way it is also possible to see if the Simulink model has chosen the correct input to give to the mechanical arm according to the flickering light the user was looking at, showing it on the unity application itself.

MatLab and Simulink

MatLab Simulink is a program that allows engineering simulation; it has a lot of tools and facilities that can be used in different kind of situation. In this thesis it will be used for creating the pipeline for collecting and processing data coming from the brain.

Arduino IDE

It is the ambient for creating script to use for controlling the Arduino and the connected hardware.

DesignSparkMechanical

A program for the 3d design; the model created can be 3D printed

Chapter 4

Mechanical Arm

In this chapter will be described how the mechanical arm used in the thesis was chosen, some problem that had to be faced, their solution and also the modification done to the mechanical arm.

4.1 Choosing of the mechanical arm

We have to say that at first the aim of this thesis was simply to connect a mechanical arm to the BCI and perform some simple movements using SSVEP signals for this purpose. With this consideration the objective at the beginning was to find a simple mechanical arm with at least six degrees of freedom similar to a human arm that was at the same time not too expensive: several options where considered included some old industrial mechanical arms, but at the end it was decided to use the Arduino Tinkerkit (shown in fig. 4.1). The Arduino Tinkerkit is a quite inexpensive option compared to other alternatives, but still suitable for our purpose.

The Arduino Tinkerkit is an anthropomorphic robot with three revolute joints; the structure is similar to the one of a human arm. The "task Space" of this Robotic arm is a sector of sphere but the accuracy of the arm movement is not constant allover the task space even though it has got a good dexterity. Its end-effector is a simple gripper but when needed it can be substituted with other instruments.

After the arm was purchased the task we wanted to perform with it was delineated with a greater precision; try to use the mechanical arm for playing chess (we will see that we were not able to complete this task); the BCI part was needed for the selection of the cell in which put the pawn. These information were passed to the Arduino controller which made the arm move consequently so that it was able to take the pawn in the selected cell and put it in the other selected cell.

Figure 4.1: Tinkerkit Braccio choosen. Image taken from `https://www.robotstore.it/Braccio-Robotico-Tinkerkit-Kit-di-montaggio?gclid=CjwKCAjwqeWKBhBFEiwABo_XBhPY-1O6lP8Ix70jofVX-7Wlfrg_Tm6D4dgE806lj9tLiw8-IjBl5RoCxFkQAvD_BwE`

4.2 Connection Arm with the Arduino and sizing of the problem

One of the first thing done with the mechanical arm was connecting it with Arduino controller in order to perform initially some simple operation like taking object from one place and moving it in another one using simulated input and not the one coming from the BCI; in the flow chart shown fig. 4.2 it is resumed the flow of data that will be used in this case.

The input come from a Simulink block; inside the Arduino controller a code is uploaded so that the input can be handled and the mech arm moved. The code is created inside the Arduino IDE (an Arduino software for the programming). The connection between Arduino and the Simulink scheme that generate the input is created as shown in fig. 4.3.

The fcn block is used for an early elaboration of the input data: the input consist in a constant block with the value of the cells we want to use for taking and putting the pawn. In the next chapter we will analyze a little further the function

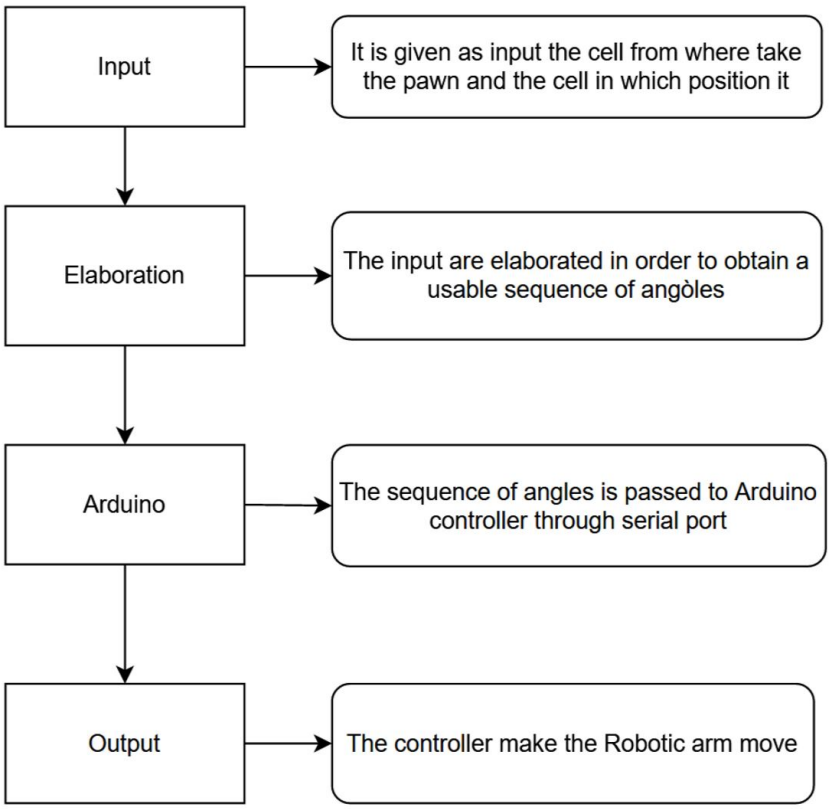


Figure 4.2: Flow chart that resume data flow

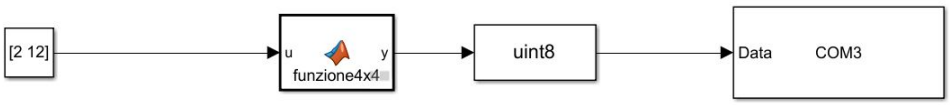
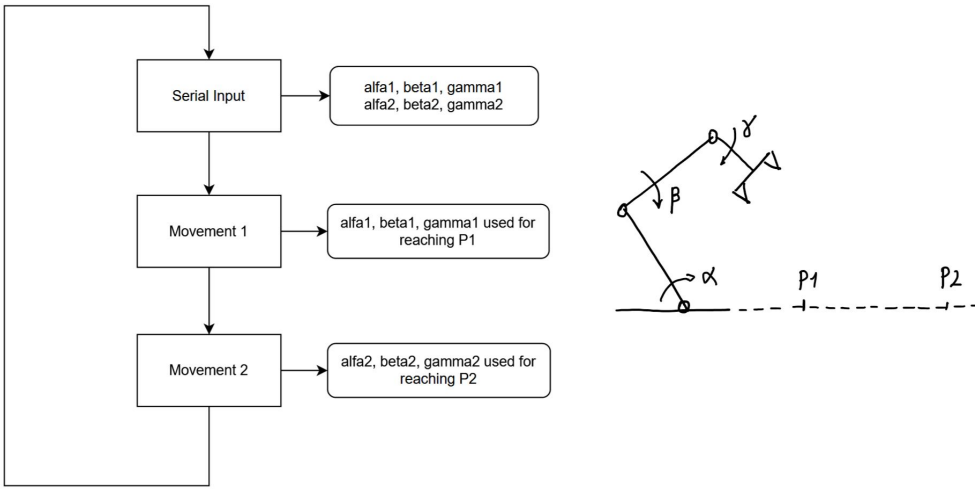


Figure 4.3: SImulink scheme from input to Arduino.

fcn (funzione 4x4) and their different versions; for now it is sufficient to know that they are used for elaborate the data and give as output an angle sequence that then will be given to the Arduino controller to generate the movement of the robotic arm.

4.3 Program uploaded on Arduino

How it was said above, the input effectively given to the Arduino controller is a angle sequence; the flow chart below shows a scheme of what is implemented in the controller.



The part of the code that allows the reception of info from Simulink is shown below:

```
    a=Serial.read();//the 'header 'H' is read
    delay(20);//NOTICE: the delay are fundamental
    //info about the first cell
    angoli[0]=Serial.read();// number of step for the stepper
    delay(20);
    sign[0]=Serial.read();//info about the direction in which make
   the stepper move
    delay(20);
    angoli[1]=Serial.read();
    delay(20);
    angoli[2]=Serial.read();
    delay(20);
    angoli[3]=Serial.read();
    delay(20);
    //info about the second cell
    angoli[4]=Serial.read();// number of step for the stepper
    delay(20);
    sign[1]=Serial.read();//info about the direction in which make
   the stepper move
    delay(20);
    angoli[5]=Serial.read();
    delay(20);
```

```
21      angoli[6]=Serial.read();
22      delay(20);
23      angoli[7]=Serial.read();
24      delay(20);
25      b=Serial.read();// terminator '/r' is read
26      delay(20);
```

The part of the code that is used for moving the mechanical arm is similar to the code lines shown below:

```
1       Braccio.ServoMovement(20,90,angoli[1],angoli[2],angoli
    [3],90,10);//Braccio Movement
2       delay(2000);
3       Braccio.ServoMovement(20,90,angoli[1],angoli[2],angoli
    [3],90,73);//Braccio close the gripper
4       delay(2000);
5       Braccio.ServoMovement(20,90,45,180,180,90,73);//Braccio return
    to original position
6       delay(2000);
7       digitalWrite(dirPin,LOW);
```

This code in the following of the thesis will remain almost the same because it allows to realize the two movement of taking and putting a pawl in a chess board after the arriving of input coming from Simulink. At the same time, the Simulink scheme, as we will see in the following, presents different version in order to solve some problem that showed up and which will be described in the next paragraphs.

4.4 Limits of the mechanical arm

We uploaded on the Arduino controller a code (whose principles are shown in fig. 4.4) similar to the one showed in 3.3 but with a modification: the input simulated does not come from the Simulink scheme but is generated inside the Arduino.

Through this simple code it is possible to analyze the limits of the movement of the mechanical arm; in particular, as showed in fig. 4.5 we tried to find the nearest and the furthest position reachable by the arm in order to have an idea on how to dimension the chess board.

The pmin is at a distance of about 17cm while pmax is at a distance of about 27cm from the center of the mechanical arm. In addition in order to control the precision of the arm, it was checked how many equidistant positions could be reached and which was such distance; we discovered the possibility of reaching ten different position each of them spaced of about 2cm each other.

Consequently we chose to realize a chess board with 16cm length for side and with cells of 2cm dimension, for the 8x8 chessboard case, while for the 6x6 chessboard

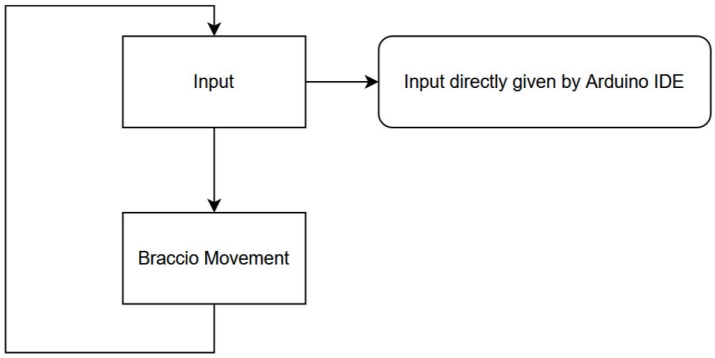


Figure 4.4: Flaw chart of the code used for Limits of mechanical arm

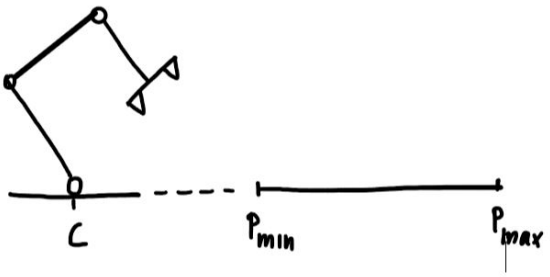


Figure 4.5: Pmin and Pmax relative to the mechanical arm

the length was 16,2cm and the cell had dimension 2.7cm.

4.5 Precision problem

The chess board and the mechanical arm are both fixed in position with respect to the reference system.

If we want to reach all the cells of the chess board it is necessary that all the portions of the mechanical arm are able to move with sufficient precision. As shown in the above paragraph the mechanical arm is able to reach eight equidistant cells immediately in front of it. The problem show up when it is required to reach one of those cell on the side of the chess board, a movement that require the rotation of the mechanical arm basis. Analyzing with a simple code the precision of each motor of each junction of the mechanical arm we discovered that they have a precision of

about five degree.

If for the motors of forearm, wrist and arm of the mechanical arm this is not such a great problem because combining their movement it is possible to overcome the small precision, for what concern the motor of the basis, the precision of five degree does not allow to reach all the cells. In the following paragraph we will show the solution we used for solve it and other possible solution that could have been implemented.

4.6 Possible solution to the precision problem

If we want to reach the lateral cells of the chess board the five degree precision of the motor of the basis is not sufficient: it is needed to make some modification on the mechanical arm. In the following some possible solution and their pro and cons

Solution 1: Polar chess board

As shown in fig. 4.6

- PRO: the solution does not require any supplementary hardware because we could use the actual abilities of the mechanical arm; consequently the costs would be low.
- CONS: the chess board would not be a classical chessboard.

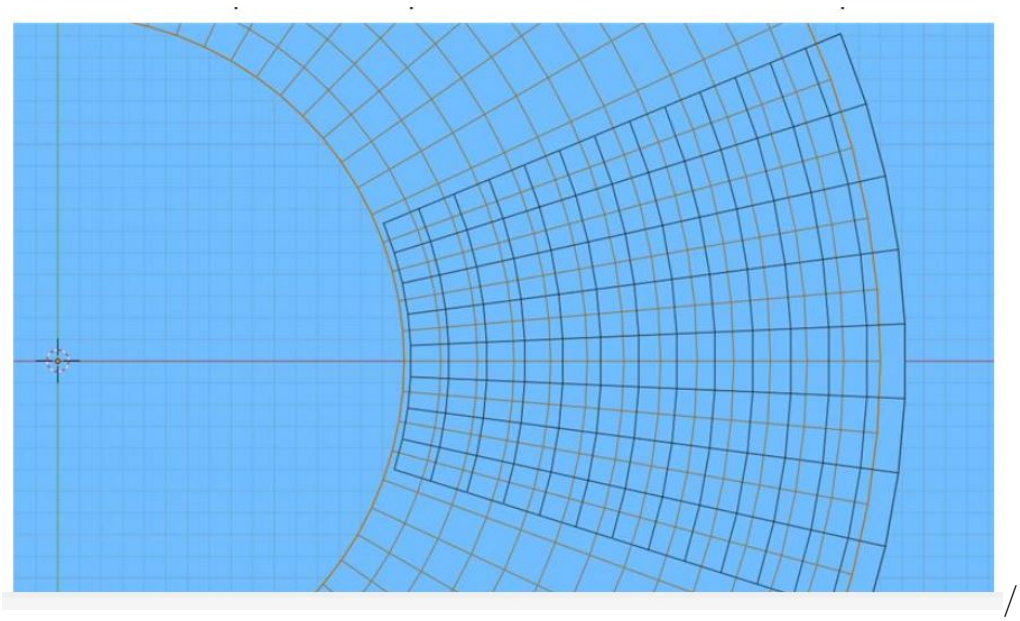


Figure 4.6: Exemple of a polar chessboard.

Solution 2: put wheel on the mechanical arm

As shown in fig. 4.7

- PRO: it would allow us to do not perform any more the rotational movement of the basis of the mechanical arm.
- CONS: it would require considerable modification on the hardware side: in addition to the mounting of the wheel it would requires a little motor for propulsion and a brake system; it may be required an additional controller for controlling both the propulsion motor and the brake system. Because of the wheel, equilibrium problem may show up because the basis may not be so stable on the wheel; for this reason the cost of this kind of solution may not be so low. It may be also required some sort of feedback system.

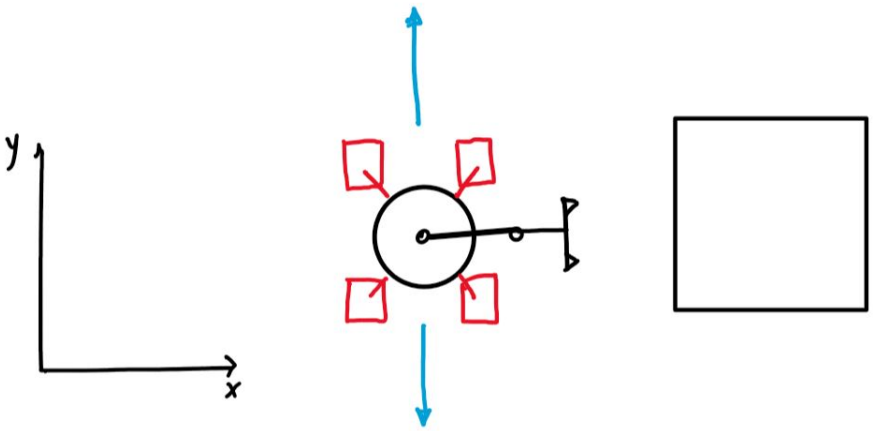


Figure 4.7: Solution with the wheel.

Solution 3: mobile plan

As shown in fig. 4.8

- PRO: it would be a immediate solution, similar to solution 2 but with the advantage of a possible greater precision: it would require an additional motor that could be controlled with the same controller used by the mechanical arm. The movement of the chess board (or similarly of the basis of mechanical arm) could be performed by the additional motor and a screw-nut screw mechanism.
- CONS: this solution require some hardware modification and additional element, but despite this it continue to be quite cheap solution; it may not be required a feedback system.

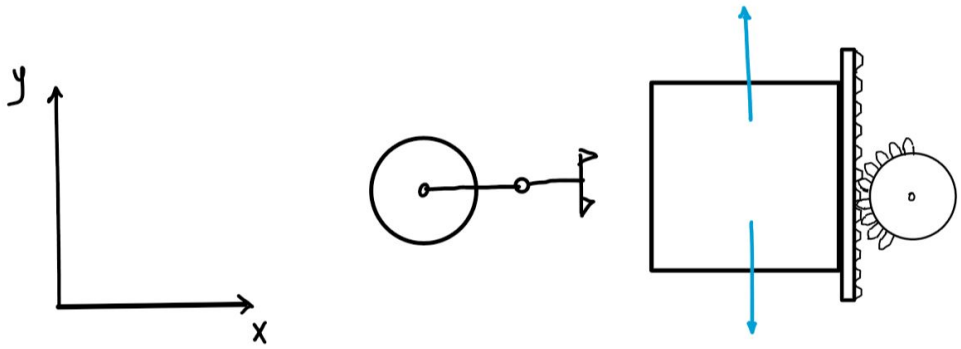


Figure 4.8: Mobile chessboard solution

Solution 4: changing of the inclination of the plane on which the mechanical arm is posed

As shown in fig. 4.9

- PRO: there is no hardware modification to introduce except the introduction of a inclined plane.
- CONS: the chess board would be seen as inclined in the space and no more as horizontal: consequently position on the chess board would not be reached giving as input polar coordinate (in other word an angle and a ray length). Consequently it is no more sure that the precision we have discussed about in the previous paragraph that allows to reach 8 equidistant cell is sufficient in this case; at the same time the movement would be more complex and less natural to be seen.

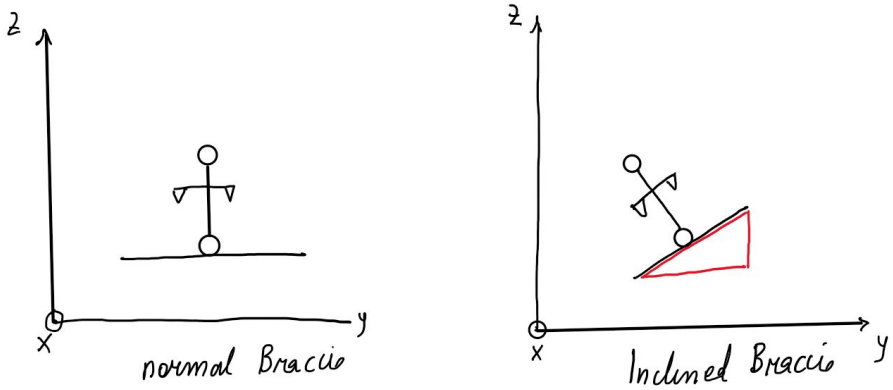


Figure 4.9: image of inclined chess board.

Solution 5: move the chessboard using the mechanical arm

- PRO: it does not require any hardware modifications.
- CONS: this solution may not be sufficiently precise because there is no feedback for controlling effectively the position reached by the chess board; at the same time this has not to be too heavy because the element that moves it is the arm itself. The precision may be improved with the introduction of sensors but this will complicate the codes for elaborating also the info coming from the sensors; other solution proposed had not this problem.

Soluzione 6: change the motor of the basis

- PRO: changing the base motor would be a quite cheap operation and the new motor could continue to be controlled by the Arduino controller.
- CONS: the motor we have seen have different dimension from the one coming with the Tinkerkit; for this reason it would be necessary to modify a little bit the basis of the mechanical arm.

At the end we have chosen the 6th solution because it seemed to be the most elegant and quite cheap since only an additional motor is required.

4.7 Modification to the mechanical arm part1

For solving the precision problem, among the various solution proposed it was chosen the 6th solution because it seemed to be the most elegant and quite economic due to the fact that only an additional motor is required. The movement in this way is similar to the one done by a human arm when he play chess. The motor that was chosen for that purpose is a stepper motor: the "Longruner TB6600 Nema 17" shown in fig. 4.10.

The dimension of that stepper motor is different from the Tinkerkit base motor; for this reason modification as shown in fig. 4.11 were made in order to use the new motor.

The final realization can be seen in fig. 4.12.

It is a simple system of gear with transmission ratio 1:2; the number of teeth of the bigger gear is 60 while the number of teeth of the smaller one is 30. The design of these gears was made by using "DesignSparkMechanical" and the models obtained are shown in fig. 4.13.

Thanks to this system the precision of the rotational movement of the mechanical arm was increased: we can now reach a precision of rotation of about 1 degree.

Figure 4.10: Longruner TB6600 Nema 17. Image taken from `https://www.amazon.it/stepper-Longruner-passo-passo-segmenti-stampante/dp/B07FKJK1H9/ref=asc_df_B07FKJK1H9/?tag=googshopit-21&linkCode=df0&hvadid=343286373758&hvpos=&hvnetw=g&hvrand=8691443076650736806&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=20543&hvtargid=pla-562980456200&th=1`

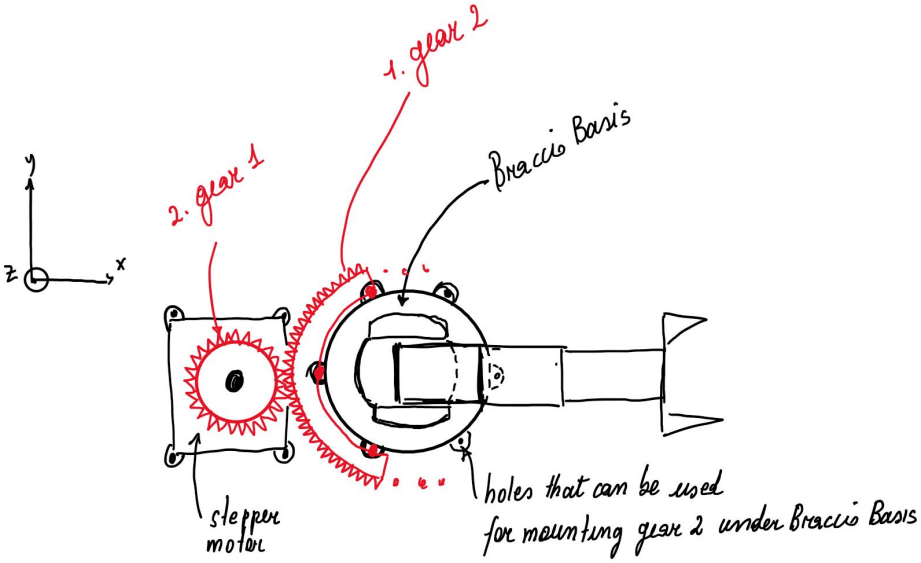


Figure 4.11: Draft of the modification done to the basis of the mechanical arm.

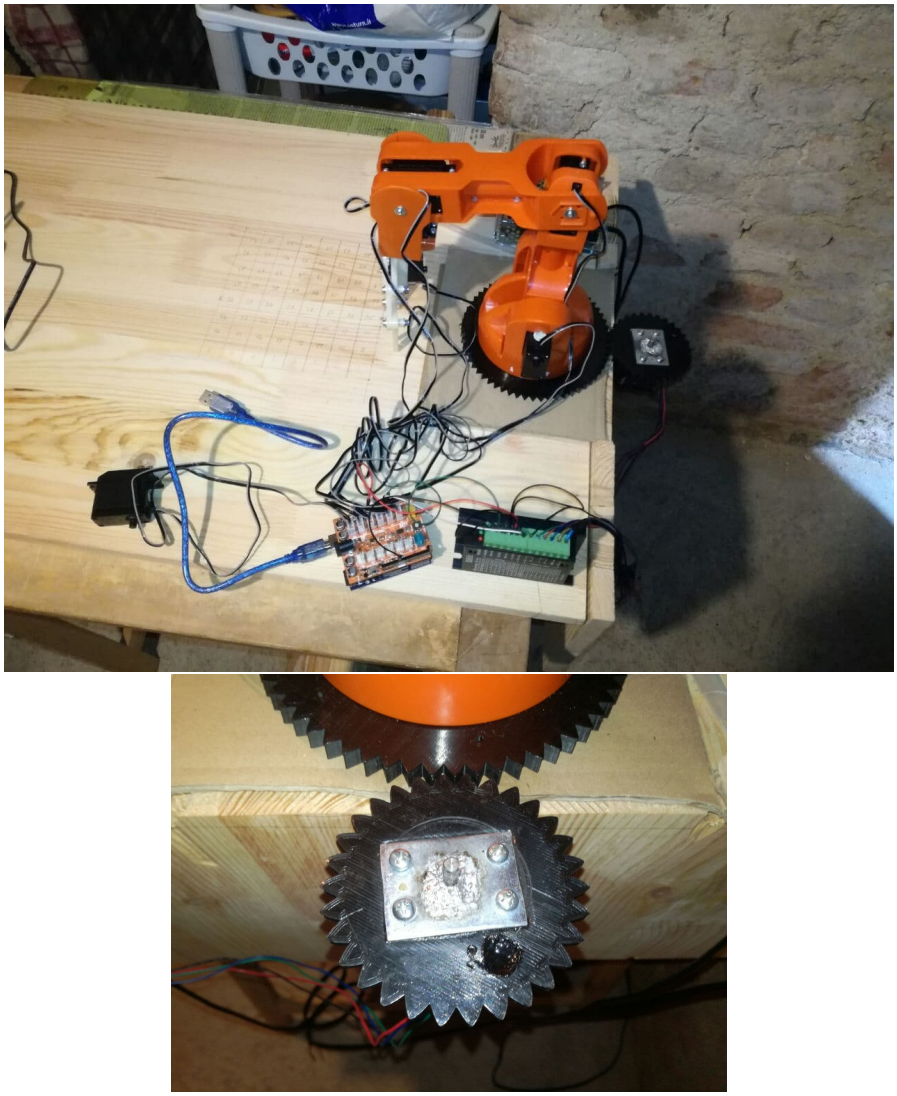


Figure 4.12: Modification done to the basis

4.8 Modification to the mechanical arm part2

The weight of the arm, when the mechanical arm is stretched as shown in fig. 4.14, generate a torque that would overturn the mechanical arm around point O. In order to avoid this, it was decided to pivot the bases over a wood surface as shown in fig. 4.15.

For avoiding the mechanical arm to slip over the pivot we tried to increase the friction putting cardboard and pins: in fig. 4.16 it is shown schematically

After all the modification we have discussed, the mechanical arm has sufficient

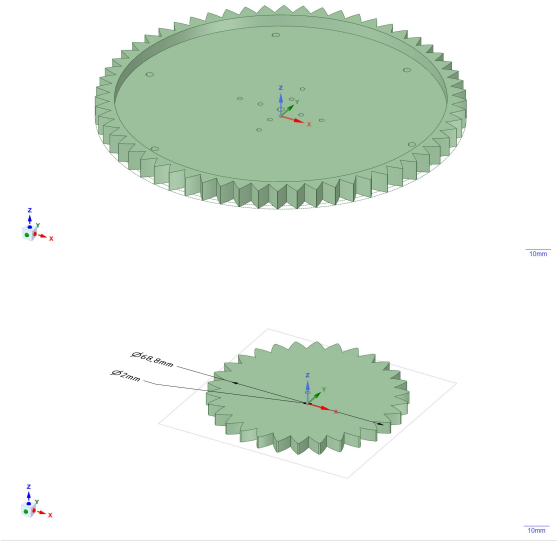


Figure 4.13: The two gears used with the new stepper motor.

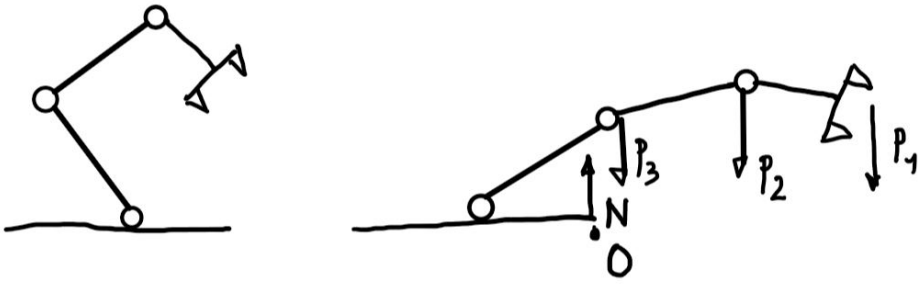


Figure 4.14: Equilibrium problem when the mechanical arm is completely stretched.

precision to try to use it for playing chess, but in a 6x6 chessboard.

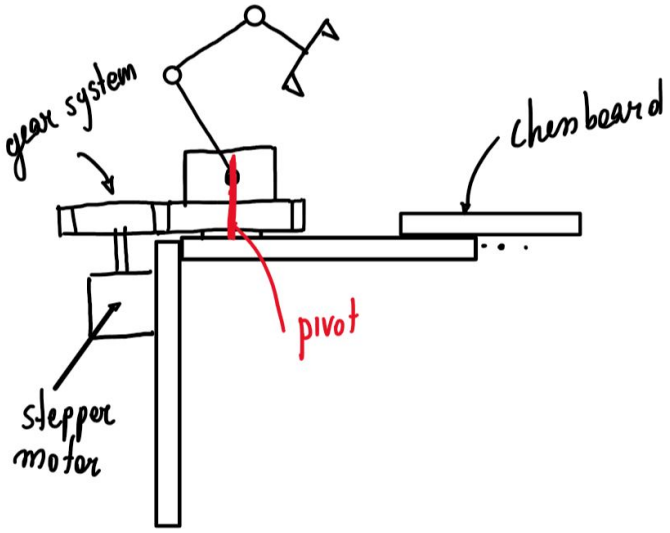


Figure 4.15: System for avoiding equilibrium problem thank to a pivot

Figure 4.16: Pin for avoiding vertical slip of the mechanical arm on the pivot.

Chapter 5

Connection with BCI

In this chapter we will discuss about how the connection between the Arduino controller that control the mechanical arm and the BCI part is created: in particular, we will focus on the Simulink model and their different versions used for this purpose.

5.1 Elaboration of data

In the Simulink scheme that is shown in fig. 5.1, we elaborate the input (in the scheme it is a constant Simulink block but in the final model the input will come from the BCI, as shown in fig. 5.6) so that we will send through the serial port, at which the Arduino board is connected, a sequence of angles that will allow the arm movement.

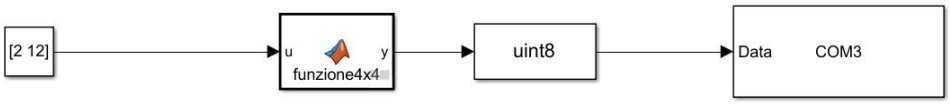


Figure 5.1: The block that from input (2 cell) give sequence of angles that pass through the serial port to Arduino controller

The main element that allows the elaboration of data is the block fcn (funzione4x4): it acquires the info about the chess cell from where we want to take the pawn and the cell where we want to put it, and converts it into a set of angles, that are then passed to the Arduino through the serial port block. Two different version were made for the function fcn and in the following paragraphs we will show them.

Version1 fcn: inverse kinematic

The flow chart of how the data are elaborated inside the fcn block is in fig. 5.2. About the point 2 of the flow chart: knowing the dimension and the numeration of

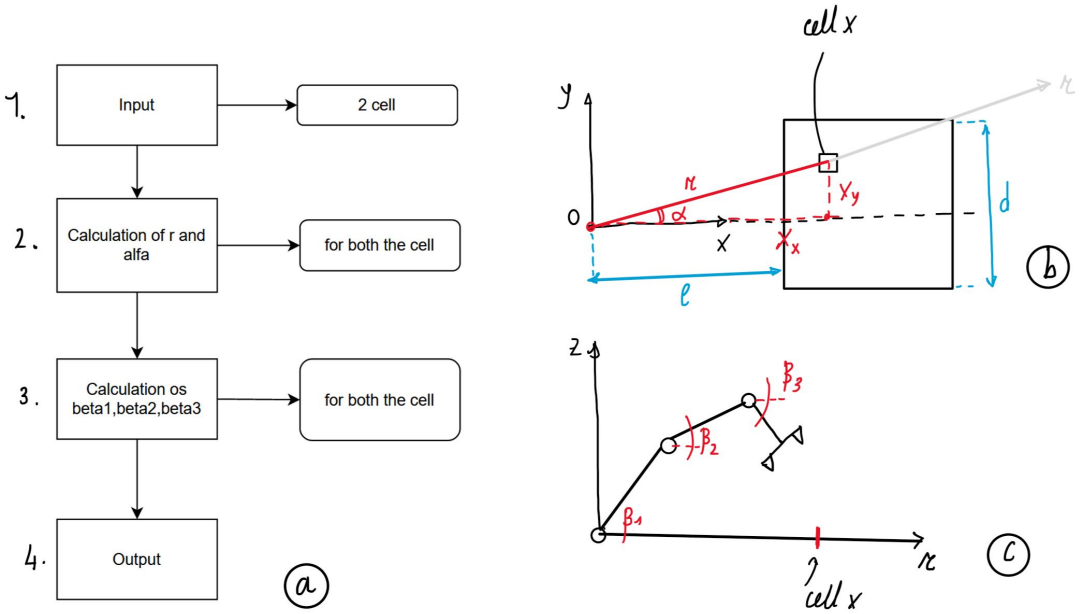


Figure 5.2: structure of version 1 of fcn

the cells, the dimensions of the chessboard, and the distance between the mechanical arm and the chessboard, we can calculate r and alpha through trigonometry. Indeed, given the cell number and knowing that there are 8 rows and the fact that for each of them there are 8 cells, it is possible to calculate Xx and Xy through a switch function; thanks to this, it is immediate to calculate alpha and r. Alpha will be the angle at which we have to make the motor of the base rotate while the information on r will be used for a calculating the set of angles beta. In fact about the point 3 of the flow chart: knowing r, the set of angles beta is obtained by solving a simple 2d problem: the Inverse Kinematic problem for a planar manipulator with 3 cylindrical joint whose solution is shown in section 2.4.1: this is the part that will change in version 2 of fcn. This type of calculation requires the ability of each joint to assume precisely the set of angle calculated above; this is not the case of the mechanical arm we are using because as we said in previous chapter the precision we have is about 5 degree, that is not sufficient. Thus, using this trigonometrical method we are not able to reach precisely the point we are pointing to. Due to that, a less "automatic" way was tried that will be shown in the following part.

Version2 fcn

In the flow chart in fig. 5.3 another procedure for obtaining the set of angle betas necessary to move the mechanical arm is shown.

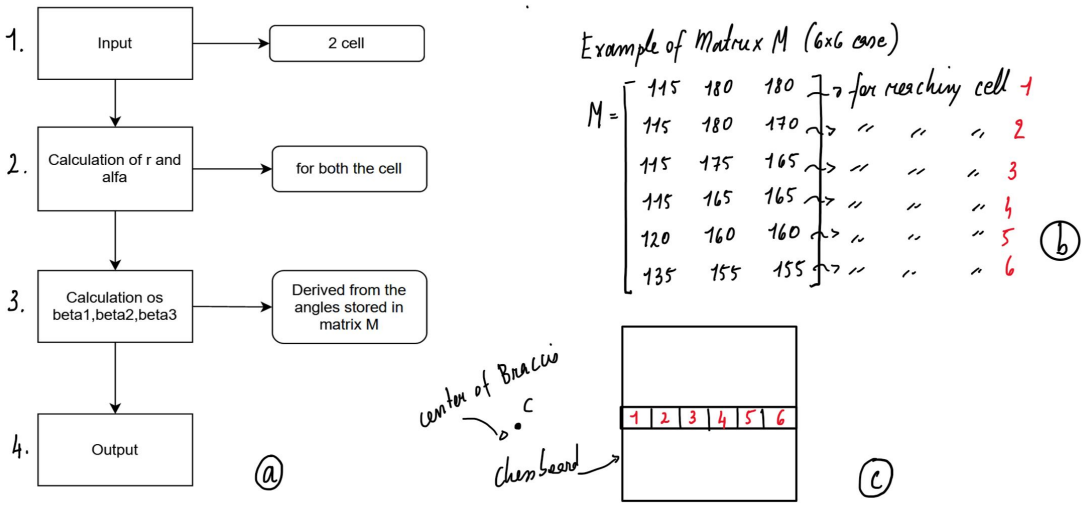


Figure 5.3: Version 2 of fcn and example of matrix M in 6x6 case

Once that alpha has been calculated exactly at the same way as in version 1, the data are used for the calculation of set beta angles; the alpha angle will be used for making the motor of the base rotate. The info on r is no more used while, instead of that, the number of the cell is used for determining in which row that cell is situated. Using that data and referring to fig. 5.3 b, it is possible to access a matrix M that will give us the set of beta angles required for reaching the considered cell. The matrix M is created measuring empirically the angles needed for reaching the cells immediately in front of the mechanical arm. Each row of the Matrix corresponds to the beta angles set needed to reach the cell at the corresponding row of the chessboard. In this way, knowing the column at which the selected cell is located, and knowing its row. it is possible to select the row of the matrix M necessary to reach the cell. This method allow us to deal with simplicity with the lack of precision of the mechanical arm; also in this way, not all the cell are efficiently reached. For this reason, in order to avoid to waste all the modification done, it was decided to try with a 6x6 chess board or even a 4x4 instead of a 8x8; In fig. 5.4 we report the M matrix for these cases

```
                        a=[115,180,180];
                        b=[115,180,170];
a=[115,180,180];        c=[115,175,165];
b=[120,165,175];        d=[115,165,165];
c=[125,165,165];        e=[120,160,160];
d=[130,155,160];        f=[135,155,155];
```

Figure 5.4: raw of matrix for 4x4 case and 6x6

5.2 Complete pipeline

We will show the scheme that we used for placing electrodes on the scalp of the user during both the training and the testing phase. The SSVEP can be measured using EEG and there are a lot of papers in literature that show how to position electrodes in the occipital area, as for example [13–20] where the most common positions are in Oz, O1, O2, POz, and Pz referring to fig. 5.5.

We will now analyze the complete pipeline that allows the data acquisition from a EEG and give them to the function we have seen above (shown in fig. 5.6).

The main element in this Simulink scheme are:

- block that set the communication with the EEG;
- the Classifier.

The Classifier is the nucleus of the BCI: starting from the EEG data, we extract features that are then provided to a classifier whose role is to assign a label to the signal acquired. The most commonly used classifiers are Support Vector Machines (SVMs) and Linear Discriminant Analysis (LDA) that, as written in [2, 21], show the best performances in this context. In this thesis a SVM was used. A one-vs-All setting is implemented so that a separate classifier for each class is trained in order to discriminate whether a sample is part or not of a specific class, as written in [2].

Data coming from the EEG are potentials of different channels. Generally, there is a time interval in which the data acquired are used for a control purpose and an equal interval of time of rest (as in fig. 5.7), during which the system continues to acquire data without using them. In that interval, in fact, the user can rest from seeing the flickering lights. Each measurement interval is called trial and the sum of resting interval and measurement interval is handled through the clock signal block. The duration of the interval is a parameter that can be modified if necessary.

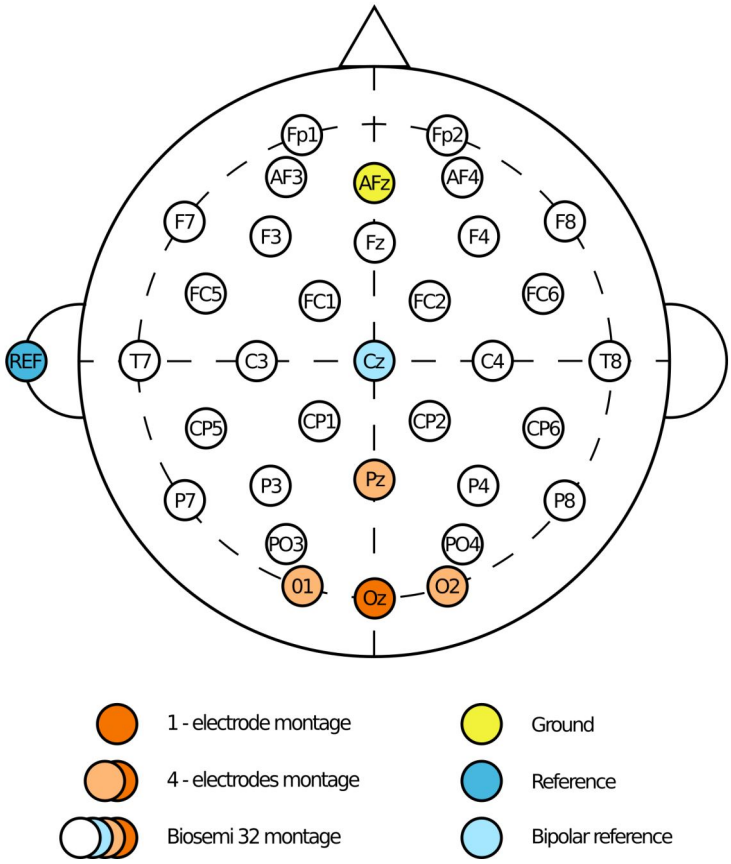


Figure 5.5: Electrodes position on the scalp for EEG. Image taken from [2]

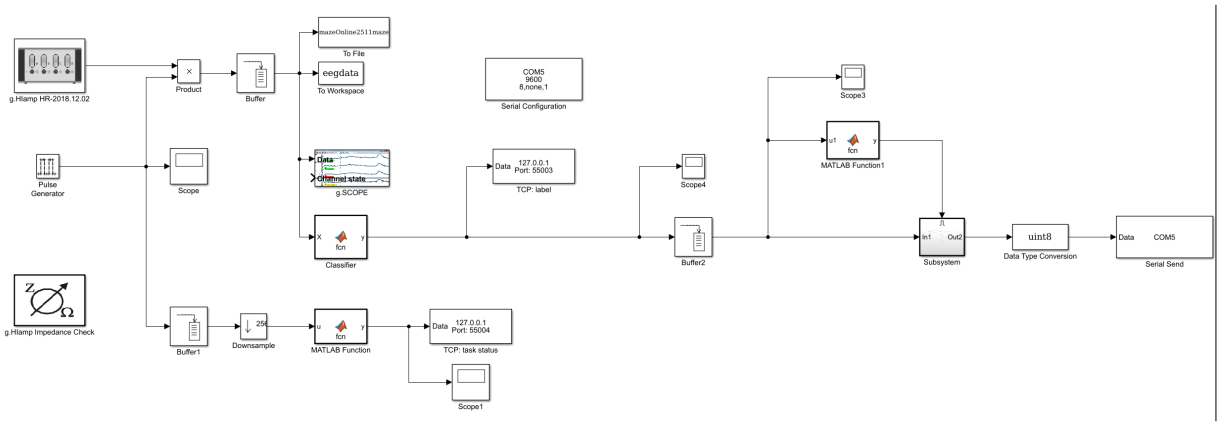


Figure 5.6: The complete Pipeline

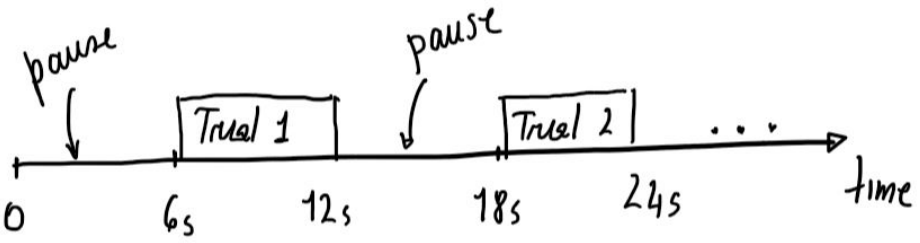


Figure 5.7: Sequence of trial and pause

The data coming from the trial are stored, bufferized and sent to the Classifier. This last one classifies these data according to a model collected in a file that is the result of a training phase that we will describe in the next paragraph.

5.3 Training phase

The Classifier needs a training phase in order to classify in real time the data coming from the EEG; the model shown in fig. 5.8 is used.

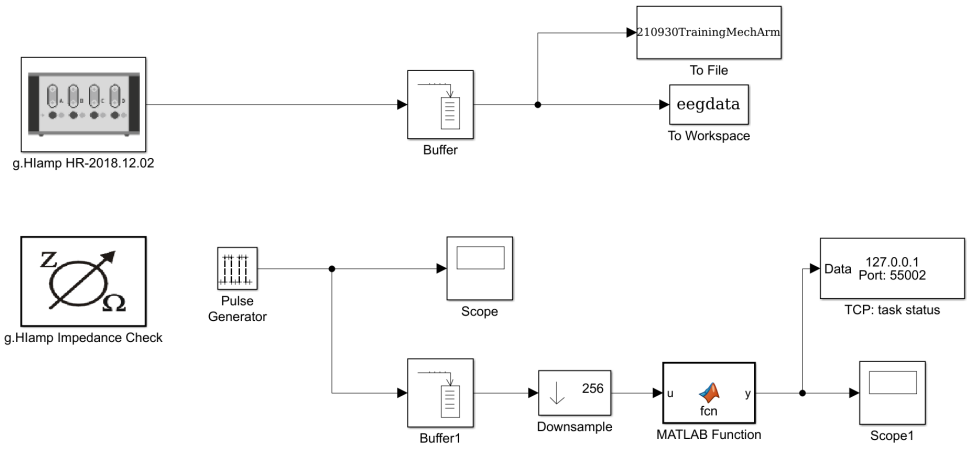


Figure 5.8: Pipeline for the training

Using a sequence of trials in which we give to the user a predetermined light to look at, we collect data for which we know exactly what the user was looking at. Each set of data can be subdivided according to particular characteristics (features). There are several techniques for features extraction in the literature as written in [22]. The characteristic we want to highlight is the PSD (Power Spectral

Density) as in [23]. In the training, and later, in the testing, we computed the power spectrum of the signal coming from the EEG through the periodogram and through the Welch's method. In Welch's method the signal is firstly subdivided in a series of overlapping portions of equal length that are then windowed for the computation of the periodogram of each portion. Finally, the results obtained are averaged.

We link to each flickering light a certain class. Consequently, for each trial, the brain signals we acquired have particular characteristics that will be used to "recognize" the class. During the training, we collect the trials that belong to a certain class and we figure out the main characteristics that make these signals similar to each other, so that we are able to divide signals with these characteristics (that consequently indicate the class) with respect to signals that have not these characteristics (that do not represent the class). This is done for each class so that, at the end of the training, a model containing these information is created and we are able to classify unknown signals according to their characteristics and on how much these characteristics are near to the ones of a certain class rather than the characteristics of another one.

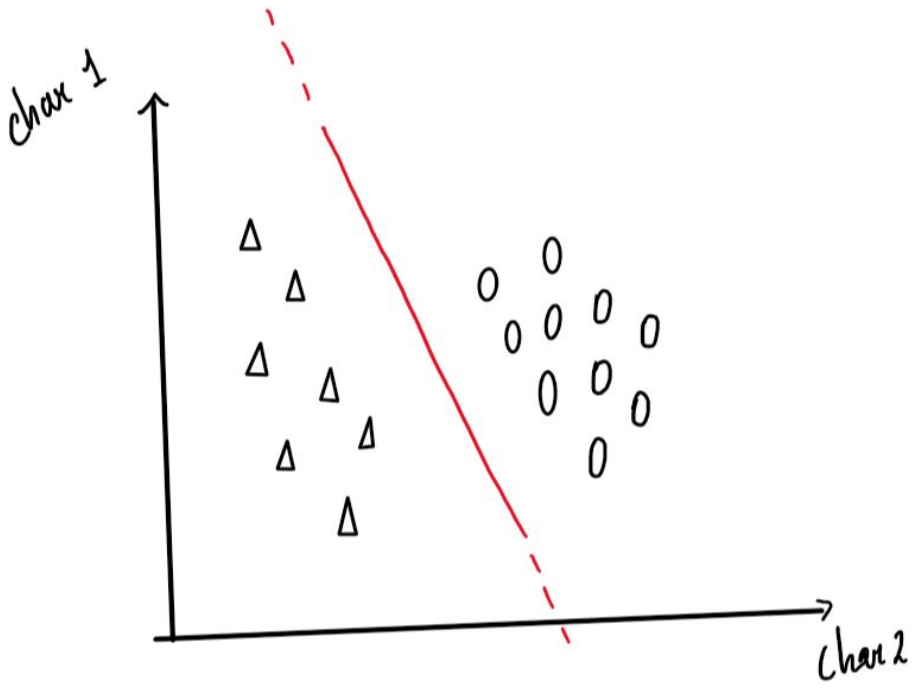


Figure 5.9: Example of 2 class objects differentiated by 2 characteristic.

In fig. 5.9 it is shown an example (not related to the real training) with data of

two types of class (triangle and circle) placed in a Cartesian plane according to two characteristics, char1 and char2. It is possible to divide the two classes with a red line in relation to the characteristics of the data. Once the line is created, it is possible to classify the new unknown data.The same is done with the EEG data although increasing the number of dimensions and classes.After a training session where a model is created, according to this model the new data are classified.

In the following chapter we will describe more in detail the data acquired and used for the training of the classification.

Chapter 6

Results

In this Chapter we will focus our attention on the results obtained both in training and testing (online) phase.

6.1 Results of the Training

In this section we will discuss about the results obtained after the training sessions.

We created a predetermined sequence of flickering light and we made the user watch it. The user needs to gaze at a flickering light for a certain period and after that he will have the same amount of time as rest. Then these steps are repeated as many times as the length of the predetermined sequence. After that we use the data acquired during the active time for the training of the classifier. We began trying to train the classifier with six different classes, keeping in mind that each class corresponds to the typology of EEG signal in response to, in this case, each one of the 6 flickering lights at different frequency. We soon discovered that it was too ambitious (at least in the beginning), because the signals acquired were not sufficient to provide a satisfying classification, and the results of training were not enough reliable. We have in fact to remember that also the user has to be trained to look at the flickering light and focus only on it. We tried consequently to reduce the number of classes from six to three, and we increased that number in a second moment, once the subject was feeling confident about his ability to distinguish among the flickering patterns. We also tried a different set of frequencies at which make the lights flicker and similarly we acquired data from different users (User1 and User2) in order to select the ones that guarantee a more reliable performance; for what concern the data acquired from User1 we have to say that we used only one electrode in the Oz as in [24–27] while in the User2 case we use electrodes in Oz, O1, O2, and Pz as in [3]. In the following we will analyze the data we acquired in all these different cases.

3 classes with frequency 7Hz,11Hz,13Hz

As shown in fig. 6.1, where the first image refers to the Unity interface, the second to the data coming from User1 and the third to the data coming from User2, we can see that with three classes, therefore using images flickering respectively at 7Hz, 11Hz and 13 Hz, there are peaks centered at the corresponding frequencies for both the users.

4 classes with frequency 13Hz,7Hz,5Hz,17Hz

As shown in fig. 6.2, where the first image refers to the Unity interface, the second to the data coming from User1 and the third to the data coming from User2, we can see that with four classes at frequency 5Hz, 7Hz, 11Hz (substituted with 17Hz with User2), 13Hz there are peak value at the corresponding frequencies in both the user.

6 classes with frequency 7Hz, 13Hz, 5Hz, 17Hz, 9Hz ,11Hz

As shown in fig. 6.3, where the first image refers to the Unity interface, the second to the data coming from User1 and the third to the data coming from User2, we can see that with six classes at frequency 7Hz, 13Hz, 5Hz, 17Hz (substituted with 12Hz with User1), 9Hz ,11Hz there are peak value at the corresponding frequencies in both the user.

6.1.1 Results of training

At the moment we can see that the best frequency to use are 7Hz, 13Hz, 5Hz, 17Hz, 9Hz ,11Hz. The data coming from User2 guarantee better performances and permit to reach six different classes: for this reason User2 will be in the next pages the user who will perform the task with the robot. The Data coming from User1 show smaller peak value and at the same time an increasing trend at high frequency that was not optimal for the classification, shadowing the harmonic peak of the frequencies. After the training, a model is produced so that it can be used for the classification of data acquired in real time in order to control the mechanical arm and make it move; this will be shown as result in the following section where a detailed sequence of passage done for controlling the mechanical arm will be described

6.2 Result of the Online phase

In this section we will show the control of the mechanical arm, in order to perform some particular tasks in real time. This was possible thanks to a previous phase of

training done for making the classifier able to classify the data coming from the EEG.

6.2.1 Sequence

The user is set in front of a screen and has to select in freedom a flickering light at which focus his attention. The user will select at first the column of the cell from where he wants to take the pawn. The data acquired in the trials of 6s are buffered and passed to the classifier that recognize them and gives as output the classes. After that, there is a pause of 6s, followed by the selection of the row of the cell. In this way, the cell from were the robotic arm needs to take the pawn will be chosen. Afterwards, there is a pause of 6s followed by 6s in which the cell selected is shown in the screen and another 6s of pause after which there is the selection of the cell in which to put the pawl: the sequence of passages is the same seen in the lines before.

The output of the classifier is passed to the function fcn we have seen in chapter 5 that generates the set of angles necessary to reach the two different cells. After the two cells (starting and final point) were determined the mechanical arm performs the movement, taking the pawn from and to the positions chosen.

While the mechanical arm is moving, the selection of the two cells is repeated in loop.

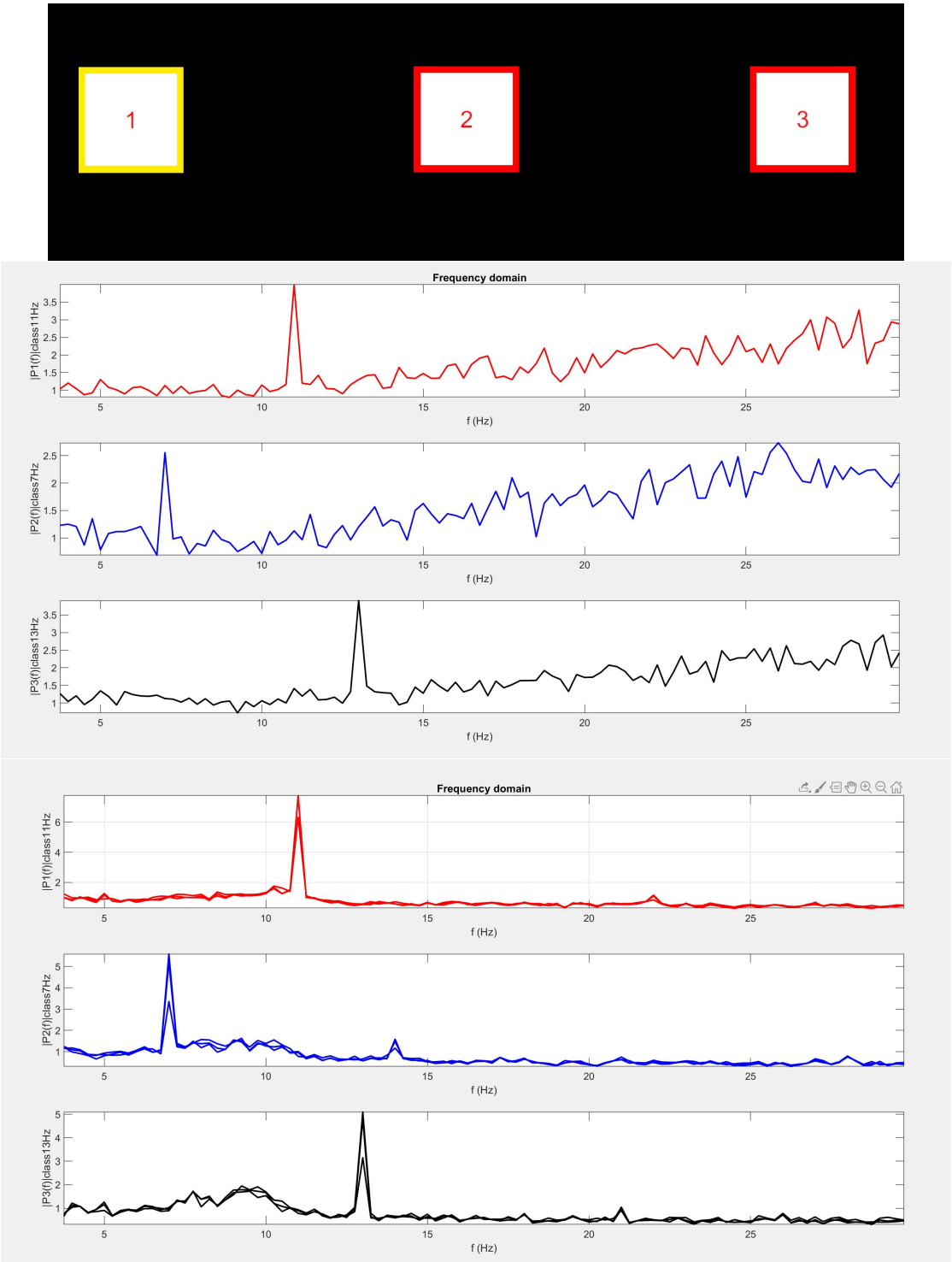


Figure 6.1: At the top What the user see in the training for 3x3 training; In the middle Data from user User1; At bottom Data from user User2

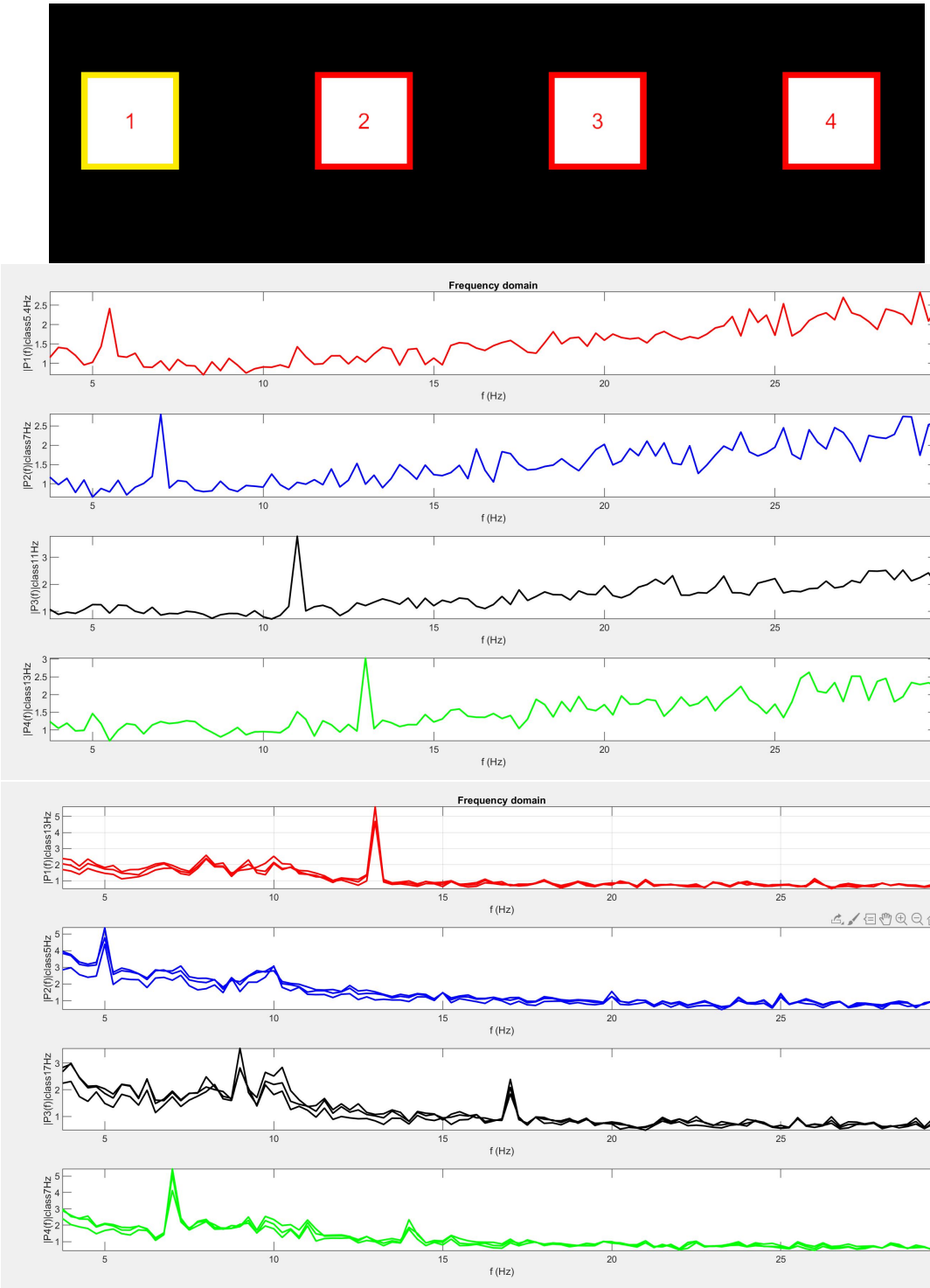


Figure 6.2: At the top What the user see in the training for 3x3 training; In the middle Data from user User1; At bottom Data from user User2

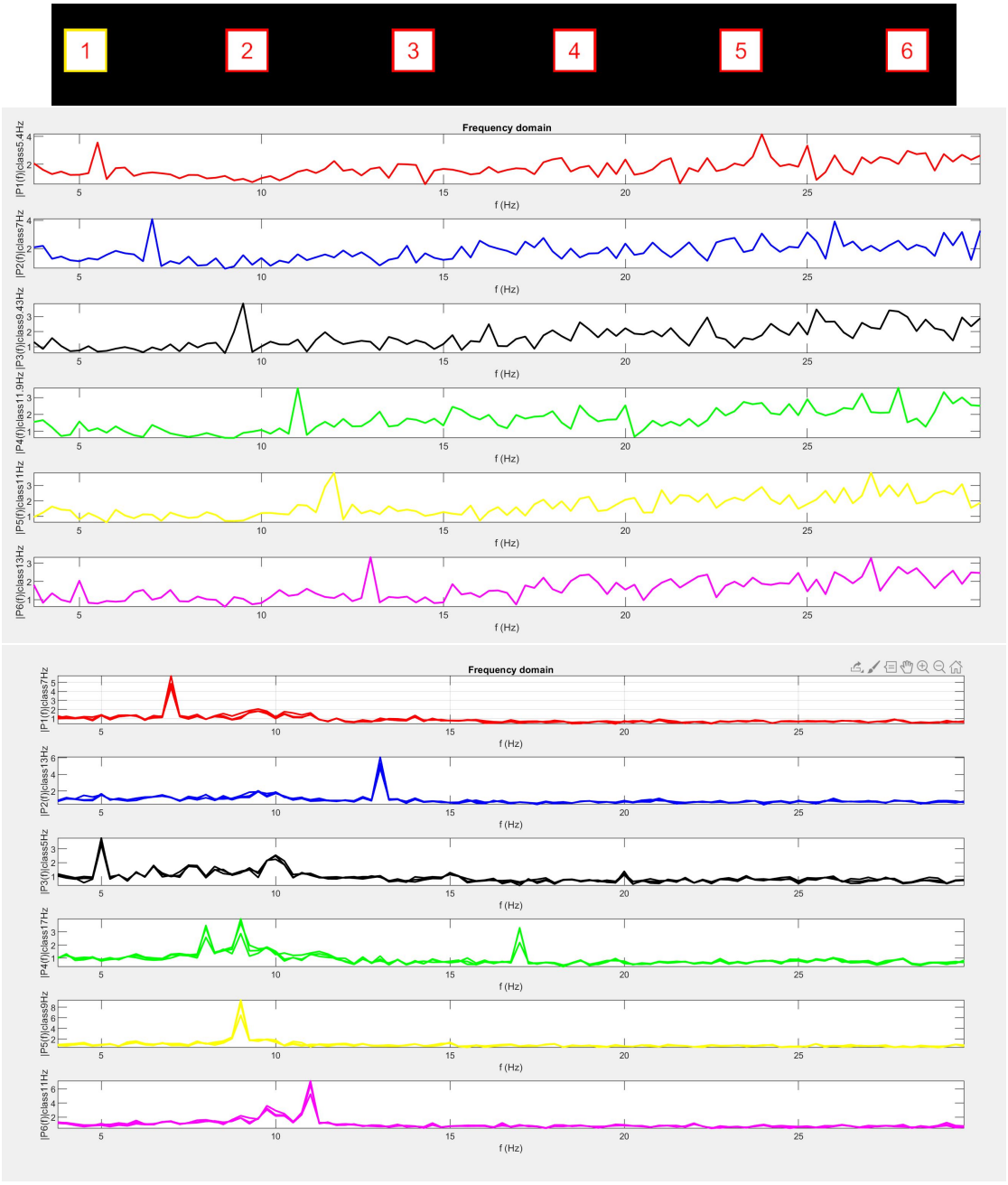


Figure 6.3: At the top What the user see in the training for 3x3 training; In the middle Data from user User1; At bottom Data from user User2

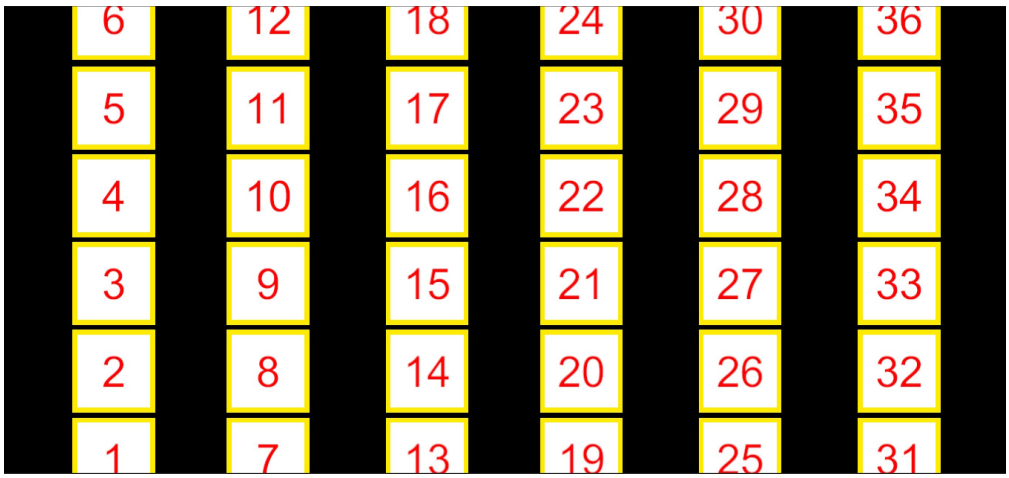


Figure 6.4: The Unity real-time interface in 6x6 case

Figure 6.5: The complete system with Unity, the mechanical arm and the user

Chapter 7

Conclusions

As we can see from the results of the previous chapter, we were able to connect the BCI with the mechanical arm through the usage of the EEG technique, Simulink and an Arduino controller and to perform some simple tasks like moving an object from one place to another one.

We were not able yet to perform completely the task of making the arm play the chess game (in a 8x8 chessboard) essentially due to two main issues:

- the number of classes that the classifier was able to recognize after training phase was about 6 and not 8 as the game requires. However, with six classes we reached an accuracy of about 90%, a quite good result;
- the precision of the movement of the mechanical arm did not allow the usage of a 8x8 chess board. In addition, the gripper used to pick and place the pawn does not allow a good positioning of the pawn. Due to this we had to refold to a 6x6 chessboard and to a task that is simpler than playing the real chess game.

The latter issue could have been avoided if we could have used a different mechanical arm and/or a different solution to increase the radial resolution of the movement, but considering the embryonic stage of the framework development we preferred to go for a simple and economic solution.

We expect in future to overcome the criticism shown above. In fact, for what concern the number of classes, to have a User better trained to work with eight Classes could solve the issue. To solve the precision problem it can be considered to substitute the robotic arm used and/or to use a magnetic gripper. In addition, further improvements can be made to simplify the sequence for the cell selection.

In conclusion we could surely state that we obtained exciting results, since we were able to successfully perform the task with the 6x6 chessboard, using the only our brain and of our eyes to move a robotic arm. Future development of this

technology could lead to more relevant application as could be the control of a robotic prosthesis by injured patients.

Bibliography

[1] Paolo Ricci. «Empowering human-computer interactions with advanced brain imaging and real-time virtual reality interfaces». In: (2020) (cit. on pp. 3, 6).

[2] Arturo Micheli. «Empowering human-computer interactions with advanced brain imaging and real-time virtual reality interfaces». In: (2021) (cit. on pp. 3–5, 14, 20, 40, 41).

[3] Alexander Maye, Dan Zhang, and Andreas K. Engel. «Utilizing Retinotopic Mapping for a Multi-Target SSVEP BCI With a Single Flicker Frequency». en. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.7 (July 2017), pp. 1026–1036. ISSN: 1534-4320, 1558-0210. DOI: 10.1109/TNSRE.2017.2666479. URL: http://ieeexplore.ieee.org/document/7911330/ (visited on 12/02/2020) (cit. on pp. 5, 45).

[4] G Pfurtscheller. «Induced oscillations in the alpha band: functional meaning». In: *Epilepsia* 44 (2003), pp. 2–8 (cit. on p. 6).

[5] Luca Mesin. «Introduction to biomedical signal processing». In: 1 (2017) (cit. on p. 6).

[6] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, and Bruno Arnaldi. «A review of classification algorithms for EEG-based brain–computer interfaces». In: *Journal of neural engineering* 4.2 (2007), R1 (cit. on p. 11).

[7] Christopher JC Burges. «A tutorial on support vector machines for pattern recognition». In: *Data mining and knowledge discovery* 2.2 (1998), pp. 121–167 (cit. on p. 11).

[8] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on p. 11).

[9] Vikramaditya Jakkula. «Tutorial on support vector machine (svm)». In: *School of EECS, Washington State University* 37 (2006) (cit. on p. 11).

[10] Fabien Lotte. «Study of electroencephalographic signal processing and classification techniques towards the use of brain-computer interfaces in virtual reality applications». PhD thesis. INSA de Rennes, 2008 (cit. on p. 11).

[11] Theodoros Evgeniou and Massimiliano Pontil. «Support vector machines: Theory and applications». In: *Advanced Course on Artificial Intelligence*. Springer. 1999, pp. 249–257 (cit. on p. 11).

[12] Kristin P Bennett and Colin Campbell. «Support vector machines: hype or hallelujah?» In: *ACM SIGKDD explorations newsletter* 2.2 (2000), pp. 1–13 (cit. on p. 11).

[13] Ivan Volosyak, Diana Valbuena, Thorsten Lüth, Tatsiana Malechka, and Axel Gräser. «BCI Demographics II: How Many (and What Kinds of) People Can Use a High-Frequency SSVEP BCI?» en. In: *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING* 19.3 (2011), p. 8 (cit. on p. 40).

[14] Brendan Z. Allison, Jing Jin, Yu Zhang, and Xingyu Wang. «A four-choice hybrid P300/SSVEP BCI for improved accuracy». en. In: *Brain-Computer Interfaces* 1.1 (Jan. 2014), pp. 17–26. ISSN: 2326-263X, 2326-2621. DOI: 10.1080/2326263X.2013.869003. URL: http://www.tandfonline.com/doi/abs/10.1080/2326263X.2013.869003 (visited on 12/02/2020) (cit. on p. 40).

[15] Pablo Martinez, Hovagim Bakardjian, and Andrzej Cichocki. «Fully Online Multicommand Brain-Computer Interface with Visual Neurofeedback Using SSVEP Paradigm». en. In: *Computational Intelligence and Neuroscience* 2007 (2007), pp. 1–9. ISSN: 1687-5265, 1687-5273. DOI: 10.1155/2007/94561. URL: http://www.hindawi.com/journals/cin/2007/094561/abs/ (visited on 12/02/2020) (cit. on p. 40).

[16] Christoph Guger, Brendan Z. Allison, Bernhard Großwindhager, Robert Prückl, Christoph Hintermüller, Christoph Kapeller, Markus Bruckner, Gunther Krausz, and Günter Edlinger. «How Many People Could Use an SSVEP BCI?» en. In: *Frontiers in Neuroscience* 6 (2012). ISSN: 1662-4548. DOI: 10.3389/fnins.2012.00169. URL: http://journal.frontiersin.org/article/10.3389/fnins.2012.00169/abstract (visited on 12/02/2020) (cit. on p. 40).

[17] Nikolay V Manyakov, Nikolay Chumerin, and Marc M Van Hulle. «Multichannel Decoding for Phase-Coded Ssvep Brain–Computer Interface». en. In: (2012), p. 8 (cit. on p. 40).

[18] Yangsong Zhang. «Multivariate synchronization index for frequency recognition of SSVEP-based brain–computer interface». en. In: *Journal of Neuroscience Methods* (2014), p. 9 (cit. on p. 40).

[19] Petar Horki, Teodoro Solis-Escalante, Christa Neuper, and Gernot Müller-Putz. «Combined motor imagery and SSVEP based BCI control of a 2 DoF artificial upper limb». en. In: *Medical & Biological Engineering & Computing* 49.5 (May 2011), pp. 567–577. ISSN: 0140-0118, 1741-0444. DOI: 10.1007/s11517-011-0750-2. URL: http://link.springer.com/10.1007/s11517-011-0750-2 (visited on 12/02/2020) (cit. on p. 40).

[20] Zhihua Wang, Yang Yu, Ming Xu, Yadong Liu, Erwei Yin, and Zongtan Zhou. «Towards a Hybrid BCI Gaming Paradigm Based on Motor Imagery and SSVEP». en. In: (), p. 10 (cit. on p. 40).

[21] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, and Bruno Arnaldi. «A review of classification algorithms for EEG-based brain–computer interfaces». en. In: (), p. 26 (cit. on p. 40).

[22] F Lotte, L Bougrain, A Cichocki, M Clerc, M Congedo, A Rakotomamonjy, and F Yger. «A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update». en. In: *Journal of Neural Engineering* 15.3 (June 2018), p. 031005. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/aab2f2. URL: https://iopscience.iop.org/article/10.1088/1741-2552/aab2f2 (visited on 12/02/2020) (cit. on p. 42).

[23] Jdel R Millan and Josep Mouriño. «Asynchronous BCI and local neural classifiers: an overview of the adaptive brain interface project». In: *IEEE transactions on neural systems and rehabilitation engineering* 11.2 (2003), pp. 159–161 (cit. on p. 43).

[24] Po-Lei Lee, Chia-Lung Yeh, J. Y-S Cheng, Chia-Yen Yang, and Gong-Yau Lan. «An SSVEP-Based BCI Using High Duty-Cycle Visual Flicker». en. In: *IEEE Transactions on Biomedical Engineering* 58.12 (Dec. 2011), pp. 3350–3359. ISSN: 0018-9294, 1558-2531. DOI: 10.1109/TBME.2011.2162586. URL: http://ieeexplore.ieee.org/document/5959958/ (visited on 12/02/2020) (cit. on p. 45).

[25] Hao-Teng Hsu et al. «Evaluate the Feasibility of Using Frontal SSVEP to Implement an SSVEP-Based BCI in Young, Elderly and ALS Groups». en. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 24.5 (May 2016), pp. 603–615. ISSN: 1534-4320, 1558-0210. DOI: 10.1109/TNSRE.2015.2496184. URL: https://ieeexplore.ieee.org/document/7335644/ (visited on 12/02/2020) (cit. on p. 45).

[26] Surej Mouli, Ramaswamy Palaniappan, Ian P. Sillitoe, and John Q. Gan. «Performance analysis of multi-frequency SSVEP-BCI using clear and frosted colour LED stimuli». en. In: *13th IEEE International Conference on BioInformatics and BioEngineering*. Chania, Greece: IEEE, Nov. 2013, pp. 1–4. ISBN: 978-1-4799-3163-7. DOI: 10.1109/BIBE.2013.6701552. URL: http:

//ieeexplore.ieee.org/document/6701552/ (visited on 12/02/2020) (cit. on p. 45).

[27] Takeshi Sakurada, Toshihiro Kawase, Kouji Takano, and Tomoaki Komatsu. «A BMI-based occupational therapy assist suit: asynchronous control by SSVEP». en. In: *Frontiers in Neuroscience* 7 (2013). ISSN: 1662-453X. DOI: 10.3389/fnins.2013.00172. URL: http://journal.frontiersin.org/article/10.3389/fnins.2013.00172/abstract (visited on 12/02/2020) (cit. on p. 45).