INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Human electrooculography interface

João Cordovil Bárcia

Dissertação para obtenção do grau de Mestre em

Engenharia Física Tecnológica

Júri

Lisboa, Outubro de 2010

Acknowledgements

I would like to thank Horácio Fernandes for being my supervisor during this past year and allowing me to develop a thesis which is slightly outside the regular confines of the Integrated Master (MSc) in Physics Engineering which I am now completing.

This work would definitely not have been possible without the immense help, patience and generosity of João Fortunato and Rui Dias from IPFN. Quite often their technical knowledge and workbench virtuosity while handling electronics where crucial as much as in propelling this research as in maintaining my sanity.

Rui Fernandes from CANTIC (Centro de Avaliação em Novas Tecnologias de Informação e Comunicação) which works with young students suffering from severe motor disabilities and chronic diseases was also of great importance. Although, due to time restraints, it was unfortunately not possible to explore his knowledge and resources to the fullest he was always available to give positive, constructive and creative feedback regarding my work and, most of all, in making me believe that it could have a real impact as an assistive technology.

I also owe a great deal to the communities who maintain and support the Processing and Arduino open source platforms upon which much of this work was based. Without the aid of these very simple yet powerful tools my work wouldn't have been nearly as motivating. Besides the platforms themselves and the user contributed libraries, the community was always prompt to reply to the questions I posed.

The great support, love and care shown by my friends, family and colleagues during this past six years is a major driving force in every thing I do in life and this thesis is no exception, if anything it is a “living” example of it.

Resumo

Título: Interface electrooculográfico para humanos

Autor: João Cordovil Bárcia

Grau: Mestrado Integrado em Engenharia Física

Coordenador: Horácio Fernandes

Muitas pessoas adquirem à nascença ou em determinado ponto das suas vidas deficiência severas a nível motor e da fala que prejudicam seriamente a sua capacidade de interagir com o mundo exterior a eles mesmos, apesar de possuirem capacidades cognitivas e sociais de outro modo perfeitamente funcionais.

Em muitos destes casos a tecnologia para os ajudar está ao alcance humano mas não se encontra completamente desenvolvida ou sofre de alguma falha. Seja ela custo, design, complexidade, entre outros. Isto faz com que não seja praticável o uso generalizado destas tecnologias apesar dos muitos avanços de louvar que foram alcançados nas últimas duas décadas.

A detecção do movimento ocular e do piscar de olhos através da electrooculografia permite-nos desenhar um interface homem-máquina completo e seguro envolvendo pares de eléctrodos, uma etapa de filtragem e amplificação, um conversor analógico-digital e um software de interface dedicado.

O objectivo princial do trabalho apresentado nesta tese é, em última análise, a criação de um interface homem-máquina sólido de baixo custo baseado em movimento dos olhos que irá permitir a que pessoas com as dificuldades já referidas tenham acesso às ferramentas informáticas mais comumente usadas e das quais estão habitualmente privadas. Isto permitirá a um grande número de pessoas não apenas uma comunicação facilitada com aqueles mais próximos de si mas também a possibilidade de comunicar no espectro inteiro da internet e até a hipótese de realizar actividades gratificantes e productivas de um modo muito mais fácil e poderoso.

Palavras-Chave:

Interface Homem Máquina

Electrooculografia

Tecnologias de apoio a portadores de deficiência

Abstract

Title: Human electrooculography interface
Author: João Cordovil Bárcia
Degree: Masters in Physics Engineering
Coordinator: Horácio Fernandes

Many people acquire either from birth or at a given point of their lives severe physical and speech disabilities which greatly damage their interaction with the world outside themselves, despite having otherwise fully functional cognitive and social capabilities.

In many of this cases the technology to aid them is at human reach but it is either not completely developed or suffers from some shortcoming. Being it either cost, design, complexity, amongst others This makes it not practical for most of these technologies to be used in a wide-spread manner, despite the many laudable advances which have been achieved in the last two decades.

Detecting eye movement and blinking through electrooculography allows us to design a complete and reliable human-computer interface system involving electrical dipole sensing electrodes, a filtering and amplification stage, an analog to digital converter and a software with dedicated interface.

The main purpose of the work presented in this thesis is, ultimately, to create a reliable, affordable and easy-to-use eye based human-computer interface that will enable the afore mentioned people to make use of the most commonly required computer tools from which they are otherwise deprived. This would provide to a large number of people not only an easier communication with those closer them but the possibility of communicating with the whole world wide web and even the chance to engage more easily and powerfully in productive and fulfilling activity.

Keywords:
Human Computer Interface
Electrooculography
Assistive Technology

Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

CRP	Corneo-retinal Potential
EOG	Eleoctrooculogram/Electrooculography
EEG	Electro-encefalogram
HCI	Human-computer interface
LIS	Locked-In Syndrome
ADC	Analog-to-Digital Converter
EMG	Electromyalgy
BCI	Brain-Computer Interface
ALS	Amyotrophic lateral sclerosis
GUI	Graphic User Interface
Ag	Silver
AgCl	Silver Chloride
USB	Universal Serial Bus
RS-232	Recommended Standard 232 (commonly used as computer serial port)
IDE	Integrated Development Envirorment
SMD	Surface Mount Device
DIP	Dual In-line Package
K-NN	K-Nearest Neighbour
PCB	Printed Circuit Board

CHAPTER I

Introduction

In order to better comprehend how the EOG technique is possible **Chapter II – Fundamentals of Electrooculography** gives a basic insight into the inner composition of the human eye, the types of movements it is able of producing and how inherent potential differences can be captured and analysed in order to detect this movements.

Chapter III – To Whom it may be useful describes the multiple uses that the proposed device delivers, focusing on the main one to which it was designed – as an assistive technology.

An overview of the current comparable technology is presented in **Chapter IV – State of the Art**. The pros and cons of the proposed device are presented against equivalent assistive technologies (brain-computer interfaces and electromyalgy), alternative eye tracking methods (mostly video based) and directly against already existing EOG devices.

The complete process of research and development is explained in depth in **Chapter V – The Device**, accompanying the materials, methods and options taken throughout the work. This chapter is organized according to the inherent structure of the device – electronics, microcontroller, software and casing. The electronics part of the work is where the signal is filtered and amplified accordingly to be able to be interpreted by the microcontroller which interprets the signal and communicates to the graphic user interface software if a known pattern is detected.

In **Chapter VI – Results** the device's performance is evaluated in terms of accuracy, cost and ease-of-use with what is considered to be a very positive outcome.

The future of the device in terms of improvements and possibility of public release is discussed in **Chapter VII – Conclusions and Prospects**.

All of the works which have provided some insight into the development of the proposed device are listed in **Bibliography** while the **Appendixes** contain the source code of all of the applications developed during the course of this work.

CHAPTER II

Fundamentals of Electrooculography

"Electricity is actually made up of extremely tiny particles
called electrons, that you cannot see with the naked eye
unless you have been drinking."
Dave Barry, *The Taming of the Screw*, 1983

Electrooculography is, simply put, the science of measuring the resting potential of the eye and its variations. The cornea is in fact positive relatively to the posterior part of the sclera which can be considered the front-most and rear-most parts of the eye bulb respectively. This is called the corneo-retinal potential (CRP).

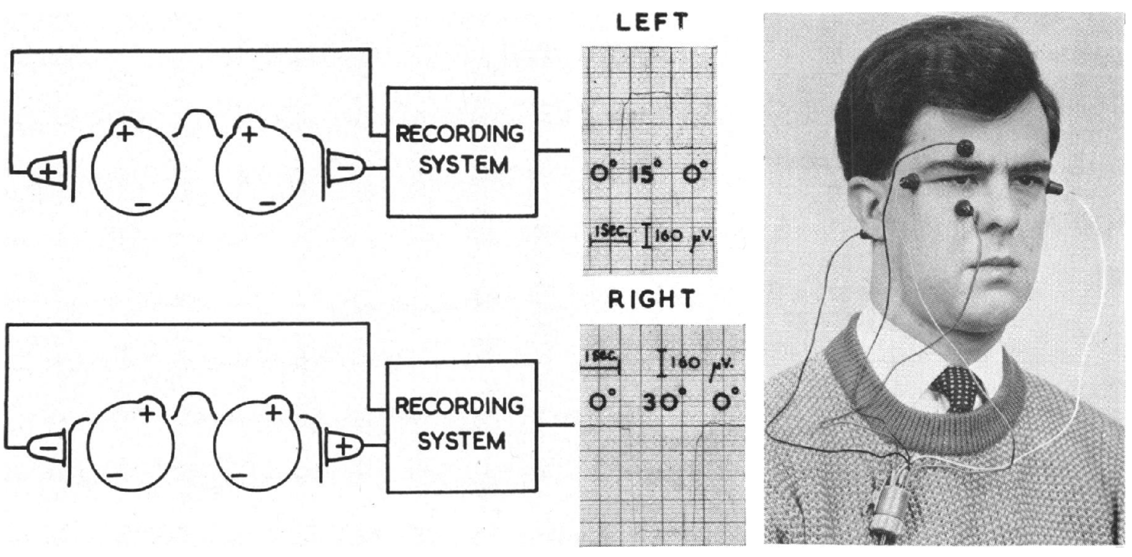


Fig. 2.1: Excerpts of an early EOG work - *A Second Survey With Electro-oculography*, 1960

Although in 1849 Du Bois-Reymond[1] had already discovered the existence of a resting potential throughout the eye, it was only during the second half of the 20th century that there was a visible rise in the efforts dedicated to the study of the analysis of this potential. It is quite revealing that a paper published in 1959 by B. Schackel was still being titled *Pilot Study in Electrooculography*[9]. Until the end of the 20th century its most widespread use had been in the field of ophtalmological diagnosis and varied physiological research on fields as different as sleep disorders and neurodegenerative diseases[10][11][12][13]. Most recently some of the focus has switched to the tracking of eye movements, accompanying the evolution of the paradigm of multimedia and interface computation[14][15][16][17][18][19].

It is this later use that is of interest for the present work. Nevertheless we should first understand how all this potentials come to be and how we are able to "harness" them turning it into usable information.

2.1 An introduction to the mechanics of the eye

Our ability to "see" starts when light reflects off an object at which we are looking and enters the eye. As it enters the eye, the light is unfocused. The first step in seeing is to focus the light rays onto the retina, which is the light sensitive layer found inside the eye. Once the light is focused, it stimulates cells to send millions of electrochemical impulses along the optic nerve to the brain. The portion of the brain at the back of the head (the visual cortex) interprets the impulses, enabling us to see the object.

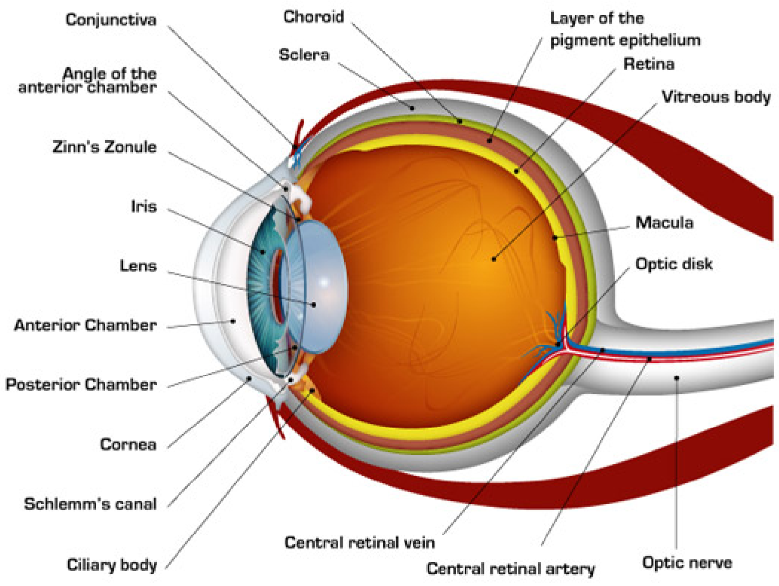


Fig. 2.2: Diagram of a human eye

Light entering the eye is first bent, or refracted, by the cornea -- the clear window on the outer front surface of the eyeball. The cornea provides most of the eye's optical power or light-bending ability.

After the light passes through the cornea, it is bent again -- to a more finely adjusted focus -- by the crystalline lens inside the eye. The lens focuses the light on the retina. This is achieved by the ciliary muscles in the eye changing the shape of the lens, bending or flattening it to focus the light rays on the retina.

This adjustment in the lens, known as accommodation, is necessary for bringing near and far objects into focus. The process of bending light to produce a focused image on the retina is called "refraction". Ideally, the light is "refracted," or redirected, in such a manner that the rays are focused into a precise image on the retina.

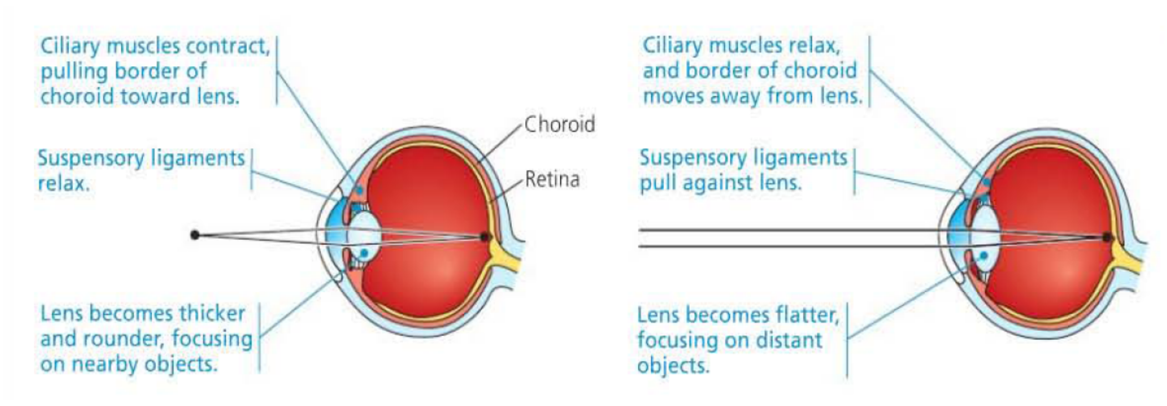

Ciliary muscles contract, pulling border of choroid toward lens.
Suspensory ligaments relax.
Choroid
Retina
Lens becomes thicker and rounder, focusing on nearby objects.
Ciliary muscles relax, and border of choroid moves away from lens.
Suspensory ligaments pull against lens.
Lens becomes flatter, focusing on distant objects.

Fig. 2.3: Focusing in the mammalian eye for near vision and distance vision

Two types of receptors -- rods and cones -- are present. Rods are mainly found in the peripheral retina and enable us to see in dim light and to detect peripheral motion. They are primarily responsible for night vision and visual orientation. Cones are principally found in the central retina and provide detailed vision for such tasks as reading or distinguishing distant objects. They also are necessary for colour detection. These photoreceptors convert light to electrochemical impulses that are transmitted via the nerves to the brain.

Millions of impulses travel along the nerve fibres of the optic nerve at the back of the eye, eventually arriving at the visual cortex of the brain, located at the back of the head. Here, the electrochemical impulses are unscrambled and interpreted.

Yet for many reasons which will be addressed in chapter 2.2 our eyes have the ability to track or focus on specific points in space. In order to do that they are, unlike our other senses, able to move independently from the head.

2.2 Different types of eye movement

Human ocular movement has been widely studied in neurophysiology and psychology. These studies indicate that the extraocular muscles Fig 2.4 work together to achieve four basic types of eye movements: saccades, smooth pursuit movements, vergence movements, and vestibulo-ocular movements.

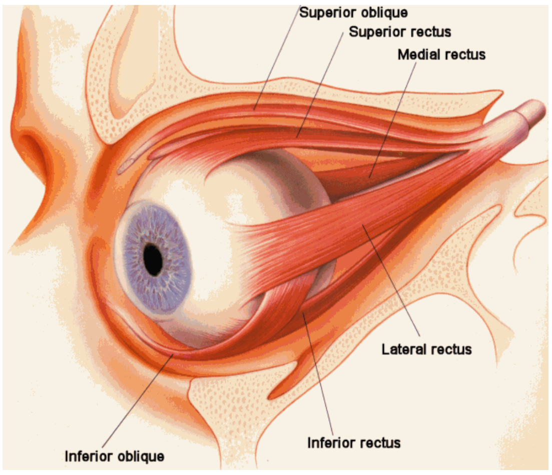


Fig. 2.4: Lateral view of the extraocular muscles

Saccades

Saccades are rapid, ballistic movements of the eyes that abruptly change the point of fixation. They range in amplitude from the small movements made while reading, for example, to the much larger movements made while gazing around a room. Saccades can be voluntary, but occur reflexively whenever the eyes are open, even when fixated on a target. The rapid eye movements that occur during an important phase of sleep are also saccades. The time course of a saccadic eye movement is shown in below. The red line indicates the position of a fixation target and the blue line the position of the fovea (central part of the retina). When the target moves suddenly to the right, there is a delay of about 200 ms before the eye begins to move to the new target position.

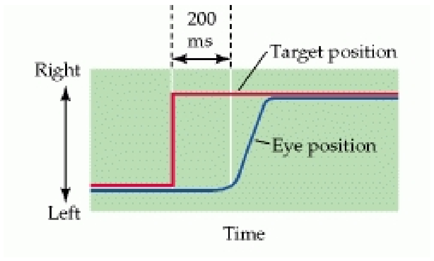


Fig. 2.5: The metrics of a saccadic eye movement.

After the onset of a target for a saccade (in this example, the stimulus was the movement of an already fixated target), it takes about 200 ms for eye movement to begin. During this delay, the position of the target with respect to the fovea is computed (that is, how far the eye has to move), and the difference between the initial and intended position is converted into a motor command that activates the extraocular muscles to move the eyes the correct distance in the appropriate direction. Saccadic eye movements are said to be ballistic because the saccade-generating system cannot respond to subsequent changes in the position of the target during the course of the eye movement. If the target moves again during this time (which is on the order of 15–100 ms), the saccade will miss the target, and a second saccade must be made to correct the error.

Saccadic eye movements will be the ones truly relevant for the development of the current work. Yet, it is beneficial to understand the other types of movement the eye is able to produce in order to be able to classify them as artefacts and remove them from electronic readings when necessary.

Smooth pursuit movements

Smooth pursuit movements are much slower tracking movements of the eyes designed to keep a moving stimulus on the fovea. Such movements are under voluntary control in the sense that the observer can choose whether or not to track a moving stimulus.(Saccades can also be voluntary, but are also made unconsciously.)

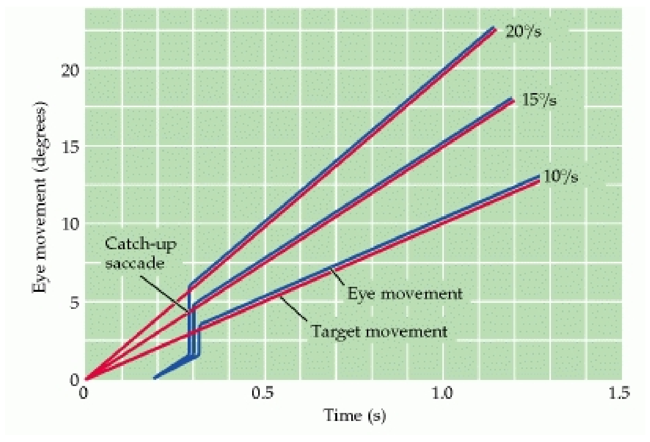


Fig. 2.6: The metrics of smooth pursuit eye movements

These traces show eye movements (blue lines) tracking a stimulus moving at three different velocities (red lines). After a quick saccade to capture the target, the eye movement attains a velocity that matches the velocity of the target.

Surprisingly, however, only highly trained observers can make a smooth pursuit movement in the absence of a moving target. Most people who try to move their eyes in a smooth fashion without a moving target simply make a saccade.

The smooth pursuit system can be tested by placing a subject inside a rotating cylinder with vertical stripes. (In practice, the subject is more often seated in front of a screen on which a series of horizontally moving vertical bars is presented to conduct this “optokinetic test.”) The eyes automatically follow a stripe until they reach the end of their excursion. There is then a quick saccade in the direction opposite to the movement, followed once again by smooth pursuit of a stripe. This alternating slow and fast movement of the eyes in response to such stimuli is called optokinetic nystagmus.

Vergence movements

Vergence movements align the fovea of each eye with targets located at different distances from the observer. Unlike other types of eye movements in which the two eyes move in the same direction (conjugate eye movements), vergence movements are disconjugate; they involve either a convergence or divergence of the lines of sight of each eye to see an object that is nearer or farther away. Convergence is one of the three reflexive visual responses elicited by interest in a near object. The other components of the so-called near reflex triad are, as we have seen before, the accommodation of the lens, which brings the object into focus, and pupillary constriction, which increases the depth of field and sharpens the image on the retina.

Vestibo-ocular movements

Vestibulo-ocular movements stabilize the eyes relative to the external world, thus compensating for head movements. These reflex responses prevent visual images from “slipping” on the surface of the retina as head position varies. The action of vestibulo-ocular movements can be appreciated by fixating an object and moving the head from side to side; the eyes automatically compensate for the head movement by moving the same distance but in the opposite direction, thus keeping the image of the object at more or less the same place on the retina. The vestibular system detects brief, transient changes in head position and produces rapid corrective eye movements. Sensory information from the semicircular canals directs the eyes to move in a direction opposite to the head movement. While the vestibular system operates effectively to counteract rapid movements of the head, it is relatively insensitive to slow movements or to persistent rotation of the head. For example, if the vestibulo-ocular reflex is tested with continuous rotation and without visual cues about the movement of the image (i.e.,with eyes closed or in the dark), the compensatory eye movements cease after only about 30 seconds of rotation. However, if the same test is performed with visual cues, eye movements persist. The compensatory eye movements in this case are due to the activation of the smooth pursuit system, which relies not on vestibular information but on visual cues indicating motion of the visual field.

2.3 The Resting Potential

All cells have voltages across their plasma membranes. Voltage is electrical potential energy-a separation of opposite charges. The cytoplasm is negative in charge relative to the extracellular fluid because of an unequal distribution of anions and cations on opposite sides of the membrane. The voltage across a membrane, called a membrane potential, ranges from about - 50 to - 200 millivolts (the minus sign indicates that the inside of the cell is negative relative to the outside).

When the membrane potential of a cell can go for a long period of time without changing significantly, it is referred to as a resting potential, standing potential or resting voltage. This terms are used for the membrane potential of non-excitable cells, but also for the membrane potential of excitable cells in the absence of excitation.

The membrane potential acts like a battery, an energy source that affects the traffic of all charged substances across the membrane. Because the inside of the cell is negative compared with the outside, the membrane potential favours the passive transport of cations into the cell and anions out of the cell. Thus, two forces drive the diffusion of ions across a membrane: a chemical force (the ion's concentration gradient) and an electrical force (the effect of the membrane potential on the ion's movement). This combination of forces acting on an ion is called the electrochemical gradient.

In the case of ions, then, we must refine our concept of passive transport: An ion diffuses not simply down its concentration gradient but, more exactly, down its electrochemical gradient. For example, the concentration of sodium ions (Na+) inside a resting nerve cell is much lower than outside it. When the cell is stimulated, gated channels open that facilitate Na+ diffusion. Sodium ions then "fall" down their electrochemical gradient, driven by the concentration gradient of Na+ and by the attraction of these cations to the negative side of the membrane. In this example, both electrical and chemical contributions to the electrochemical gradient act in the same direction across the membrane, but this is not always so. In cases where electrical forces due to the membrane potential oppose the simple diffusion of an ion down its concentration gradient, active transport may be necessary.

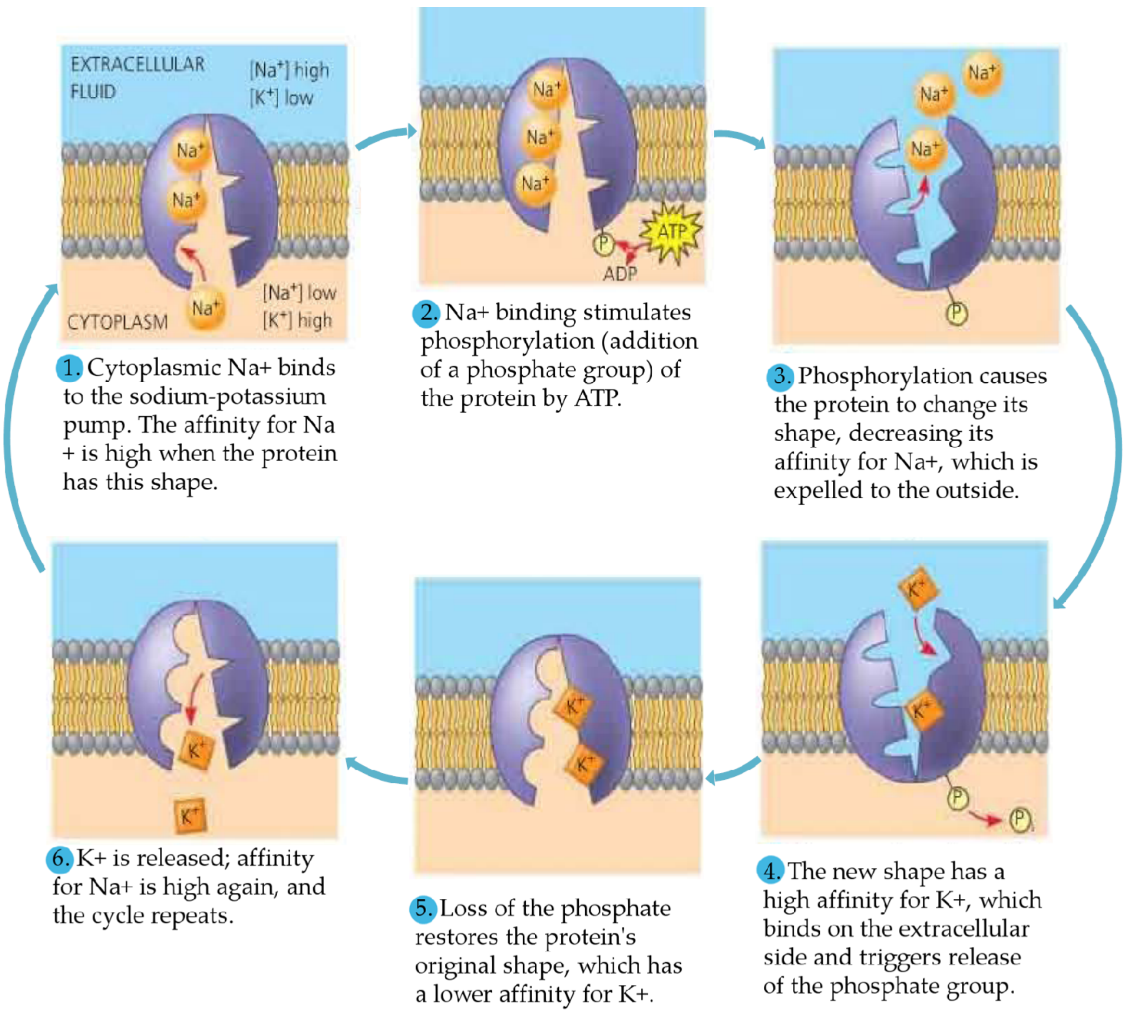

EXTRACELLULAR FLUID
[Na+] high
[K+] low
Na+
Na+
Na+
CYTOPLASM
[Na+] low
[K+] high
1. Cytoplasmic Na+ binds to the sodium-potassium pump. The affinity for Na + is high when the protein has this shape.
Na+
Na+
Na+
ATP
P
ADP
2. Na+ binding stimulates phosphorylation (addition of a phosphate group) of the protein by ATP.
Na+
Na+
Na+
P
3. Phosphorylation causes the protein to change its shape, decreasing its affinity for Na+, which is expelled to the outside.
K+
K+
P
P
4. The new shape has a high affinity for K+, which binds on the extracellular side and triggers release of the phosphate group.
K+
K+
5. Loss of the phosphate restores the protein's original shape, which has a lower affinity for K+.
K+
K+
6. K+ is released; affinity for Na+ is high again, and the cycle repeats.

Fig. 2.7: The sodium-potassium active membrane pump

The transport system shown in Fig. 2.7 pumps ions against steep concentration gradients: Sodium ion concentration (represented as [Na$^+$] is high outside the cell and low inside, while potassium ion concentration ([K$^+$]) is low outside the cell and high inside. The pump oscillates between two shapes in a pumping cycle that translocates three sodium ions out of the cell for every two potassium ions pumped into the cell. The two shapes have different affinities for the two types of ions. ATP powers the shape change by phosphorylating the transport protein (that is, by transferring a phosphate group to the protein).

Some membrane proteins that actively transport ions contribute to the membrane potential. An example is the sodium-potassium pump. Notice in Fig. 2.7 that the pump does not translocate Na+ and K+ one for one, but pumps three sodium ions out of the cell for every two potassium ions it pumps into the cell. With each "crank" of the pump, there is a net transfer of one positive charge from the cytoplasm to the extracellular fluid, a process that stores energy as voltage. A transport protein that generates voltage across a membrane is called an electrogenic pump. The sodium-potassium pump seems to be the major electrogenic pump of animal cells.

This difference in ionic concentration that exists between the inside and outside of this membrane can be related to the diffusion potential – also known as the Nernst Potential. The diffusion potential level across a membrane that exactly opposes the net diffusion of a particular ion through the membrane is called the Nernst potential for that ion. The magnitude of this Nernst potential is determined by the ratio of the concentrations of that specific ion on the two sides of the membrane. The greater this ratio, the greater the tendency for the ion to diffuse in one direction, and therefore the greater the Nernst potential required to prevent additional net diffusion. The following equation, called the Nernst equation, can be used to calculate the Nernst potential for any univalent ion:

$$E_{eq,K^+} = \frac{RT}{zF} \ln \frac{[K^+]_{\text{out}}}{[K^+]_{\text{in}}} \tag{2.1}$$

Where:

- $E_{eq,K+}$ is the equilibrium potential for potassium, measured in volts
- R is the universal gas constant, equal to 8.314 joules·K^{-1}·mol^{-1}
- T is the absolute temperature, measured in kelvin
- z is the number of elementary charges of the ion in question involved in the reaction
- F is the Faraday constant, equal to 96,485 coulombs·mol^{-1}
- $[K+]_{out}$ is the extracellular concentration of potassium, measured in mol·m^{-3}
- $[K+]_{in}$ is likewise the intracellular concentration of potassium

Which at normal human body temperature of 37°C is given by:

$$E_{eq,K^+} = 61{,}54 mV \log \frac{[K^+]_{\text{out}}}{[K^+]_{\text{in}}} \tag{2.2}$$

When using this formula, it is usually assumed that the potential in the extracellular fluid outside the membrane remains at zero potential, and the Nernst potential is the potential inside the membrane. Also, the sign of the potential is positive (+) if the ion diffusing from inside to outside is a negative ion, and it is negative (–) if the ion is positive. Thus, when the concentration of positive potassium ions on the inside is 10 times that on the outside, the log of 10 is 1, so that the Nernst potential calculates to be –61 millivolts inside the membrane.

However most of the membranes are, in reality, permeable to several different ions causing the diffusion potential to develop depending on three factors:

- the polarity of the electrical charge of each ion,
- the permeability of the membrane (P) to each ion, and
- the concentrations (C) of the respective ions on the inside (i) and outside (o) of the membrane.

Thus, the following formula, called the Goldman equation, or the Goldman-Hodgkin-Katz equation, gives the calculated membrane potential on the inside of the membrane when two univalent positive ions, sodium (Na+) and potassium (K+), and one univalent negative ion, chloride (Cl–), are involved.

$$E_m = \frac{RT}{F}\ln\left(\frac{P_K[K^+]_{\mathrm{out}}+P_{Na}[Na^+]_{\mathrm{out}}+P_{Cl}[Cl^-]_{\mathrm{in}}}{P_K[K^+]_{\mathrm{in}}+P_{Na}[Na^+]_{\mathrm{in}}+P_{Cl}[Cl^-]_{\mathrm{out}}}\right) \tag{2.3}$$

This equation resembles the Nernst equation (eq. 2.1), but has a term for each permanent ion. Also, z has been inserted into the equation, causing the intracellular and extracellular concentrations of Cl- to be reversed relative to K^+ and Na^+, as chloride's negative charge is handled by inverting the fraction inside the logarithmic term. E_m is the membrane potential, measured in volts R, T, and F are as above, P_X is the relative permeability of ion X in arbitrary units (e.g. siemens for electrical conductance) and $[X]_Y$ is the concentration of ion X in compartment Y as above.

Let us study the importance and the meaning of this equation. First, sodium, potassium, and chloride ions are the most important ions involved in the development of membrane potentials in nerve and muscle fibres, as well as in the membranes that compose the eye. The concentration gradient of each of these ions across the membrane helps determine the voltage of the membrane potential.

Second, the degree of importance of each of the ions in determining the voltage is proportional to the membrane permeability for that particular ion. That is, if the membrane has zero permeability to both potassium and chloride ions, the membrane potential becomes entirely dominated by the concentration gradient of sodium ions alone, and the resulting potential will be equal to the Nernst potential for sodium. The same holds for each of the other two ions if the membrane should become selectively permeable for either one of them alone.

Third, a positive ion concentration gradient from inside the membrane to the outside causes electronegativity inside the membrane. The reason for this is that excess positive ions diffuse to the outside when their concentration is higher inside than outside. This carries positive charges to the outside but leaves the non-diffusible negative anions on the inside, thus creating electronegativity on the inside. The opposite effect occurs when there is a gradient for a negative ion. That is, a chloride ion gradient from the outside to the inside causes negativity inside the cell because excess negatively charged chloride ions diffuse to the inside, while leaving the non-diffusible positive ions on the outside.

2.4 An electronic approximation of the resting potential

Electrophysiologists model the effects of ionic concentration differences, ion channels, and membrane capacitance in terms of an equivalent circuit, which is intended to represent the electrical properties of a small patch of membrane. The equivalent circuit consists of a capacitor in parallel with four pathways each consisting of a battery in series with a variable conductance. The capacitance is determined by the properties of the lipid bilayer, and is taken to be fixed. Each of the four parallel pathways comes from one of the principal ions, sodium, potassium, chloride, and sometimes calcium. The voltage of each ionic pathway is determined by the concentrations of the ion on each side of the membrane. The conductance of each ionic pathway at any point in time is determined by the states of all the ion channels that are potentially permeable to that ion, including leakage channels, ligand-gated channels, and voltage-dependent channels.

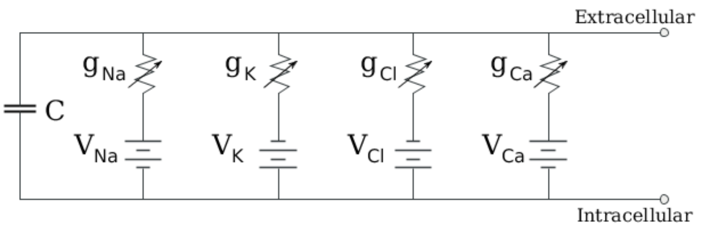


Fig. 2.8: Equivalent circuit for a patch of membrane

In Fig. 2.8 we can observe a diagram of an equivalent circuit consisting of a fixed capacitance in parallel with four pathways each containing a battery in series with a variable conductance.

For fixed ion concentrations and fixed values of ion channel conductance, the equivalent circuit can be further reduced, using the Goldman equation as described below, to a circuit containing a capacitance in parallel with a battery and conductance. Electrically this is a type of RC circuit (resistance-capacitance circuit), and its electrical properties are very simple. Starting from any initial state, the current flowing across either the conductance or capacitance decays with an exponential time course, with a time constant of τ = RC, where C is the capacitance of the membrane patch, and R = $1/g_{net}$ is the net resistance.

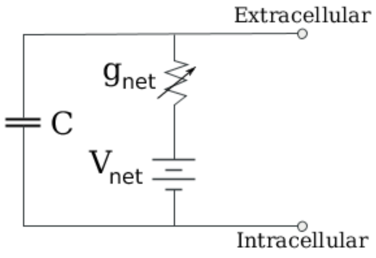


Fig. 2.9: Reduced circuit obtained by combining the ion-specific pathways using the Goldman equation

For realistic situations the time constant usually lies in the 1—100 millisecond range. In most cases changes in the conductance of ion channels occur on a faster time scale, so an RC circuit is not a good approximation; however the differential equation used to model a membrane patch is commonly a modified version of the RC circuit equation.

2.5 Light Potential

Besides the potential difference in the eye-bulb which is due to the previous stated motives, CRP also presents some dependency relatively to the amount of light entering the eye[22][23]. This correlation, however, is of much smaller amplitude than the standing potential of the eye and does not present a noticeable problem. A maximum shift of 30% relatively to the mean value in base-line potential has been achieved when maximizing light potential variation through a very specific cycle of switching on and off a light source directly into the patient's eye. As such, the light potential will not be considered relevant for the present work.

2.6 The resting potential in the human eye and its analysis

Due to this phenomenon the eyeball may actually be regarded as a small battery, with a positive pole in the cornea and a negative pole in the retina although it's mechanics are slightly more complex than in the case of a single cell, neuron or muscle fibre as has been discussed so far. In the eye most of the resting potential has been attributed to the retina although some research has been done which proves the lens and maybe the cornea also play important roles.

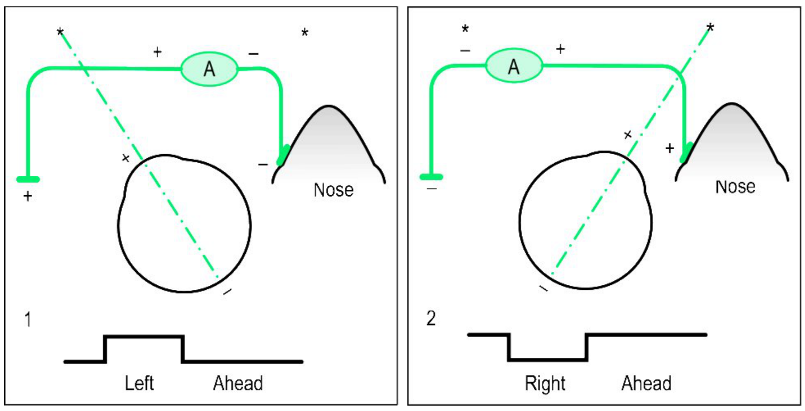


Fig. 2.10: Skin surface potential variations due to CRP

The corneoretinal potential is roughly aligned with the optic axis. Hence it rotates with the direction of gaze. Changes in the position of the eyeball cause changes in potential at the skin surface around the eye socket which can be measured by surface electrodes placed on the skin around the eyes. In the first diagram of Fig. 2..10 a left gaze is displayed - the cornea approaches the electrode near the outer canthus resulting in a positive-going change in the potential difference recorded from it. In the second diagram a right gaze is displayed - the cornea approaches the electrode near the inner canthus resulting in a positive-going change in the potential difference recorded from it. A is an AC/DC amplifier. Below each diagram is a typical tracing of the DC voltage. The changes of potential are therefore, for practical purposes, linearly related to the angle of the eye-ball rotation and can been used as measures of eye movement and eye position. However valid it may be, this is only an approximation to the actual biological system since there exists a non-uniformity of the tissues and the shapes of the tissues surrounding the eye.

It is possible to obtain independent measurements from the two eyes. However, unless the user suffers from some serious type of strabismus or happens to have chameleon like abilities, the two eyes move in conjunction in the vertical and horizontal direction. Regarding the horizontal movement, if we were to study the individual position of one of the eyes, an electrode would have to be placed between the eye and the nose (as is depicted on Fig. 2.10) on a position that would not only be awkward but could provide less accurate readings, since it would most likely capture a slightly combined potential from both eyes. Hence it is sufficient to measure the vertical motion of only one eye together with the horizontal motion of both eyes trough two pairs of dipoles. In order to isolate much of the noise and to have an electrical reference a ground electrode must be used as well and placed outside of the reach of influence of the CRP on the skin surface. It is most oftenly placed on the forehead. The standard electrode placement is displayed in Fig. 2.11.

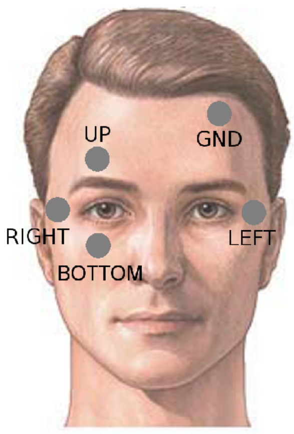


Fig. 2.11: EOG electrode positioning

The amplitude of the potential variations captured by the electrodes on the skin surface are quite small (around 750 μV), so they must be amplified before processing. Straightforward signal processing steps can be devised to condition the data so it can be reliably interpreted. Some of the noise patterns such as the 50 Hz line frequency (or 60Hz in the US) can be easily removed, applying appropriate filtering like a low pass or a notch filter. The specific filtering stages adopted for this work will be further developed on chapter 5.1. Other noise artefacts are mostly caused, for example, by the turning of an electrical switch on/off in the vicinity of the electrodes, contraction of the facial or neck muscles, slippage of the electrode due to sweat and eye blinking. However, the signals produced by eye blinks are, in fact, quite regular. This makes it easy to recognize them and categorized as such. In other words, the EOG technique can recognize eye "gestures" such as winking, blinking or a combination.

2.7 Considerations

EOG is a technique that can provide very solid and powerful results. Yet it is far from perfect and its limitations should be understood.

Individual User Calibration

Each person has its own unique electrical pattern during eye movement. This is due to differences in skin conductance and individual physiological properties as well as different ways to move the eyes. “Looking up” and especially composite movements like “Looking up-right” mean different things in terms of the rotation of the eye-ball both in speed and angle produced by each user. As such, depending on the purpose of the EOG and the pattern detection algorithms being used, it is sometimes necessary to calibrate the EOG apparatus for each new user. Automatic detection algorithms and passive systems[15] have been devised but they still present a considerably lower accuracy and tend to force the user to adapt to what “looking up-right” means for the system.

Long Term Variation of Surface Skin Potential

Although the electrical patterns produced by the movement of the eye vary from person to person, some studies indicate that each individual maintains a very stable surface skin potential given that the ambient conditions (like light, humidity, approximate electrode placement, amongst others) remain practically the same. The CRP values of 126 subjects were evaluated with a 10 month interval with a great degree of correlation[25].

Head Movement

The mentioned eye tracking technique does not inherently produce a value which defines where the user's gaze is directed to. It is not able to understand where the user is looking at without a previous screen-specific calibration which mainly depends on the screen size, the distance between the user and the screen, and the center of vision of the user. The first parameter is obviously quite static while the distance to the screen and especially the center of vision can change very harshly with simple head movements. For instance, if you tilt your head down in order to look at the center of the screen you must move your eyes up, leading the system to believe that you are looking much higher then you actually are.

Midas Touch

The primary function of the eyes is to enable vision. Using the eyes for computer input might result in conflicts. This well-known conflict is the Midas Touch problem – for the eye-gaze interface it is difficult to decide whether our gaze is on an object just for inspection or for invoking an action. Misinterpretation by the gaze interface can trigger unwanted actions wherever we look. The situation is similar when triggering actions by eye movements, i.e. gestures or blinking. The eye-gaze interface has to separate natural eye movements from intentional gaze gestures. Distraction by moving or blinking objects might also cause conflicts.

Eye Fatigue

From other input devices we know that extensive use of particular muscles or muscle groups can cause physical problems called repetitive strain injury. There are fears that this might happen to the eye muscles too. The concern is justified and should be taken seriously, but as the eyes move constantly, even while we sleep, it might not turn out to be a problem. The magnitude of this problem would only be possible to assess with a large long-term study and prolonged use of the EOG method which was not possible to achieve in this work.

Ambient Light Variations

As has been discussed during the present chapter, the CRP is not only caused by inherent standing membrane potential but by the passage of light through the photoreceptors as well. As such, when extreme ambient light variations occur a variation in skin potential readings might occur as well. Nevertheless the variations do not have a large enough amplitude to be relevant[22] [23].

Electrode Sensitivity

Throughout the course of the experimental work done towards the completion of this thesis it quickly became obvious the importance that, due to the very small scale of the signal being interpreted (hundreds of microvolt) the quality of the reading given by the electrodes is of the utmost importance. It is essential to assure that the electrodes remain stably positioned and with a good contact surface. This can pose a problem in high humidity situations or if the patient is sweating considerably. This is known as a sweat artefact. The specific hardware and and solutions taken in this work are presented in chapter 5.4.

CHAPTER III

To whom it may be useful

"Does the cosmos contain keys for opening up my diving bell? A subway line with no terminus? A currency strong enough to buy my freedom back? We must keep looking. I'll be off now."
Jean-Dominique Bauby,
Le Scaphandre et le Papillon
(written while suffering from locked-in syndrom)

The device being developed in this work is primarily directed towards people suffering from severe lack of motor and speech coordination. This set of disabilities can be brought upon by many different pathologies.

While it is definitely helpful for a large number of disabled people to be able to access basic computer tools, in extreme cases achieving it through an eye based input might actually be the only viable option. Although rare, Locked in Syndrome (LIS) is an extremely debilitating condition. It is the limit condition of lack of motor and speech coordination when a patient is not able to speak neither move any muscle in his (or her) body except for the extraocular muscles eyes. It might be brought upon by spinal cord injury, traumatic brain injury, or other physical trauma, neurological conditions including motor neuron disease such as amyotrophic lateral sclerosis, multiple sclerosis, Lesch Nyhan Syndrome, or cerebral palsy causing quadriplegia - and perhaps also loss of speech or vision. In most of this cases the only traditional method of communication is through a system of eye blinks.
On many cases the ability to communicate has a great relevance not only for the mental and social well-being of the patient but as well as to allow the sharing of symptoms which can lead to diagnosis. This provides the present technology with an inherently health-care quality.

Baud Rate

It is important to grasp the concept of Baud Rate when addressing the power of assistive technologies. Both from a scientific and a human point of view. By definition it means the number of times a signal in a communications channel changes state or varies. In order to relate it to human communication I find it adequate to use the words of Stephen Hawking as he addresses it in his Foreword to Computer Resources for People with Disabilities[4].

> The main problem of communicating without being able to speak is what is called the baud rate, the rate at which information can be conveyed. Normal speech is between 120 and 180 words a minute. By contrast, a reasonable typist can produce 40 to 60 words a minute. Thus, if people were equipped with keyboards to communicate, they could do so at only one-half to one-quarter of the speech rate.
>
> However, many people like me who cannot speak also have other disabilities. They cannot use a keyboard. Instead, they can use one or more switches, operated by a head or hand movement. This is where a person is really confronted with the rate of information flow. If you take an average word to be five characters and assume that any character can follow any other character, normal speech has an information flow rate of between 50 and 75 bits a second. By contrast, a person might be able to operate a switch at two or three bits a second.
>
> The real information flow in communication, however, is much less than this. (In the case of political speeches it is practically zero). This is because spelling out a sentence letter by letter is inefficient. Most sequences of letters don't make recognizable words, let alone meaningful sentences. It takes a handful of these bits of information (letters) to create meaningful communication (a word). So, communicating by specifying every letter is a lot of redundant effort.
>
> For someone who uses a switch to communicate, it is much more efficient to pick words or even whole phrases from a list of likely ones. Computer technology makes this possible. To translate the press of a switch into letters, words, or sentences requires computers. The development of microprocessors in the last 15 years has meant that virtually unlimited computing power is available at reasonable cost.
>
> The next step is to find efficient software to translate the input from the switch into phrases and sentences. With the Equalizer program that I use, I can manage about 15 words a minute. That is not too bad, since an information flow rate of three bits a second corresponds to 25 to 30 words a minute. But obviously there is scope for improvement. And the promise of computer technology is that improvements are always in development. (…)
>
> Stephen Hawking,
> in *Foreword to Computer Resources for People with Disabilities*,
> 1994

…and Bit Rate

Baud Rate and Bit Rate are dependent and inter-related. But the simplest explanation is that a Bit Rate is how many data bits are transmitted per second. A baud Rate is the measurement of the number of times per second a signal in a communications channel changes.

So the bit rate (bits per second or bps) and baud rate (baud per second) have this connection:

Bit Rate = Baud Rate x Number of bit per baud

To use a simple analogy, imagine that you could only communicate with the world with 2 trigger buttons (as is the case with some people who suffer from severe lack of coordination). The baud rate would be equivalent to the number of times per second you are able to press the buttons. The number of bit per baud would be two because each time you perform a “communication cycle” you can change the state of two different pieces of information. The bit rate would be the product of the former two.

This is essential to better understand the paradigms of interface design and, especially assistive interface design. An HCI technology benefits not only from how quickly you can communicate with a computer but from the number of different types of instructions you can give during a “communication cycle”.

Going beyond

This work focuses mainly on developing an assistive technology, nevertheless its possible applications span far beyond that reach. Controlling a computer or any other hardware with the eyes might be of use in many different situations such as:

- while working on an activity that involves permanent use of the hands – in a medical surgery setting you could give instructions to a video camera or control the illumination, while soldering,
- extending the number of inputs in your daily use of an operative system - changing the desktop you are working on (something common on Linux's Gnome GUI) , browsing through a photo album,
- gaming – recent years have definitely showed us that the general public is quite eager to interact with gaming platforms in fresh and inventive ways,
- as a musical controller for performance - a digital drum set played with eye movement or changing the pitch of a sound with the position of your eyes,
- while performing physical activity, like skipping a song or controlling the volume in your music player while jogging,
- and so on...

Although the goal of assistive technology is of special interest to me, the ways a user can process and interact with an uncharted and fairly new input are, in fact, pretty much endless. They depend solely upon our imagination and will to implement.

CHAPTER IV

State of the Art

“A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.“
Antoine de Saint-Exupéry

4.1 Equivalent assistive technology

This present work, as has been mentioned in the previous chapter, is mainly geared towards people who suffer from severe lack of speech and motor coordination, despite it's many potential uses. As such it is only worthwhile to compare eye tracking based HCI assistive systems with technology directed towards this same disabilities. Although in practice there is a much wider range of people who suffer from less severe disabilities to whom this technology also might be of use.

The only technologies directly equivalent to eye tracking, regarding the fact that they require absolutely no voice neither movement are the electroencefalogram (EEG), which is the analysis of the electric signal produced by brain wave patterns and the electromyography (EMG), which is the analysis of the electric signal received by muscles from the brain. EEG and EMG have been used on a clinical diagnosis and research setting for a long time and have recently been appropriated by HCI researchers. Especially EEG due to it's powerful potential. When an HCI's input is the EEG it is called a Brain Computer Interface (BCI).

It is very promising and some commercial systems geared mostly towards entertainment and gaming like the EPOC Headset by Emotiv or the NeuroSky Mindset by Neurosky have been released to the market with mild success.
Nevertheless, even these specific BCI systems which have seen quite a large investment in research and marketing are reviewed as having a lower than expected reliability and accuracy providing and entertaining although far from stable experience[44].

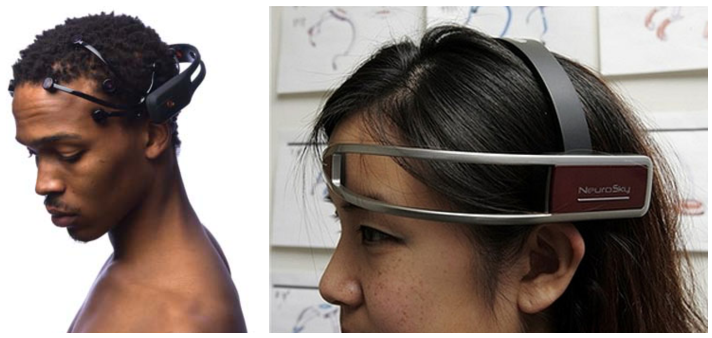


Fig. 4.1: Epoc Headset and Neurosky Mindset – two current BCI systems on the market

And then again, maybe we're just not used to control our brain in a way that is effective for BCI systems just as elder people nowadays are not used to control their hand movement in a way that is effective for cell-phone messaging and similar interfaces. Even if this is true and in the future we turn out to be techno-excluded old men (and women) in a world where a new generation grows up being used to interface with their computers directly through the brain input, this work is directed to help the current generation of disabled people in a way that suits their present skills the best.

Regarding EMG devices a commercial system developed by Control Bionics – the Neuroswitch has also been released and is specifically designed from the bottom up as an assistive technology. Instead of tracking brain wave patterns, its electrodes pick up the electric signals sent by the brain to activate the muscles weather these muscles still work or not. The basic setup is only able to detect signals from two muscles.

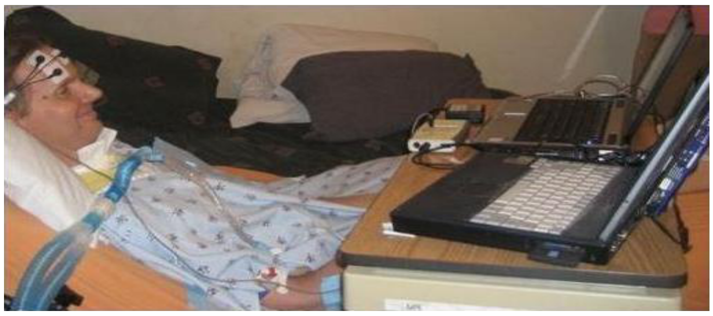


Fig. 4.2: Neuroswitch system being used by a LIS patient

Let us roughly compare the main options for people who suffer from severe lack of speech and motor coordination.

- **Human Assisted Writing** – By having a trained human assistant repeat the whole alphabet to the patient and the patient blinking the eye when the desired character has been reached. It takes between 0.5s to 15s to write a single character with a degree of error that depends on many human variables. A specially reorganized alphabet is used, considering the frequency rate of a character for the appropriate language and, with enough intuition and knowledge of the patient, the assistant becomes able to complete his words beforehand. Although the presence of a human assistant is certainly invaluable from a psychological point of view for the patient, the associated costs and level of dependency might not be supportable and/or desirable. This technique involves a somewhat steep training curve for both the patient and the assistant as well .
- **Button Triggers –** Similar to the previous technique. The human assistant is replaced by a computer which consecutively selects letters in a reorganized alphabet and the eye blink is replaced by a physical switch for patients who have minimum muscle control. The economic and dependence costs are much lower than with a human assistant, but it is not possible for patients who suffer from real Locked In Syndrome. The baud rate is still quite low and you usually can only have around four bits in each baud. For instance, two different buttons with single and double click. This depends on the severity of the lack of coordination though.
- **Brain-Computer Interface** – The previously mentioned BCI interfaces are not specifically targeted towards assistive technologies, as such no data was found related to their effectiveness as a text input mechanism. Another standard BCI system was evaluated in comparison – the P300. Through the P300 BCI system you take approximately 21seconds to write a single character with an 81% accuracy after a 30minute training session[21].
- **Electromyography** – No reliable data has been gathered relatively to the accuracy and stability of EMG devices. Yet, the cost of the only widely available EMG assistive commercial system is extremely high (approximately 11 000€)[45].
- **Eye Tracking Methods** - Even if we do not consider into the equation the dedicated software which will be developed, all of this methods have a much lower baud rate than that which is in principle allowed by real-time eye tracking methods like EOG.

Yet, it is imperative to acknowledge that all these technologies need not be placed against each other in competition. In specific cases they can (an most times should) be used in cooperation and complement each other for a more complete interface system that makes a wider use of all the skills available to the user.

4.2 Alternative eye tracking methods

Techniques based on reflected light

The most common eye gaze tracking method uses light (mainly infrared light) reflected by the eye (on the cornea, or further into the eye) which is then captured by a video capture device and processed by a computer. In order to highly increase the reliability of the system the light is usually in the spectrum of the infrared to reduce ambient lighting like lighbulbs and such which are usually in the visible range. Due to this, special video cameras or IR filters have to be applied. Many different video processing techniques and algorithms are available, the five which are mainly under research are:

- **Limbus Tracking –** The limbus is the boundary between the (normally white) sclera and the darker iris. This technique tracks the position and shape of the limbus relatively to the head. Therefore either the head must be held still or the apparatus must be fixed to the users head. It's only precise for horizontal movement, since the eyelids frequently cover the bottom and lower parts of the limbus.
- **Pupil Tracking** – This technique is similar to limbus tracking but tracks the border between the iris and the pupil. This border is usually sharper and not as easily covered by the eyelids although the contrast difference between the two regions is quite lower.
- **Corneal and Pupil Reflection Relationship** – When light is shone into the user's eye, several reflections occur in the boundaries between the lens and the cornea, they are called Purkinje Images. This image, together with the reflected light of the retina can be seen by a computer system as a very bright spot inside a less bright disc. The main difficulty of this method is having a good view of the eye. Lateral movement can place the image out of focus or outside the view of the camera.
- **Corneal Reflection and Eye Image Using an Artificial Neural Network** – By using a wide-angle camera the entire head of the user stays in the field of view. Through some more advanced set of video-processing algorithms the computer automatically detects where the eye is and returns a set of coordinates for the eye gaze. Although it usually only needs to be calibrated once for each user, the callibration can take up more than 30 minutes and the accuracy is not very good.
- **Purkinje Image Tracking –** Through the use of the previously mentioned Purkinje images you can also directly track the direction of the gaze through he Dual-Purkinje Image technique. This technique provides a high frequency and accuracy but the surrounding lighting must be heavily controlled.

Many projects have been developed in the recent past in the open source community involving techniques based on reflected light like the more popular openEyes project from Iowa State University or the very interesting Eyewriter by the Free Art and Technology (FAT), OpenFrameworks, the Graffiti Research Lab, and The Ebeling Group communities created to aid Tempt, a famous grafitter who recently acquired ALS. This system is built with the main objective of developing a platform which will allow Tempt to sketch grafittis on his computer only through eye communication.

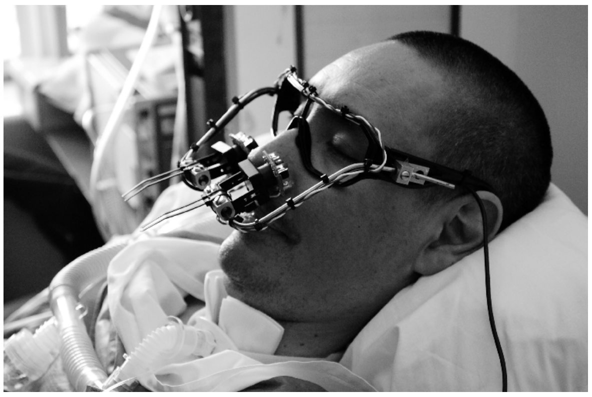


Fig. 4.3: Tempt using the Eyewriter open source eye-based HCI

There is also a Portuguese project dedicated to presenting a commercial assistive solution based on eye control. It is an interesting project named Magic Eye. These projects are, however, video-based and as such they differ from EOG, presenting the strengths and weaknesses that will be addressed when comparing these two technologies.

Techniques based on contact lenses

Through the use of special contact lenses it is possible to make quite accurate recordings of the direction of the gaze. Two methods have been studied. The first one being the engraving of one or more plane mirror surface on the lens and subsequently use one the reflection methods mentioned above and the second by implanting tiny induction coils into the lens which can then be recorded by high-frequency electromagnetic fields placed near the user's head.
Despite it's high accuracy this method is obviously highly intrusive not to mention the possible health hazards concerning high-frequency electromagnetic fields.

Comparison of the previous methods to EOG

On the downside, the measured EOG signals are subject to drift from several sources as has been discussed in chapter 2.7: changing skin resistance, electrode slippage or polarization, even a variable CRP due to light accommodation and level awareness. Noise pick-up from other electrical devices can be minimized by careful shielding, but action potentials of the other facial muscles can mask the desired signal. The most obvious shortcoming, uncorrectable by its very design, is the need for attachments directly on the user's face – five electrodes usually. Set-up is cumbersome, requires calibration for each individual and although actual discomfort is low, mental and physical awareness can be very high, creating a large long-term "annoyance factor". This method may be unacceptable to some subjects, especially if it is to be used in a social context.

On the positive side, EOG equipment is very cheap, easy to assemble with minimum access to a modest electronic workbench and can be used with glasses or contact lenses, unlike most other methods. The necessary apparatus does not obstruct the visual field, and is completely insensitive to head movement. The response time is very low compared to the reflection methods as well.

Although the use of a camera seems like it would have a lower acquisition cost since many of us already have webcams at house, either stand-alone or directly integrated in the computer, this is not true. In order to further improve the reliability of this technique a system involving infra-red cameras and/or bulky headgear with multiple sensors is required to increase reliability by removing head movements and other artefacts. As such the “annoyance factor” remains as much of a problem with video-based eye tracking as with EOG and at a higher cost (the cheaper systems begin at almost 2000€)[46].

Despite all it's current shortcomings video processing software has been growing at a large pace. In the past decade effective real-time face tracking algorithms seemed infeasible and today they are seamlessly integrated with multiple hardware like video cameras and cell phones. We can never tell what the power of computer processing and computation research might bring in the future but ultimately the strongest asset that EOG already undoubtedly provides is its solid and reliable experience which is what it all comes down to for the end-user.

4.3 Current research on EOG interface technology

Several studies show that EOG can be implemented as an easy to operate and reliable interface. Eye movement events detected in EOG signals such as saccades, fixations and blinks have been used to control robots and weelchairs[18], a wearable system for medical caregivers, a context-aware gaming acessory[19] among others. Some studies are also dedicated to the development of a less intrusive and more portable device[20].

Some commercial solutions exist as well, like those developed by H. Lusted of BioControl Systems with a moderately high cost.

	Button Triggers	BCI (P300)	EMG	Light Reflection				Eyewriter	Commercial EOG	Present Work
				Limbus Tracking	Pupil Tracking	Corneal/pu-pil relation	Dual Purkinje			
Response speed	Immediate	200-500 ms	250-500 ms	-	Some delay	-	Some delay	-	Real-time	350ms *
Baud-Rate	1 per button	4 **	5 **	pointer device	pointer device	pointer device	pointer device	pointer device	pointer device	9 **
Accuracy	100,00%	60%-90%	-							> 90%
Intrusive	None	High	Mild	Light	Light	Light	Light	Mild	Mild	Mild
Calibration necessary	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	***
Head movement sensitivity	None	Light	None	High	Light	-	Mild	-	None	None
Commercial cost	25 €	370 €	11.000 €	2.000€ – 35.000€	7.000€ - 32.000€	16.000€ - 86.0000€	27.000€ - 47.000€	145 €	-	20-30€

* - this delay is due to the "reflux" artefact. The device has the possibility of reacting real-time with proper improvements

** - the actual number of different orders which can be activated depends on the patterns which are trained (e.g. - the device could also recognize one eye blinks or double blinks)

*** - training was necessary for the present work, but there might be some improvements proposed in the final chapter through which it would not be necessary

Table 1: Comparison of equivalent assistive techniques

The previous table used information gathered from the following sources:

- Eye Controlled Media - Present and future state[26]
- Inclusive Technology website[47]
- BioControl Systems[48]
- "Online Detection of P300 and Error Potentials in a BCI Speller" [27]
- "On the Use of Electrooculogram for Efficient Human Computer Interfaces"[21]
- Makezine Blog [49]
- "A practical EMG-based human-computer interface for users with motor disabilities"[28]
- "Brain-Fed Switch a world first"[45]

In order to provide a true comparison between each system a proper test should be done with the same sample group while executing the same action under the same controlled environment. This is not an exhaustive comparison and serves only as a general reference about the strengths and weaknesses that each type of technology delivers. Some of the references are also slightly dated, but they are otherwise the most competitive values that I was able to find for each technique.

Quite often, particularly in the case of video-based technologies, these methods are used as pointing devices. Patmore et al. describes an EOG system that provides a pointing device for people with physical disabilities[29], Kaufman[16] also describes a system for controlling a cursor with success rates around 80%. However it is not the objective of the present work to create a pointing device.

4.4 Improving Through Simplicity

Stripping Down the Electronics

Having as one of the core design philosophies of this work the production of a technology that is simple, effective, cheap and provides a solid end-user experience, a very straightforward decision was that the eye input will not work as a direct pointing device with a continuous range of possible values but as a directional button switch activated through saccadic movement and blinking.

Recurring to an analogy based on common interfaces – the eyes will not work as a joystick but as an 8-directional pad. Built upon this premise, a system can be built with an extremely simple hardware which, supported by the right software, provides a much more solid experience then trying to directly control a mouse with your eyes. Immediately, as you are not directly controlling through your gaze, the Head Movement and Midas Touch issues mentioned on chapter 2.7 are solved. This decision brings, of course, many demands upon the design of the GUI that is to be developed. This issues will be addressed on chapter 5.3.

One might argue that if the same "reduction" was to be made to reflection based systems they would provide a similar experience while at the same time being less intrusive. That is not the case though, as we have already seen the kind of gear and associated cost that is necessary to provide a reliable equivalent to EOG.

A lighter and more reliable pattern detection

The previous works on EOG studied which presented details on their pattern detection algorithm were either over-simplistic or over-complicated.

The most basic method consisted on determining the saccade events through a simple threshold surpass detection. When saccades occur, signal peaks are produced and if the value which is being read is higher than a predefined threshold a saccade event is registered. Although extremely light, this algorithm is very unreliable, not suitable for diagonal saccades due to minor signal variations and prone to give many false positives. A simple electrode displacement or sweat artefact can shift the signal in a way which is interpreted by the algorithm as a saccade event.

Another, more complex and reliable algorithm based on the K-Nearest Neighbour algorithm is proposed by Usakli et al.[21]. This algorithm presents a very reliable pattern detection at the expense of high processing requirements. The values being read are stored in a buffer array which is permanently being compared to a set of predefined saccade signals. The following Euclidean distance formula was used:

$$L(x,y)=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2} \tag{4.1}$$

If the distance is lower than a determined threshold distance a saccade event is registered. The necessary calculation for this distance formula impose the processor a high processing speed as well as a memory requirement to store the data arrays larger than that which is allowed by a low cost processor. In this case they used 20 members (different pre-recorded signals for the same saccade movement) each with 251 samples. Using 9 different classes with values from a 10bit ADC (which produces a 2byte value) would require:

$$20\,members\times 251\,samples\times 9\,classes\times 2\,byte\ variable = 90360\,bytes \approx 88.24\,Kbytes \tag{4.2}$$

This is just for the permanent array storage. More space is still necessary for the program itself and many processor variables well as additional memory for the readings buffer. A memory requirement of around 100K bytes is obviously not a value intended for a simple low cost processor – the ATmega328 which is used in the current work, for instance, has 32Kbytes of Flash memory, 1Kbyte of EEPROM and 2Kbytes of RAM. The complete K-Nearest Neighbour is only viable because, in the work which proposes this method, the ADC transfers the signal directly to the computer to be processed. In the present work the signal processing is intended to be fully done on an independent processor. As such a modified algorithm is proposed in chapter 5.2.1.

Plug-and-play capabilities

The proposed device was designed to work “out-of-the-box” without any special need for individual user and/or setting configuration both from a hardware and a software perspective. A simple connection of the USB cable to the computer and a proper application of the electrodes to the user's forehead is sufficient.

Building a dedicated GUI

In order to deliver the user the best possible experience a specific visual interface which takes the input limitations in consideration must be designed. The design process of this interface is explained in depth in chapter 5.3.

Modular system

The device is designed from the ground up to be able to be split into its three independent parts and each one to be used individually expanding its longevity and ease of upgrade. This potential is further explained in the beginning of the next chapter.

CHAPTER V

The Developed Device

"Any gadget, even a big one (…), gets boring after a while. But a deepening of meaning is the most intense potential kind of adventure available to us"
Jaron Lanier, You are not a gadget, 2010

The actual implementation of the proposed device has been separated into three different stages which communicate unidirectionally.

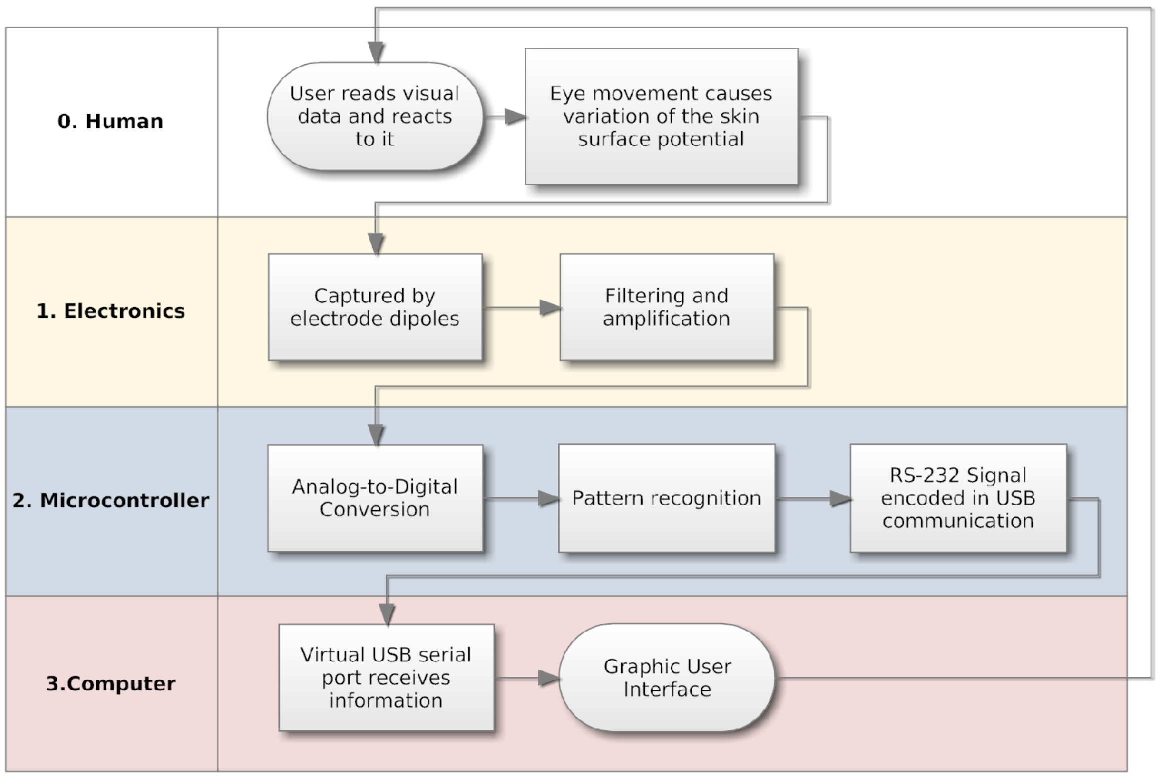

0. Human
User reads visual data and reacts to it
Eye movement causes variation of the skin surface potential
1. Electronics
Captured by electrode dipoles
Filtering and amplification
2. Microcontroller
Analog-to-Digital Conversion
Pattern recognition
RS-232 Signal encoded in USB communication
3.Computer
Virtual USB serial port receives information
Graphic User Interface

Fig. 5.1: Swimlane stepflow diagram – how to achieve a Human-Computer interface in three steps

For the first stage two pairs of ECG Ag/AgCl electrodes were applied to capture the skin surface potential variations with reference to a ground provided by a fifth electrode. This signal is transmitted via a twisted pair cable to an electronic circuit which subsequently filters and amplifies it. The filtering is composed by two stages – a high pass and a low pass. Each stage has independent properties and amplification gain but they can otherwise be perceived as one band pass filter whose main purposes are to remove DC-signal components and eliminate the 50Hz noise induced by the line (or mains) frequency.

The second stage occurs in an Atmega328 microcontroller which converts the incoming filtered and amplified analog signal to a sequence of digital values. In order to find recognized, pre-trained eye movements this values are processed through a simplified hybrid pattern recognition algorithm. If a relevant pattern is found, the USB controller in the Arduino board is used to transmit the respective data to the computer.

When eye saccade data is received by the computer through a virtual USB serial port it is fed into a dedicated custom-built Java GUI. The GUI is programmed in Processing, an open source programming language and integrated development envirorment (IDE) built upon the Java language with the objective of producing graphical output with a minimum effort. The GUI runs as a stand-alone multi-platform application which has been successfully tested in Windows 7, Ubuntu Linux 9.10 and Mac OS X.

A crucial goal was to design a completely modular and independent system which allows any of the developed stages to extend much further beyond the originally intended scope. Any of these modules can be used individually or partially integrated and are easily adjustable for different purposes:

- Stage 1 – With some minor alterations it can be used to capture and process biomedical signals similar to EOG like ECG, EMG, etc.
- Stage 1 and 2 – To control any device which is might be useful to interact with eight-directional saccades and blinking. Through a simple USB interface many applications could be achieved for this device as has been addressed in the third chapter's section Going Beyond.
- Stage 3 – Any situation where a discrete-8 directional GUI can be of use. Either as an assistive technology as well, based for instance on button-triggers instead of eye saccades, for disabled people who still have a minimum degree of motor control or as a mobile java application.
- etc.

5.1 Electronics – Analog Signal Processing

As has been addressed in chapter 4.4, for the sake of simplicity, only sudden variations of skin surface potential caused by saccadic eye movements are of interest. As such, an approximately "derivative" approach has been taken to the signal which immediately solves the very present problem of DC drifts. This involves the usage of a high-pass filter with a low cut-off frequency. The actual frequency range of the usable signal is a very narrow bandwidth of around 1Hz-35Hz. The skin surface differential potential is in the several hundreds of microvolts range. Such a small voltage is bound to be affected by the mains noise which is induced by the general-purpose alternating current electric power supply. This signal has a frequency known as mains frequency which is 50Hz for the majority of the world and 60Hz for North and Central America and a few other countries. Therefore a low-pass filter is necessary and a notch filter applied to this specific frequency was also taken into consideration.

The ADC of the ATMega328 being used has an input reading range of 0-5V therefore we will need to generate an output signal amplified for a maximum of approximately 3V peak-to-peak value. It is safer to design the gain for a 3V peak-to-peak instead of the maximum value of 5V peak-to-peak to insure that the limits are not exceeded, either due to imperfections on signal amplification and stability or to a user who has an out of the ordinary high skin surface potential. Some users do have a high CRP and/or higher conductance which can lead to a higher than average skin surface potential variation. In order to achieve this the signal must be amplified for a 4V peak-to peak value and an offset around the 2,5V mark. In "On the Use of Electrooculogram for Efficient Human Computer Interfaces"[21] the total linear gains used were 5332,8 for the horizontal channel and 7595,2 for the vertical channel. They do not specify, however, the final voltage obtained. Nevertheless this values were used as a starting reference.

In short, we need to design a circuit which contains:

- an instrumentation amplifier with a linear gain of approximately 5000-7500.
- a low pass filter (and possibly a notch filter) to cut general high frequency noise and, specifically, the 50Hz mains noise.
- a high-pass filter to cut "DC" drifts – cut-off frequency around 0.1Hz.
- signal offset around the 2,5V mark.

5.1.1 Preliminary circuit design

A number of different circuits and methods for signal filtering and amplifying were studied in order to obtain a satisfactory result. They did not, however, produce relevant results so no information regarding them will be presented.

First working version

After the unsuccessful simplistic approach of the previous circuits a more detailed and extensive method was adopted for the circuit design. A three stage instrumentation amplifier was designed from scratch, based on a standard design.

A differential amplifier is an electronic amplifier that multiplies the potential difference between two inputs by a constant factor. In the case of the present work it is used to multiply the difference in skin-surface potential between two points close to the eye. For this type of situations where great accuracy and stability is necessary both from a short-term and long-term perspective a specific differential amplifier is used – the instrumentation amplifier. By buffering the inputs the need for input impedance matching is eliminated and a circuit with a very low DC offset, low drift, low noise, very high open-loop gain, very high common-mode rejection ratio, and very high input impedance is produced.

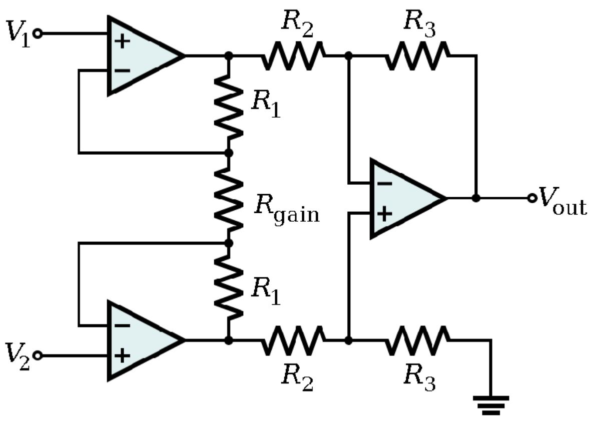


Fig. 5.2: Typical instrumentation amplifier schematic

For the proposed design, the first two stages are comprised of a typical instrumentation amplifier with the addition of a high-pass filter on the first stage and a low-pass filter on the second stage. This is achieved through the proper placement of capacitors on the typical instrumentation amplifier design. The third stage is a notch filter. The high pass filter was implemented as the first stage of the circuit to prevent saturation from occurring and clipping the signal.

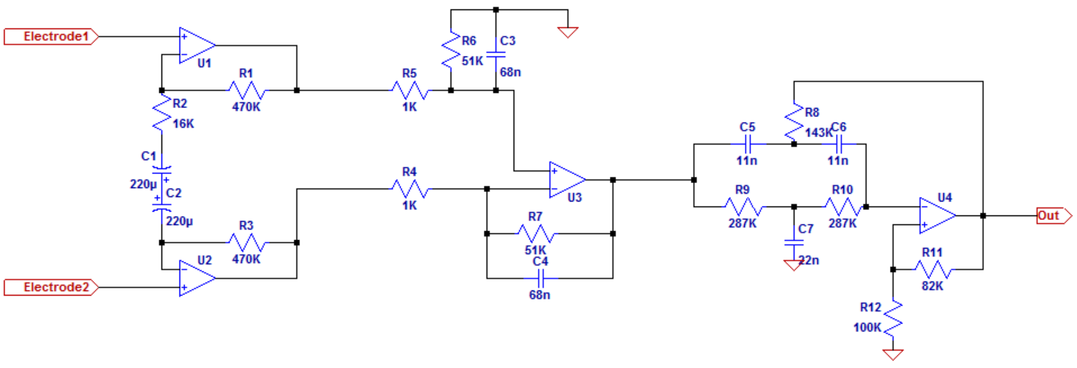


Fig. 5.3: The design of the first working version of the instrumentation amplifier (single channel)

The overall linear gain for test should be in the 5000-7000 range. As such the circuit was designed to provide a gain of 60 for the first stage, 50 for the second stage and 1.8 for the third stage resulting in a total gain of 5400. The gain value for the notch filter had to be, due to design constraints, relatively low. The gains for the other stages were divided approximately equally, having in mind component availability in the laboratory.

In order to simplify calculation work and to allow a faster switching and replacement of components during the testing phase, the component tuning was calculated with a custom Mathematica notebook.

Stage 1 – Buffered input differential high-pass filter

The goal for this first stage was to implement on the instrumentation amplifier a high pass filter that can cut any DC signal component. The cut-off frequency was initially set for 0.1Hz with a maximum linear gain of 60.

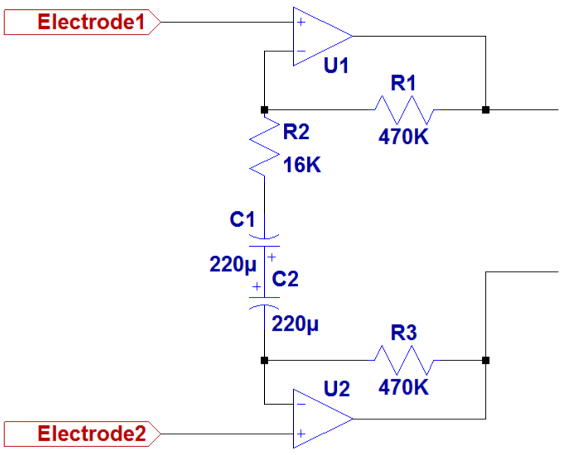


Fig. 5.4: Schematic for the first filtering and amplification stage

The transfer function for this circuit is given by the following expression:

$$\frac{V_O}{(V_2 - V_1)} = 1 + \frac{s(R_1 C_1 + R_3 C_1)}{s R_2 C_1 + 1} \tag{5.1}$$

Where V_0 is the potential difference between the two output ends, V_1 and V_2 are the voltages captured by electrode 1 and electrode 2 (respectively) and s is given by $2\pi f$. For an instrumentation amplifier to have gain which adequately translates the difference between its inputs, R_1 and R_3 must have an equal value. For the passband of this filter (high frequencies, $s \rightarrow \infty$) the gain can be given by:

$$G_{MAX} = 1 + \frac{2R_1}{R_2} \tag{5.2}$$

Applying the intended gain of 60 a relationship between R1 and R2 can be obtained:

$$60 = 1 + \frac{2R_1}{R_2} \Rightarrow R_1 = \frac{59}{2} R_2 \tag{5.3}$$

The cut-off frequency is that for which the gain is half of the maximum gain. Therefore the relationship between the resistors, the cut-off frequency and the capacitor can be obtained:

$$\frac{G_{MAX}}{2} = 1 + \frac{2sR_1C_1}{sR_2C_1 + 1} \Rightarrow C_1 = \frac{1}{s\left(\frac{2}{G_{MAX}/2 - 1} R_1 - R_2\right)} \tag{5.4}$$

According to the constraints imposed by functions 5.2 and 5.4 we are able to define the values through a system of two functions dependent on two constants (G_{MAX} and s) and three variables (R_1, R_3 and C_1). The chosen values were:

$$\begin{gathered} G_{MAX} = 60 \\ f_{cut} = 0.1\,Hz \Rightarrow s = 2\pi 0.1 = 0.2\pi \\ \\ R_1 = R_3 = 470\,k\Omega \\ R_2 = 15.667\,k\Omega \simeq 16\,k\Omega \\ C_1 = 95.040\,\mu F \simeq 110\,\mu F \end{gathered} \tag{5.5}$$

As has been shown in Fig 5.4 there was no 100μF or 110μF non-polarized capacitor present in the laboratory so two 220μF capacitors were used in series providing a 110μF capacity. After this values were chosen a Spice simulation was ran in order to test the circuit.

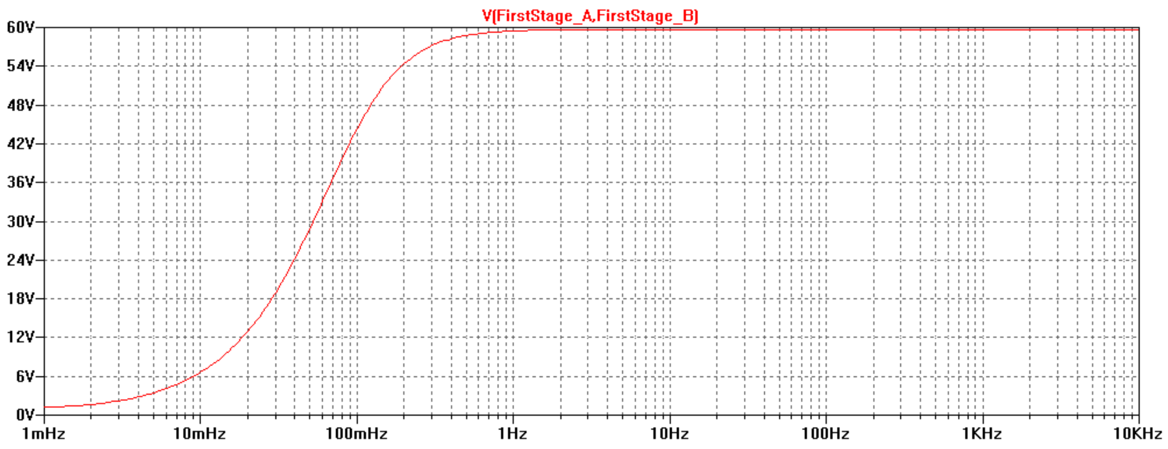


Fig. 5.5: Bode diagram spice simulation for the first filtering and amplification stage

As is observable in the previous image although the obtained linear gain is only slightly lower than expected (G_{MAX}=59.7139≈60), the compromise taken regarding component values, especially in the capacitor, induced a visible shift in the cut-off frequency (f_{cut}≈0,06Hz). At this point the shift was not considered very relevant which turned out to be a mistake as will later be discussed.

Stage 2 – Differential to single-ended low-pass filter

The goal for this second stage was to implement on the instrumentation amplifier a low pass filter that can cut high-frequency noise, if possible the specific the 50Hz mains noise. The cut-off frequency was initially set for 50Hz with a maximum linear gain of 50.

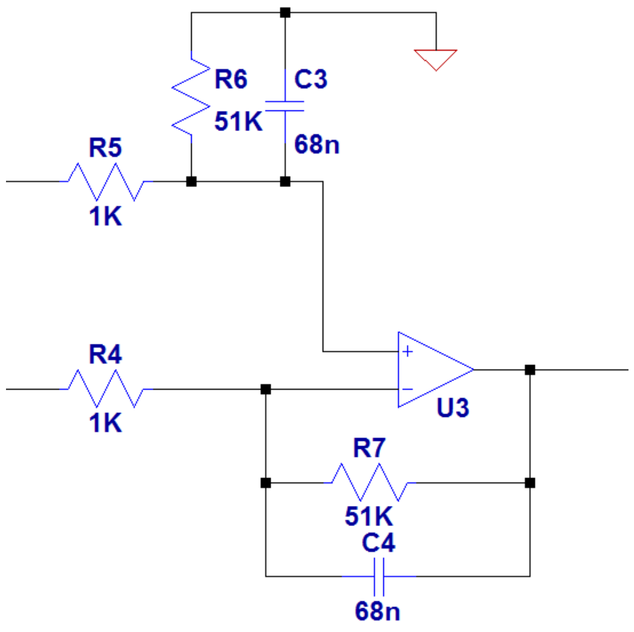


Fig. 5.6: Schematic for the second filtering and amplification stage

This stage of the circuit was analysed in two parts. For first part (G_{1MAX}) the voltage is only applied to the inverting input (v_{in1}) while the non-inverting input is grounded. For the second part (G_{2MAX}) the opposite procedure is done - the voltage is only applied to the non-inverting input (v_{in2}) while the inverting input is grounded. As s is solely dependent on the cut-off frequency, which is the same for both parts, the same value will be used for both expressions.

$$f_{cut1}=f_{cut2}=50\,Hz \;\Rightarrow\; s_1=s_2=s \tag{5.6}$$

The transfer function for this circuit can therefore be given by the following expressions:

$$\begin{aligned} \frac{v_{out1}}{v_{in1}}&=-\frac{\frac{1}{R_4C_4}}{s+\frac{1}{R_7C_4}} \\ \frac{v_{out2}}{v_{in2}}&=\frac{\frac{1}{R_5C_3}}{s+\frac{R_5+R_6}{R_5R_6C_3}}\left(1+\frac{\frac{1}{R_4C_4}}{s_1+\frac{1}{R_7C_4}}\right) \end{aligned} \tag{5.7}$$

In order to simplify the analysis for this stage of the circuit only the gain for the passband was taken into consideration, neglecting the capacitor contribution. Therefore:

$$\begin{aligned} G_{1MAX}&=\frac{R_7}{R_4} \\ G_{2MAX}&=\frac{R_6}{R_5+R_6}\left(1+\frac{R_7}{R_4}\right) \end{aligned} \tag{5.8}$$

To obtain a functional symmetric differential amplifier the gains for both parts as well as the symmetric components values must be the same:

$$G_{1MAX}=G_{2MAX} \;\wedge\; R_4=R_5,\; R_6=R_7,\; C_3=C_4 \tag{5.9}$$

For a low-pass filter the cut-off frequency is given by:

$$f_{cut}=\frac{1}{R_6C_3}=\frac{1}{R_7C_4} \tag{5.10}$$

To solve this system of two equations (based on equations 5.8, 5.9 and 5.10) dependent on two constants (G_{MAX} and f_{cut}) and three variables (R_4=R_5, R_6=R_7 and C_3=C_4) the following values were chosen:

$$\begin{gathered} G_{MAX}=50 \\ f_{cut}=50\,Hz \\ \\ R_4=R_5=1\,k\Omega \\ R_6=R_7=50\,k\Omega\simeq 51\,k\Omega \\ C_3=C_4=63.662\,nF\simeq 68\,nF \end{gathered} \tag{5.11}$$

These approximate values were chosen dependent on component availability in the laboratory. A Spice simulation was ran with the presented values.

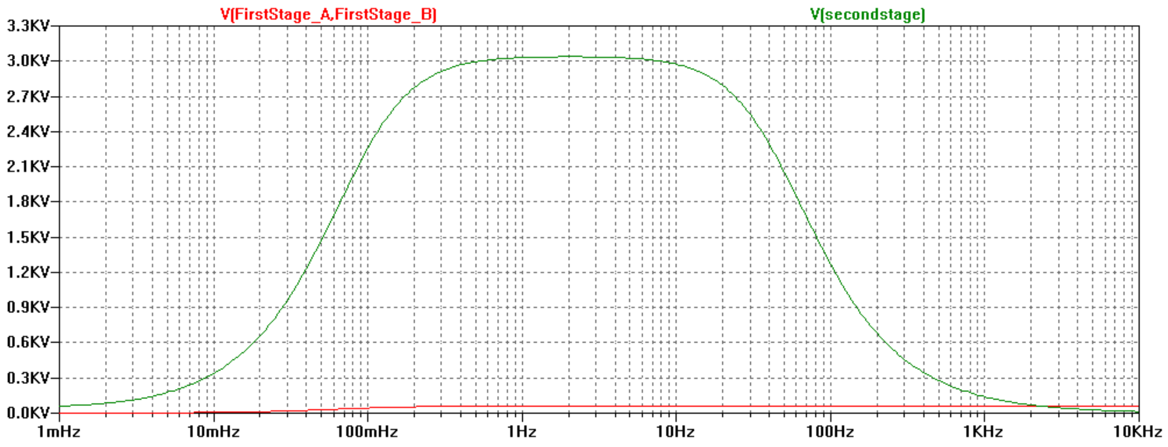


Fig. 5.7: Bode diagram spice simulation for the first and second filtering and amplification stages

The frequency response behaves according to expected. The maximum linear gain achieved was 3037.86 which is pretty closed to the 3000 which were calculated. The cut-off frequency is once again considerably shifted (80Hz) as the capacitor value variations produce a big difference in the circuit's frequency response. At this point this was considered to be of minor importance, given the inclusion of the notch filter.

Stage 3 – Notch filter with gain (Active Twin-T design)

The goal for this third stage was to add to the end of the instrumentation amplifier a notch filter specifically to cut the 50Hz mains noise which is the main source of noise for the proposed device. A notch filter is a band-stop filter with a very narrow stopband (high Q factor) usually required in applications where there is a source of noise in a very well defined frequency spectrum. An active Twin-T filter design was used as shown below.

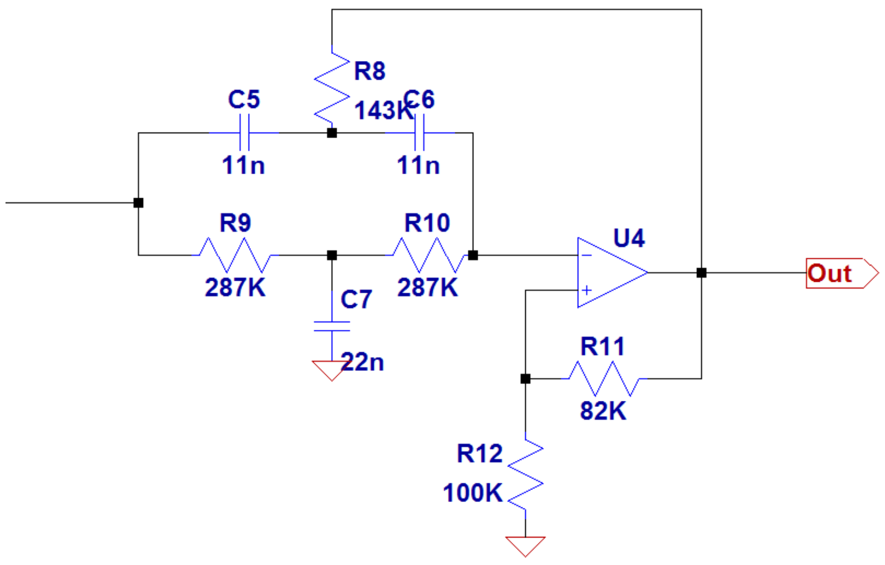


Fig. 5.8: Schematic for the third and final filtering and amplification stage

This filter which was obtained in TI's Op Amps for Everyone[30] has a very simple component tuning as the components have very direct relations between them:

$$C_5 = C_6 = \frac{C_7}{2} = C\,,\quad R_9 = R_{10} = 2\mathrm{R}_8 = R \tag{5.12}$$

Being the case, for the subsequent expressions the nomenclature C and R will be used as is presented above. The filter parameters obey to the following set of equations:

$$\begin{aligned} f_m &= \frac{1}{2\pi R C} \\ G &= 1 + \frac{R_{11}}{R_{12}} \\ A_0 &= G \\ Q &= \frac{1}{2}(2-G) \end{aligned} \tag{5.13}$$

For which:

- f_m is the mid-frequency expressed in Hz
- G is the inner gain expressed in V/V
- A0 is the passband gain expressed in V/V
- Q is the rejection quality

The twin-T circuit has the advantage that the quality factor (Q) can be varied via the inner gain (G) without modifying the mid frequency (f_m). They cannot, however be varied independently. This does not cause a serious problem as the rejection quality is not of great importance for this work. The parameters which must be imposed to the circuit are the mid-frequency which is meant to be 50Hz and the gain which should be slightly lower than 2. The following constraints were therefore imposed to the previous set of equations:

$$\begin{gathered} f_m = 50\,Hz \\ G = 1.8 \end{gathered} \tag{5.14}$$

After solving the parameter equations the following values were obtained:

$$\begin{gathered} C = C_5 = C_6 = 11\text{nF} \\ C_7 = 2C = 22\text{nF} \\ R = R_9 = R_{10} = 289\,k\Omega \\ R_8 = \frac{R}{2} = 143\,k\Omega \\ R_{11} = 82\,k\Omega \\ R_{12} = 100\,k\Omega \end{gathered} \tag{5.15}$$

This values where used for the Spice simulation direct from calculation. No approximation due to component laboratory availability had to be done for it became quite obvious after the assembling of the first two stages of the circuit that this stage would not be necessary, as will be addressed in the next chapter.

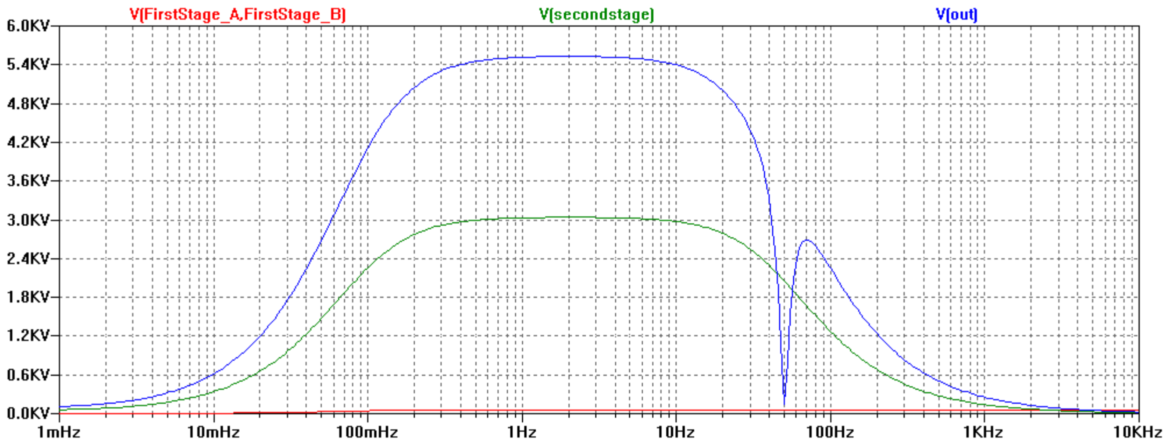


Fig. 5.9: Bode diagram spice simulation for the three stages of the instrumentation amplifier

Despite the fact that experience informed us that this third stage is not necessary, the simulation proves that it had been properly designed from a theoretical point of view. The mid-frequency is quite visibly in the 50Hz with a very pronounced attenuation and the overall maximum linear gain is 5528,37. If we take into consideration that the gain for the previous stage was 3037,86 this shows that this final stage delivers a gain of approximately 1,82 which is very close to the expected value of 1,8.

5.1.2 Preliminary circuit test and improvement

For the preliminary circuit no special care was taken when selecting the components regarding size, tolerance, material, etc. The criterion of choice was simply their availability in the laboratory. The opamp used was a TL084 JFET-Input operational amplifier. These come in DIP-14 packages and contain 4 opamps each so only one chip is necessary per channel.

As previously addressed, the first obvious realization upon assembling and testing the circuit was that there was actually no need for a notch filter. Through the correct implementation of the low-pass filter, working in conjunction with a body reference electrode and a twisted pair cable, the 50Hz noise is reduced more than necessary. The cut-off frequency of the low pass filter can even be lowered given the narrow range of frequencies (approximately 1Hz-35Hz) which is of use for the device.

The second interesting phenomenon was a slight fluctuation around the 130mHz-145mHz with an amplitude of approximately 0.2V. The presence of this fluctuation was further increased due to the previously mentioned shift in the cut-off frequency of the high-pass filter. This was easily corrected by adjusting the high pass filter frequency from 100mHz to 1Hz which causes almost no interference to our signal as we have seen before that we are working in the 1Hz-35Hz range. There is some gain loss at the lower frequencies, but the outcome of this adjustment is positive.
As expected, the horizontal and vertical signal channels should have different gains. The highest peak is given on the vertical channel when blinking. Some hands-on testing and fine-tuning was done during this stage of the experimental work in order to obtain the best possible result. The following were some of the images obtained during this period.

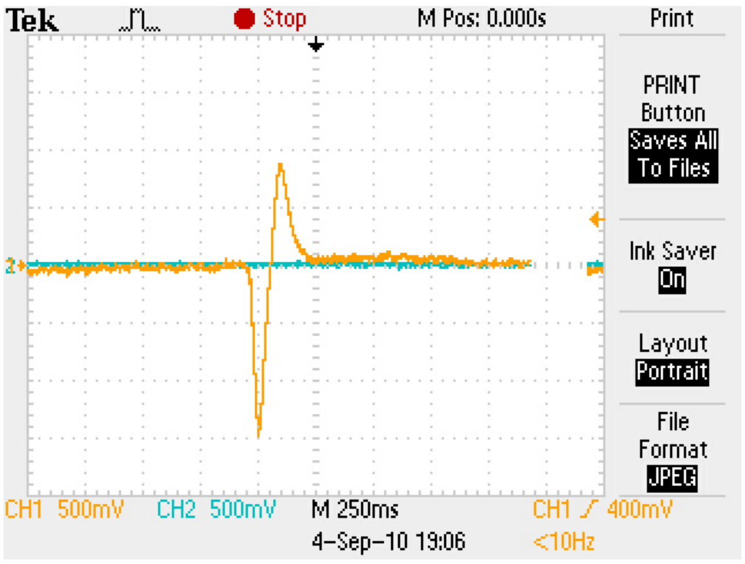


Fig. 5.10: First oscilloscope image collected

For first picture the vertical variation of the skin-surface potential caused by an eye blink was recorded. The initial component tuning was used. It has a very pronounced peak, as was expected. The quick time span of the signal is due to the high-velocity of an eye blink.

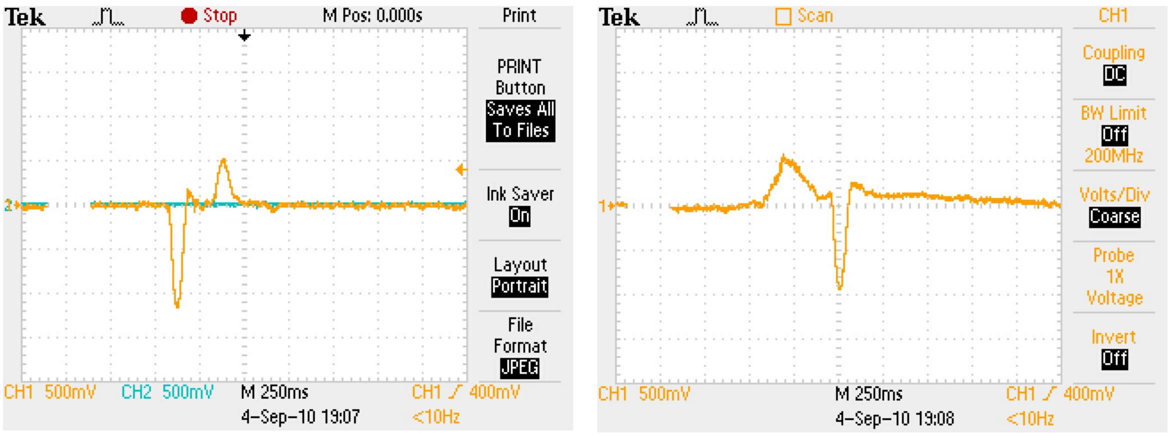


Fig. 5.11: Signals produced by an "UP" eye movement (on the left) and "DOWN" eye movement (on the right) on the vertical channel

In this recording of an "UP" and a "DOWN" saccades it is curious to note that the electrodes were inverted relatively to common EOG practice – the up movement delivering a negative signal and the down movement delivering a positive signal.

In the next picture the signal produced on the vertical channel by a sequence of three blinks is displayed. The signal coherence is quite obvious. The three instances of the eye blink produced a very similar result. This coherence is the reason why it will be possible to detect each individual pattern.

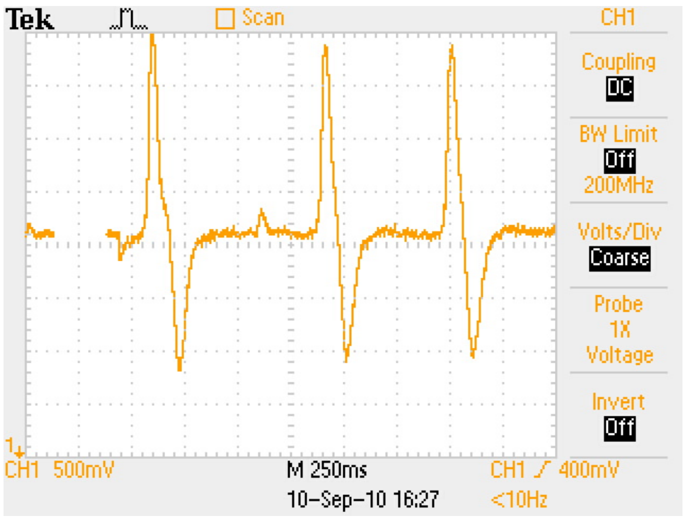


Fig. 5.12: Signal obtained in the vertical channel for three blink in a row

It is also important to notice that the signals shown above are all composed by two approximately symmetric peaks. The second one is actually a harmless artefact caused by the high pass filter stage and will be henceforth called “reflux”. The negative peaks in Fig 5.12 have a higher amplitude than the positive peaks due to a wrong resistor calibration on the first stage of the amplifier.

This first images were all collected during a period of the research process when an electronics workbench was being used to power the circuit. This workbench was providing a power supply of 10Vpp. When it was attempted to replace the circuit power supply with the supply from the Arduino board the circuit stopped responding.

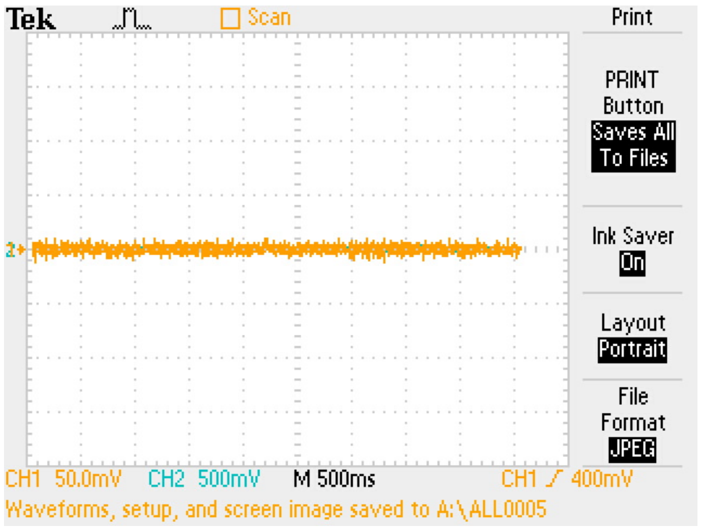


Fig. 5.13: The operational amplifier stopped responding after switching to a lower power supply

This was due to the power supply provided by the Arduino board being 0-5V while the TL084 has a minimum necessary power supply (Vs_{min}) of $7V_{pp}$. An LM324 low power quad opamp with a Vsmin of 3Vpp which was available in the laboratory was tested and, while it did support the low power supply, it was not able to produce viable results. Especially on the first stage of the instrumentation amplifier, very sensitive equipment must be used given the amazingly low order of magnitude of the signal that is being worked. Particular attention must be given to the input bias current (I_{IB}) of the operational amplifier. For instance, the TL084 used has a typical I_{IB} of 30pA and a maximum of 400pA. The LM324, although it is also presented as having a low I_{IB}, has a typical value of 45nA and a maximum value of 100nA. It is, in fact, around three orders of magnitude higher than the previous one.

Upon this situation two options were considered. The first one was to use a MAX660 CMOS Volt Converter which delivers a -5V output given a 0-5V input. This would allow the usage of the previously tested TL084 with the 0-5V power supply of the Arduino board. The second one was to find an opamp that could be powered with a supply of 5Vpp and have an extremely low input bias current while at the same time having a reduced price, in order to maintain the goal of a low-cost device. This second approach was considered to be much more elegant and some opamps were researched, acquired and tested.

		TL084 *	**LM324**	**TLC274**	**TLC2254**	**UNIT**
V_{DD}		7 to 36	3V to 30V	3V to 16V	4,4V to 16V	V
I_{DD} (4 amps)	**Typical**	5,6	0,7	2,7	0,28	mA
	Maximum	11,2	1,2	6,4	0,5	mA
I_{IO} (at 25ºC)	**Typical**	5	2.000	0,1	0,5	pA
	Maximum	100	30.000	60	60	pA
I_{IB} (at 25ºC)	**Typical**	30	20.000	0,6	60	pA
	Maximum	200	150.000	60	100	pA
Slew Rate (at unity gain) **	**Typical**	8	0,4	3,6	0,12	V/µs
	Maximum	13	-	-	-	V/µs
Bandwidth (unity gain)		3	1,3	1,7	0,2	MHz
Cost (1 unit)		0,45	0,25	0,75	1,37	€

* - All the results were obtained at 25ºC with Vdd=5V except for the TL084 which was had a supply of +-15V

** - The slew rates were obtained with slightly different parameters for each operational amplifiers, but they serve as a indicative values

Table 2: Comparison of operational amplifiers

Prices are only indicative since they vary a great deal according to minor specificities and packages. The price and properties of the cheapest DIP package was presented in the table above.

The one which, after extensive testing, definitely provided the best results was the TLC274 giving a clear signal with a 0-5V power supply and at a reasonably low cost. TLC279 is a version of the TLC274 which has a higher sensitivity to low input voltages. This could provide better results than the TLC274, yet it costs more than its triple and no DIP version was available for ordering during the research stage. Although TLC274 proved to be effective, it would be interesting to test its upgraded version for relevant improvements. Last but not least, the 2.5V signal offset had to be designed. This was achieved through the implementation of a virtual ground at 2.5V. This potential is connected to the head reference electrode as well as the (now virtual) ground reference of the final amplification stage.

5.1.3 Prototype circuit design

After all corrections had been taken into consideration an improved circuit was designed.

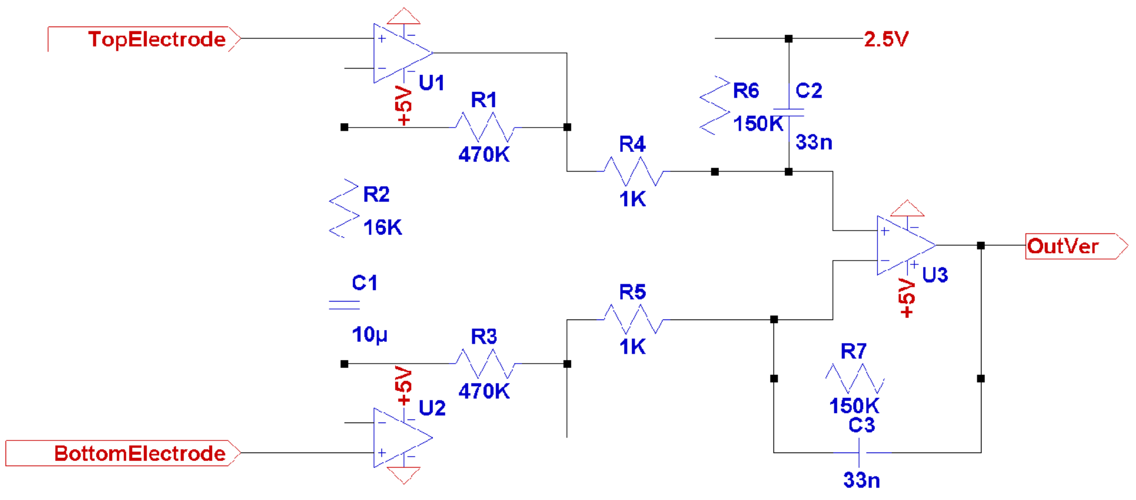


Fig. 5.14: Diagram for the vertical instrumentation amplifier channel

The only difference between the vertical and horizontal channels, besides the electrode placement, is the gain for the second stage of the instrumentation amplifier. This gain is set through R13, R14, C5 and C6, as seen in Fig. 5.15.

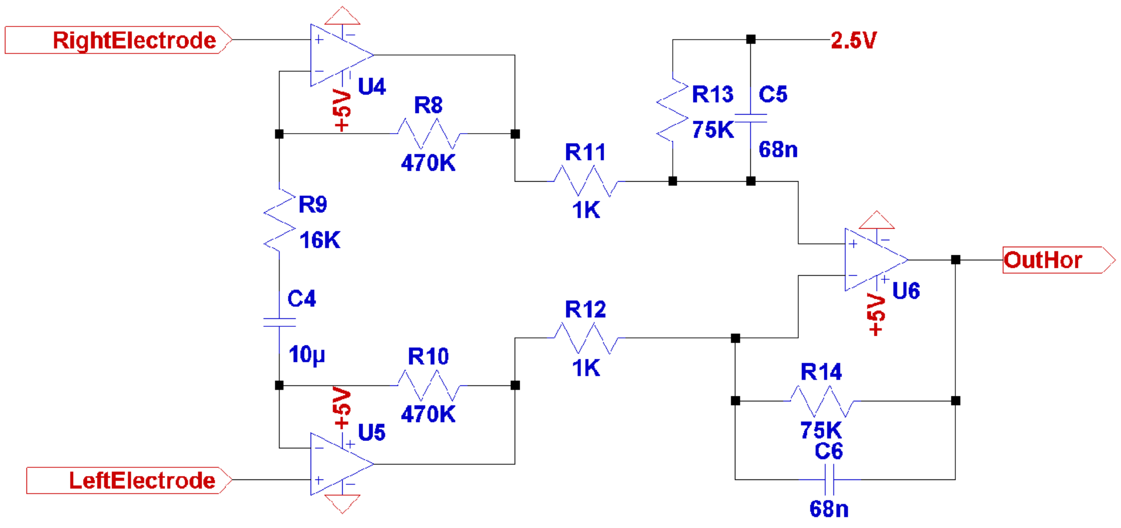


Fig. 5.15: Diagram for the vertical instrumentation amplifier channel

For the signal offset part a capacitive tension divider with a voltage follower was designed. The voltage divider is directly connected to the Arduino board power supply. This supply is USB powered which, according to USB specification, means a voltage of 5V±5% and a current draw of one to five unit loads. Each unit load corresponds to 100mA in USB1.X and USB2.0 and 150mA in USB3.0. The Arduino board, as well as the circuit, is expected to have a much lower current draw than this maximum value. The voltage divider delivers a 5V, 2.5V and ground outputs.

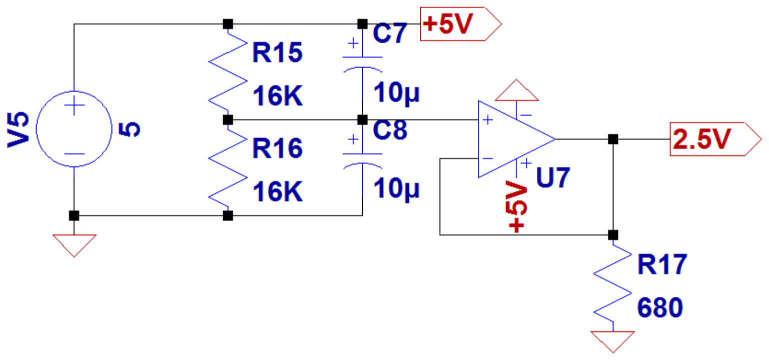


Fig. 5.16: Voltage divider schematic

The extra operational amplifier used in the voltage follower does not present any further component requirements to the PCB, given that two TLC274 microchips which were already being used still had one available opamp each. One of this was used to produce the voltage divider.

Below a simulation for the final frequency response of both the horizontal and vertical channels is displayed.

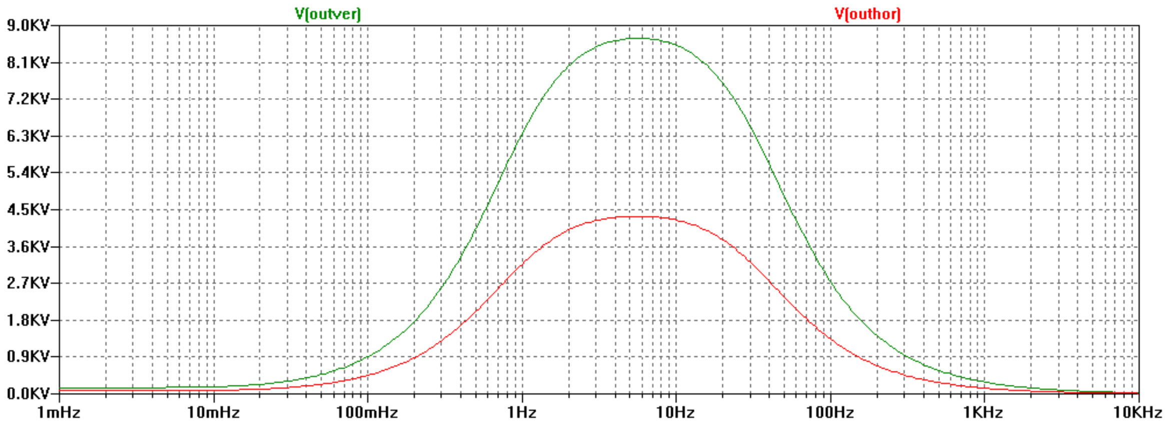
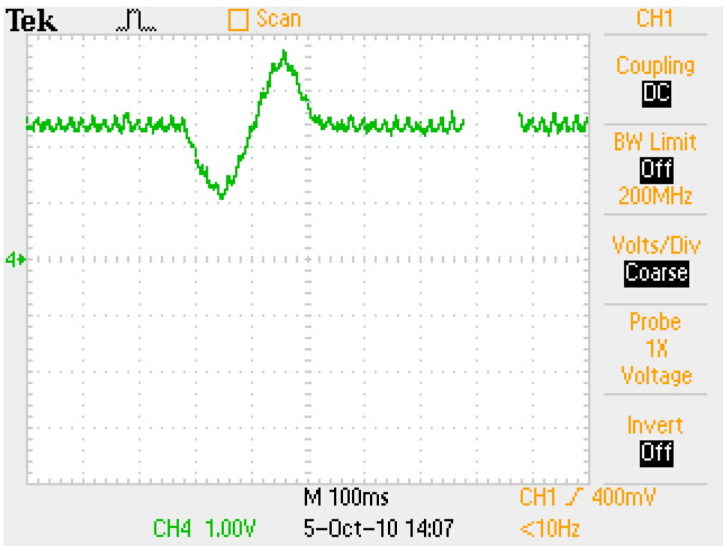


Fig. 5.17: Spice simulation of the Bode diagram of the prototype design's outputs

5.1.4 Prototype circuit test

After the circuit had been properly designed and simulated through Spice software it was implemented in a breadboard and tested to ascertain further necessary improvements. In the following image the vertical signal for an UP eye saccade is already centered around the 2,5V mark and correctly displays an amplitude of almost 3Vpp, which is the amplitude that the slightly more powerful eye blink signal should be able to reach, according to the proposed design. The slightly higher gain compared to the previous images is due to inherent differences between the previous operational amplifier which was being used (TL084) and the current and final operational amplifier (TLC274).



CH1
Coupling
DC
BW Limit
Off
200MHz
Volts/Div
Coarse
Probe
1X
Voltage
Invert
Off

Fig. 5.18: Signal captured by the vertical channel of the prototype circuit for an UP saccade

Sometimes, mostly due to non perfect electrodes, the offset will not result in an accurate centring around the 2.5V potential. If there is a minor shift in this offset it can be corrected by the microprocessor, as explained in chapter 5.2.2.

However, this offset deviation might be enough to drive the operational amplifier into the saturation zone thus rendering the signal unusable. Saturation occurs when the output of the amplifier reaches values close to the power supply voltages. As has been mentioned before, the device was designed to provide a signal amplitude of approximately 3Vpp centred around the 2,5V mark leading to values between 1V and 4V. This means that the systems is operational for hardware offset errors no higher than 1V. For a positive offset error this means centring the signal around 3,5V and negative centring the signal around 1,5V which might produce peaks outside of the 0-5V range which is the working spectrum for the operational amplifier when using a 5V power supply.

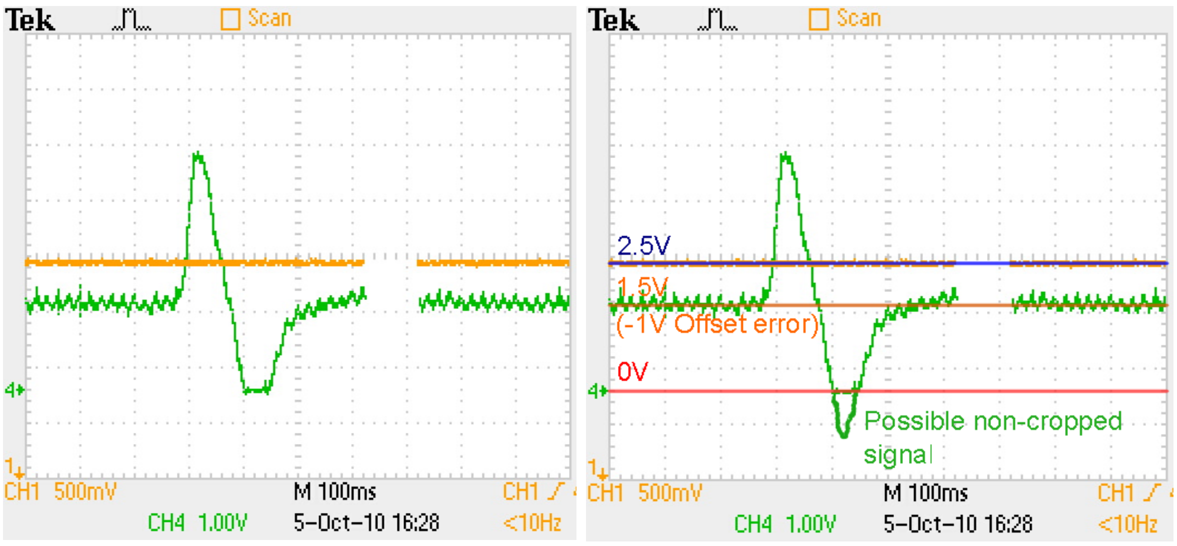


Fig. 5.19: Saturated signal captured during gain test stage

Although obtained still in component tuning stages which were returning a signal amplitude of almost 5Vpp for the blink signal (instead of the desired 3Vpp), Fig 5.19 is a good example of saturation due to offset deviation. The operation amplifier's output voltage would have reached values below 0V had it not crop the signal due to saturation. A small plateau is visible in the lower peak creating a signal pattern which could be rendered unrecognisable for the microprocessor.

5.1.5 PCB implementation

The prototype PCB was designed as a piggyback for an Arduino Duemilanove board. The Arduino platform has been chosen due to its wide-spread use, cheap cost, ease of programming, preparation for USB communication, preparation for communication with the Processing platform, my previous knowledge of it and the usage of the ATMega328 processor (the benefits of this processor will be addressed in chapter 5.2). Using the board also solves many immediate problems like power supply.

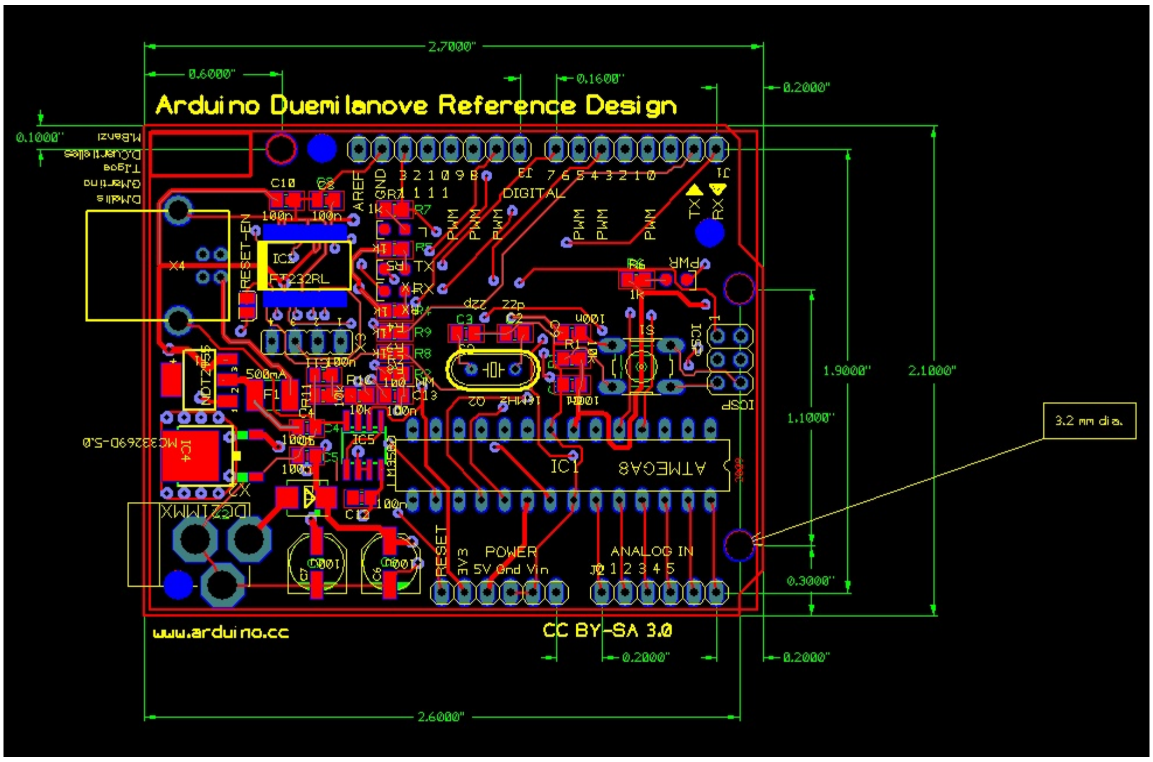


Fig. 5.20: Arduino Duemilanove PCB layout

The piggyback was designed to have four contact points with the Arduino board. Two inputs which are the 5V power supply and the ground as well as two outputs. The outputs are the horizontal and vertical signals connected to Arduino's ANALOG_IN pins 0 and 1.

Taken into account the dimensions of the Arduino Duemilanove board, the prototype board was designed as presented in Fig. 5.21. Practically all of the components have a 0805 SMD package. This package was chosen due to its practical size relatively to the board dimensions, ease of soldering and high availability. The material selected was ceramic and with a low tolerance. The only components to have a different package were the 10uF capacitors. During the design phase it was believed that 10uF SMD capacitors would greatly increase the board's production cost due to their higher than average price. Although after more intensive research it was found not to be true and some of the capacitors have actually been replaced by SMD ones with a 0603 package which were available in the laboratory. The copper traces have a 20mil width. The pads have 55X50mil for the components and 50X50mil or a 50mil diameter for pin-through. The internal diameter for pin-through holes is 28mil. The operational amplifiers were kept in their previously used and tested 14-DIP package. Opamps are relatively sensitive components and this makes it easier to replace them in a prototype board which is likely to suffer further experimentation.

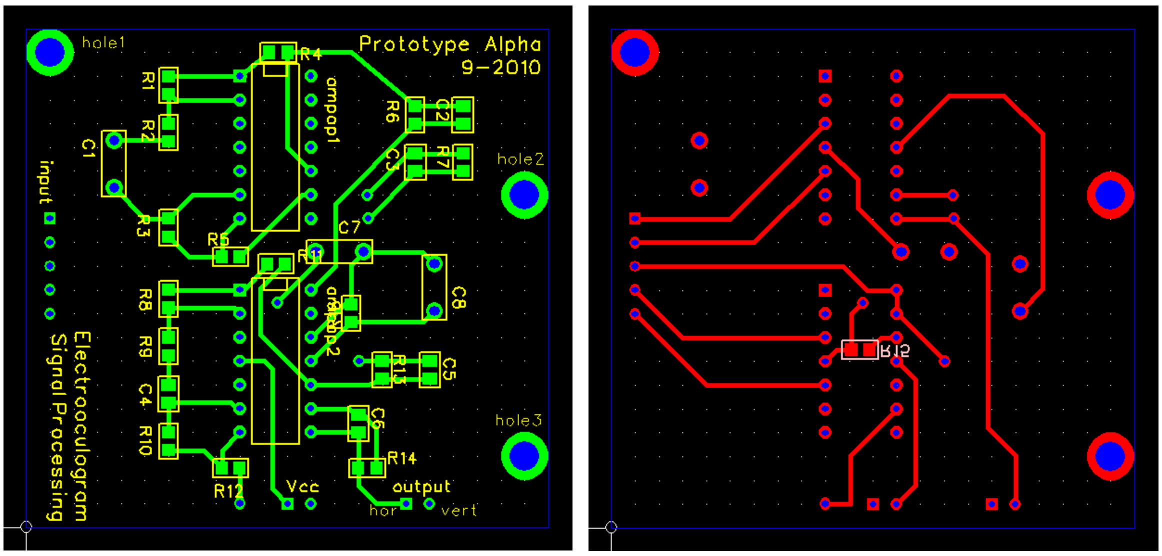


Fig. 5.21: PCB schematic top and bottom layers

For reasons which are not yet fully understood, when the breadboard circuit was translated into the PCB the gain from the instrumentation amplifier behaved differently. After much trial and error it was discovered halving the gain by switching the R4 and R5 1KΩ resistors that were previously used to 2KΩ. This provides an approximate maximum total gain of 2170 for the horizontal channel and 4350 for the vertical channel.

After the board had been printed a number of issues were found. Although they were corrected by hand and the PCB turned out to function very satisfactorily, in a future version they should be addressed:

- The board is 100mil shifted to the "left" (relatively to the orientation of Fig. 5.20 and Fig. 5.21) . This does not bring about any serious issues, but the large holes (identified as hole 1, hole 2 and hole 3) do not coincide with the ones in the Arduino board.
- The holes for the piggyback-arduino connection pins should be slightly larger.
- The 10uF capacitors should all be in SMD packages.

- The traces and pads could have been planned in a way which better suits the task of soldering.
- There is a trace which does a 90 degree angle.
- The reference electrode should be connected to a ground plane to reduce noise.
- A better pin connector could have been used to connect the electrode cables to the board.
- In order to reduce virtual ground noise a resistor with a low resistance should be placed between the 2.5V output and ground.

The complete working and corrected piggy-back version of the PCB is shown below.

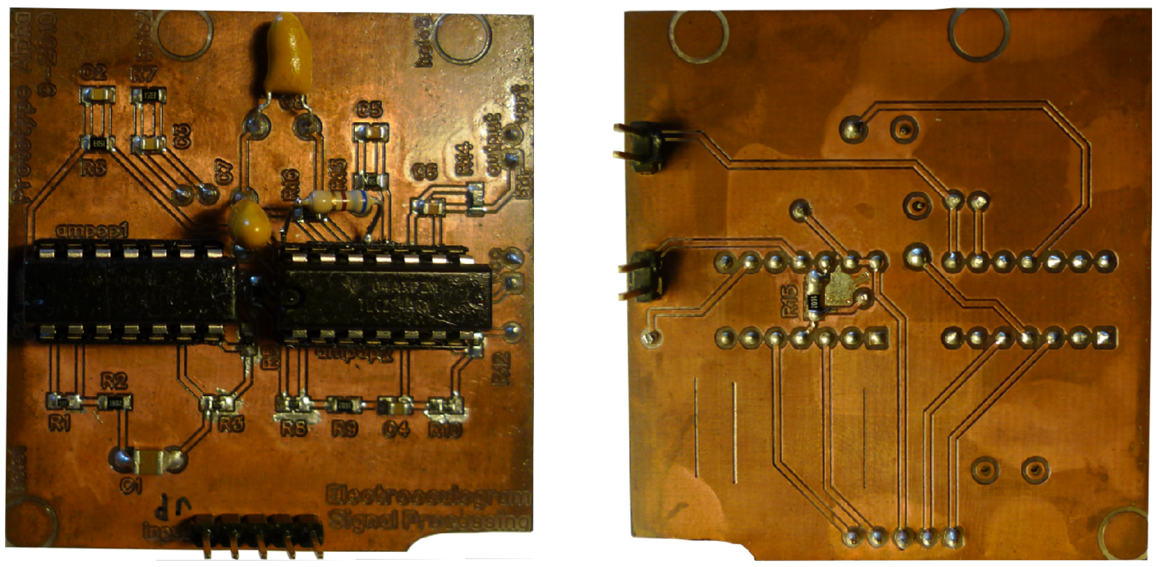


Fig. 5.22: Photo of the printed and assembled PCB top and bottom layers

The small dent which can be seen in the bottom of the board had to be carved due to the previously mentioned 100mil error which occurred when designing the PCB relatively to the Arduino Board. As can be understood more easily with the aid of Fig 5.23, the PCB was colliding with the external power input of the Arduino board causing them not to fit properly.

In a final, possibly commercial, incarnation of the PCB instead of a piggyback approach a stand-alone version would be desirable. This would greatly lower production costs as the Arduino board is easily more than half of the cost of the prototype version. The issue of costs is further explored in chapter 6.2. Besides the gain in production costs it would also increase its portability. This is not a matter of concern when applying this technology as an assistive method for physically disabled people given their low mobility but, as has been addressed many times before, this technology has the reach to expand much further than this single application. Any of the microchips used (the ATMega328 and the TLC274), which are by far the largest components in the current board, are available in more compact SMD packages. The ATMega328 has TQFP and MLF packages and the TLC274 has an FK package. The PCB board assembled as a piggy-back to the Arduino Duemilanove is shown below.

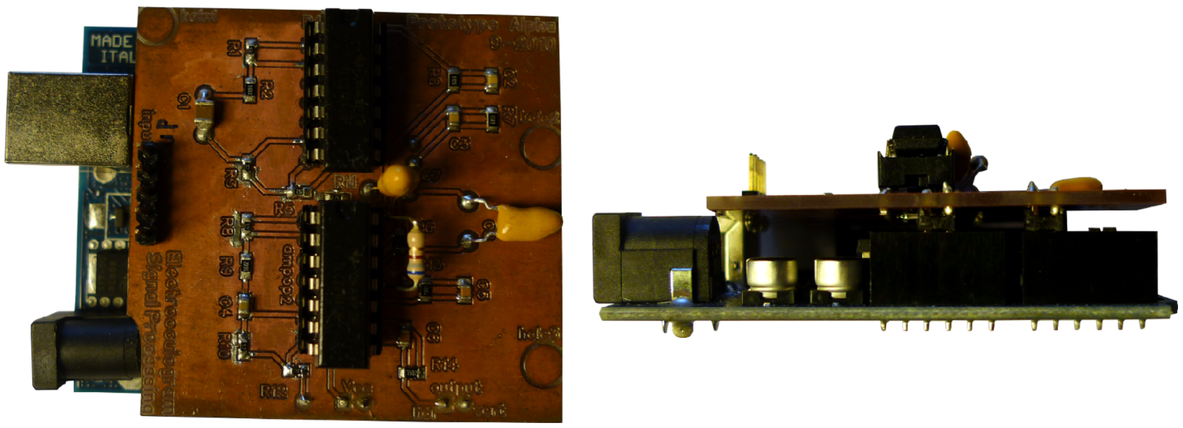


Fig. 5.23: Photo of the top and side vews from the PCB connected to the Arduino Board

The current device version based on a piggy-back format is already pocket sized (approximately 7cm long, 5.3cm wide and 2.5cm high) but if it were to be designed in a single compact board it could easily fit into something as small as a USB pen or a 9V battery.

5.2 Microcontroller - ADC and Pattern Detection Algorithms

Arduino is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. The microcontroller on the board is programmed using the Arduino programming language which is based on C/C++ and the Arduino processing environment which is based on Processing. This platform was chosen due to the following reasons:

- Very inexpensive hardware module
- Compatible with the cheap and widely used ATmega8, Atmega168 and ATmega328 family of processors
- Free software download
- Simple and clear yet flexible programming environment
- Open source and extensible software with a great number of user contributed libraries
- Open source and extensible hardware module which has plans published under a Creative Commons license
- Easy communication with Processing platform
- My experience with it

The specific Arduino board model used was the Duemilanove and ATMega328 was the chosen processor. This board was chosen because it was the most recent standard release upon the beginning of this work. Although a more recent version has come out in the meanwhile (Arduino UNO) it does not present updates that would justify substituting the previous one. The ATMega328, besides being the default processor which is packaged with this board, is the most powerful from the ATMega family of processors compatible with this Arduino board, therefore ensuring an adequate amount of processing power and storing memory at a cost which is not relevantly higher than the ATMega8 and ATmega168. For large amounts (over 100) the ATMega 328 sells costs around 2,75€/unit while the cheaper version of the same family, the ATMega8, costs around 2€[50]. This prices are only indicative since actual price slightly vary according to package type. Other Arduino boards offer support for the ATmega2560 processor. After some evaluation the usage of this more powerful processors was not considered necessary as the ATmega328 is capable enough to get the job done.

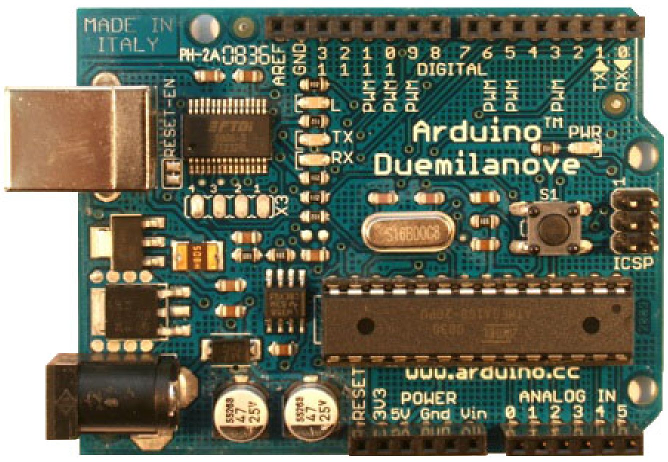


Fig. 5.24: Arduino Duemilanove board

The maximum length and width of the Duemilanove PCB are 2.7 and 2.1 inches (6.858 and 5.334 centimetres) respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Our piggy-back board was design to match these dimensions as well as the screw holes.

The PCB interacts with the Arduino board solely through 4 connectors. Two of them are dedicated to powering the PCB (Arduino pins GND and 5V) and the two others are dedicated to transmitting the filtered and amplified horizontal and vertical signals from the PCB to the ATMega328 (Arduino pins ANALOG IN 0 and ANALOG IN 1).

An FTDI FT232RL on the board channels the serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual COM port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer. This serial communication was used to transmit the eye saccade event data.

The ATmega328 microcontroller is a high performance, low power AVR 8-bit microcontroller. The 28 PDIP package was used.

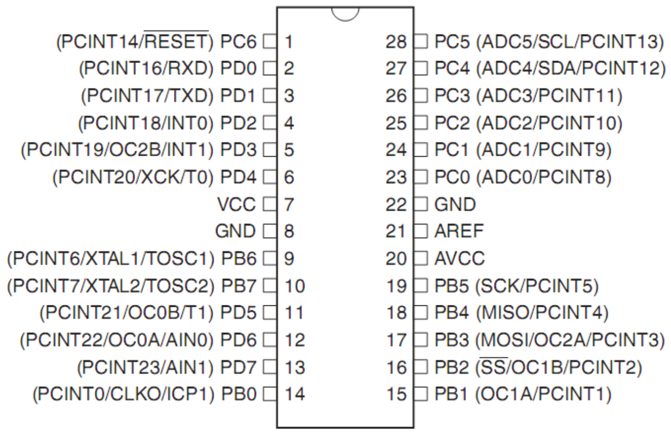


Fig. 5.25: ATmega328 28 PDIP package pinout

The ATmega328 has 6 analog inputs, each of which provide by default 10 bits of resolution, thus allowing 1024 different values at a frequency of 9KHz, which is much higher than necessary. Also by default they measure from ground to 5 volts though is it possible to change the upper end of their range using the AREF pin. Using the default 5V reference grants the ADC a an accuracy of 4,883mV. As presented in chapter 5.1, the signal amplification was designed to provide a maximum signal amplitude of 3Vpp. The minimum useful signals are small potential variations on the vertical channel during left and right eye saccades (as shown in chapter 5,2,1) which present an amplitude of approximately 800mV. The accuracy of 4,883mV is, therefore, more than sufficient for the signal variations we are studying after the signal has been amplified.

5.2.1 Pattern detection algorithm

A number of different pattern detection algorithms were studied from varied sources[5][31]. After some investigation the one which was considered to provide a better accuracy at detecting the necessary patterns while maximizing the processing potential of the ATMega328 was a variation on the Nearest Neighbour algorithm. K-Nearest Neighbour (K-NN) is a method for classifying objects based on closest training examples in the feature space. This means searching the closest K samples and according to their class define the object as belonging to the class that has the most samples among that group.

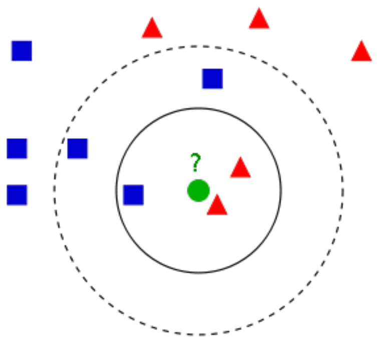


Fig. 5.26: Visualization of a K-NN search for a two-dimensional feature space

In Fig. 5.26 there are two classes of patterns (blue square and red triangle) and two dimensions in the feature space (which can be translated into a system of x-y coordinates). In green is depicted the object which the algorithm is trying to assign to one of the classes. If a 3-NN is used the object is considered to belong to the red triangle class while if a 5-NN is used it is considered to belong to the blue square class. If a lazy-learning system is being used, this new object will not only be identified as belonging to one of the classes, but it will be considered as a sample for future object classifications. Given the limits of the ATMega328's processing power a simpler version of the K-NN algorithm was used where no learning exists and each pattern class only has one sample, previously defined as the average of the training samples collected for that pattern.

The Euclidean distance algorithm was used as follows:

$$L(x,y)=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2} \tag{5.16}$$

Where L stands for the distance between the current object (x) and the pattern class which it is being compared to (y). d is the number of dimensions of the feature space.

In order to gather the training samples, decisions had to be made regarding the number of dimensions of the system as well as the time for each acquisition period. For the current work this means the number of readings used to define each of the nine pattern class samples (blink, up, up-right, right, down-right, down, down-left, left, up-left). As a test value 200 dimensions were used. Given that this work has two channels being acquired simultaneously this means that for each channel the pattern samples will have 100 readings. To determine the acquisition period, the signal duration was taken into consideration. By observing the signals which had previously been recorded with the aid of the oscilloscope, the time window considered relevant was set at 700ms.

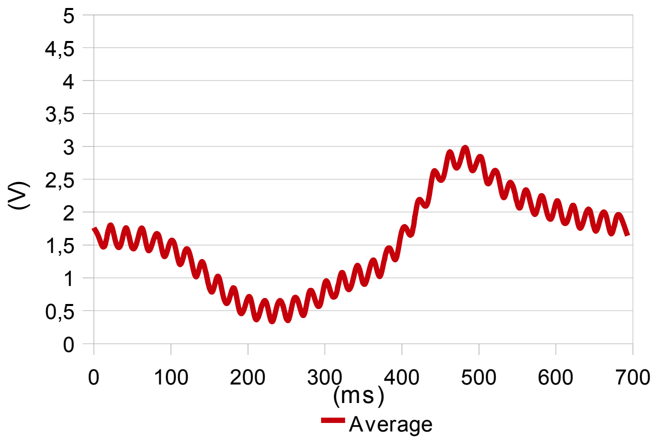


Fig. 5.27: Average signal produced on the vertical channel for a DOWN-LEFT eye movement during a 700ms interval

This time actually depends on how quickly you produce the eye saccades. The faster signals are usually the ones produced by blinking, followed by the ones produced by up, down, left and right while the diagonal ones are usually the slowest given the extra effort it takes to move the eye to an extreme diagonal position. The signal reflux was also considered as part of the pattern for this work in order to provide more accurate readings. The drawback of this is that it adds some delay to the pattern detection. The reading is not immediate because of the reflux time, which is approximately the same as the eye movement.

Having a 700ms time window during which 100 readings are done means that an acquisition cycle runs every 7ms. The implications of this is that each pattern recognition cycle must be processed under 7ms. As such, if too many reading values are fed into the algorithm, it might be too much too process under 7ms. This interdependency of the number of readings, the acquisition cycle and the duration of the pattern recognition cycle make it difficult to determine theoretically what are the values that maximize the pattern detection effectiveness given the restraints in processing power.

$$\frac{Relevant\ time\ window}{Number\ of\ readings} = Acquisition\ cycle\ period \tag{5.17}$$

$$Pattern\ recognition\ duration < Acquisition\ cycle\ period$$

$$Pattern\ recognition\ duration \propto Number\ of\ readings$$

The hardest value to determine is the relationship between the number of readings which is fed into the algorithm and the time which it takes to process it. This could only be perfectly specified by writing the code in machine code or assembly, and relating the number of clock cycles each instruction requires from the processor (1 to 4 clock cycles) to the processor clock frequency (20MHz). When programming in C based languages this becomes almost impossible, given that it is not transparent the way in which the compiler translates C logic into machine code. As such, an experimental approach was taken, starting with the values mentioned before.

All the EOG signals presented in this work were taken from the same subject and the ones displayed henceforth have been captured directly via Arduino's ADC after the signal has been filtered and amplified through the PCB. The first step was to gather a set of pattern reference samples for the various directional saccades and blinking. The process was identical for all the patterns, so in order to explain the process through which they where obtained, the involved steps will be presented for the case of the blink pattern.

Blink Pattern – Full Procedure

Firstly, the user was asked to blink the eye during a 1750ms interval. This duration is large enough to give the patient time to comfortably execute the eye movement without missing the recording time window. During this period both the Horizontal and the Vertical channels were recorded in 7ms intervals by the microprocessor which then printed the 250 horizontal values and 250 vertical values to the computer through serial communication. In the computer this values were converted from their 10bit (0-1024) value to their respective voltage range (0V-5V) and displayed graphically, as shown below.

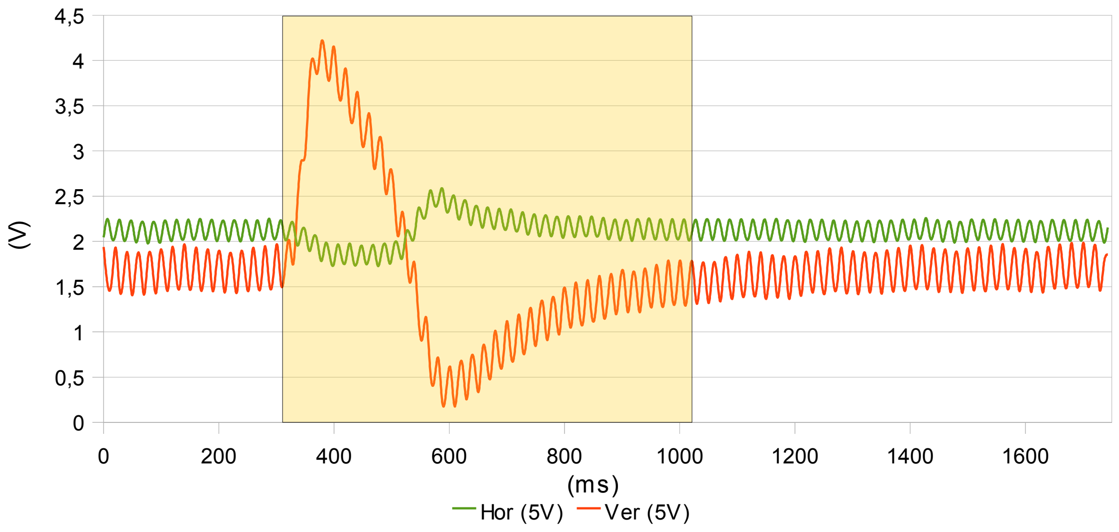


Fig. 5.28: Signal recorded during a 1750ms interval for a blink movement. The relevant time window is highlighted

Although there is a considerable amount of 50Hz noise as well as a non-perfect offset (which should be around the 2,5V mark), one can easily distinguish the typical voltage spike that the blinking movement emits while little variation occurs on the horizontal channel.

The noise could easily be reduced by the usage of the shielded cable which is proposed in chapter 5.4. The offset error was due to slight differences in electrode sensitivity between electrodes of the same pair – horizontal pair (electrodes left and right) or vertical pair (electrodes up and down). For this particular case of sample gathering the offset error was later corrected by hand, placing the pattern sample signals around the 2,5V through a simple sum. For further readings only absolutely new electrodes have been used together with an algorithm to compensate offset deviation.

The graphical display presented in Fig. 5.28 facilitates the search of the relevant time window for each sample which, in the case of this first sample, was considered to be the interval 310-1010ms.

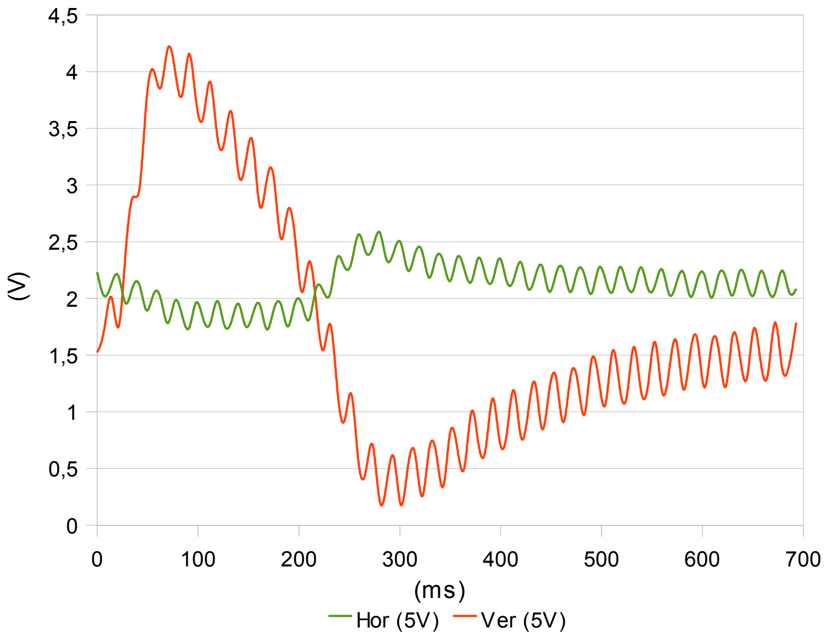


Fig. 5.29: Relevant time window for a blink movement pattern

This process was repeated 10 times allowing the creation of an average pattern. The resulting waveforms have been overlapped for each individual channel and presented in the graphics below.

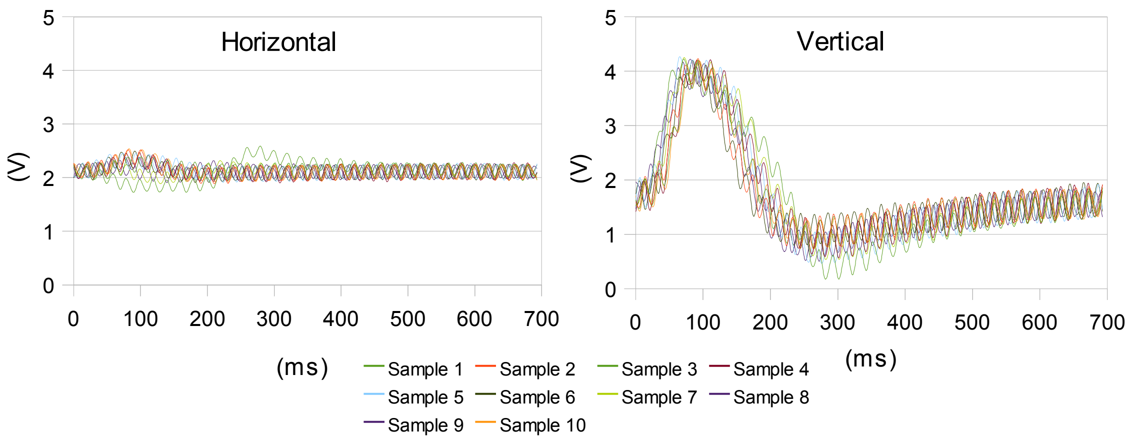


Fig. 5.30: Overlap of 10 waveforms resulting from a blink eye movement

The resulting waveforms, particularly for the blink movement, have a very constant pattern which can easily be identified. The average waveform for the 10 samples was subsequently determined as well as their standard deviation.

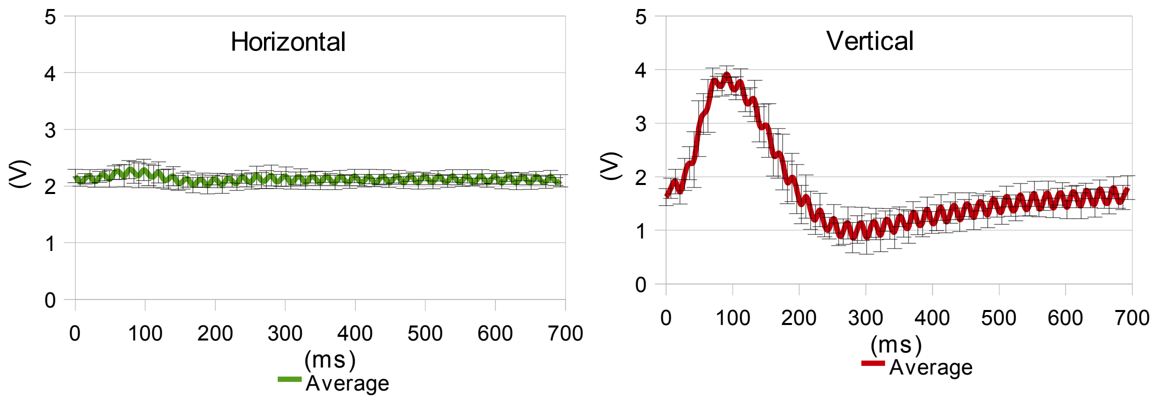


Fig. 5.31: Resulting average horizontal and vertical waveforms for the blink movement

The standard deviation for the horizontal channel values varies between 0,07 and 0,22 while for the vertical channel it varies between 0,13 and 0,52. These are very acceptable values that grant that the system will be able to effectively determine the signal due to the low variation it presents throughout the many repetitions of the eye movement.

The same procedure was taken to obtain the pattern samples for the following eye movements – up, up-right, right, down-right, down, down-left, left, up-left. The results obtained are presented below.

Up Pattern

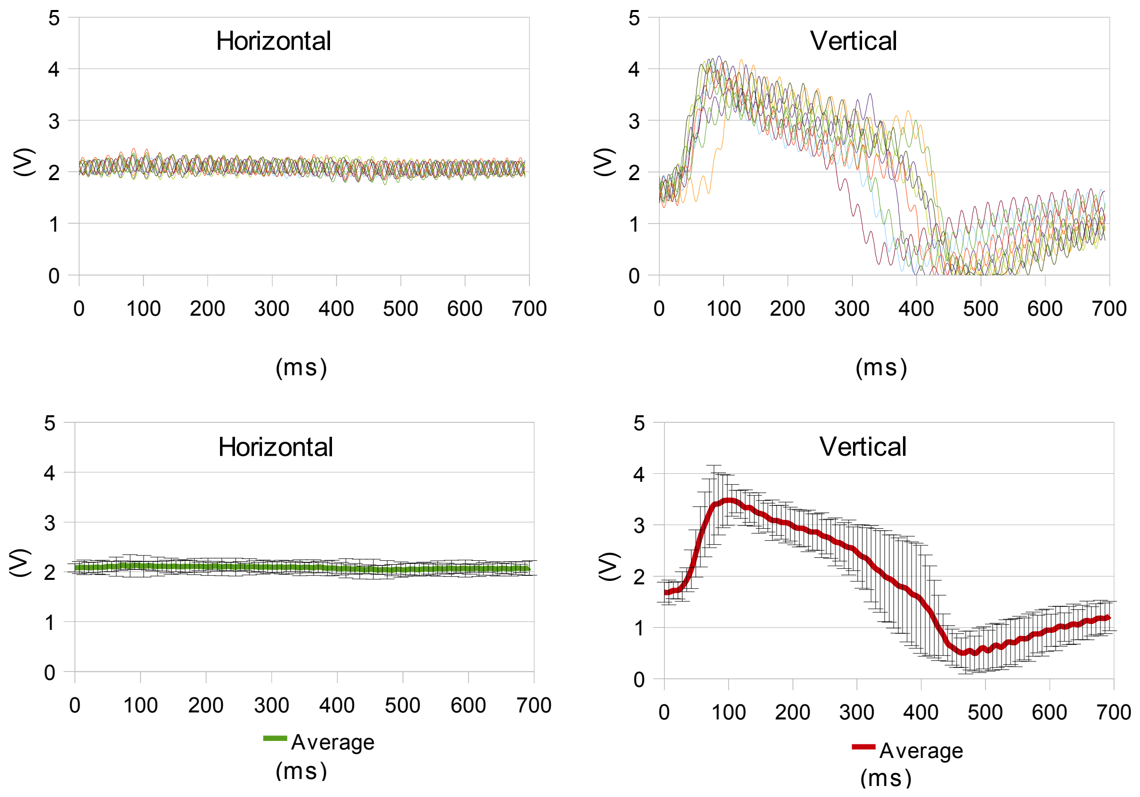


Fig. 5.32: Ten samples and resulting average horizontal and vertical waveforms for the UP movement

Up-Right Pattern

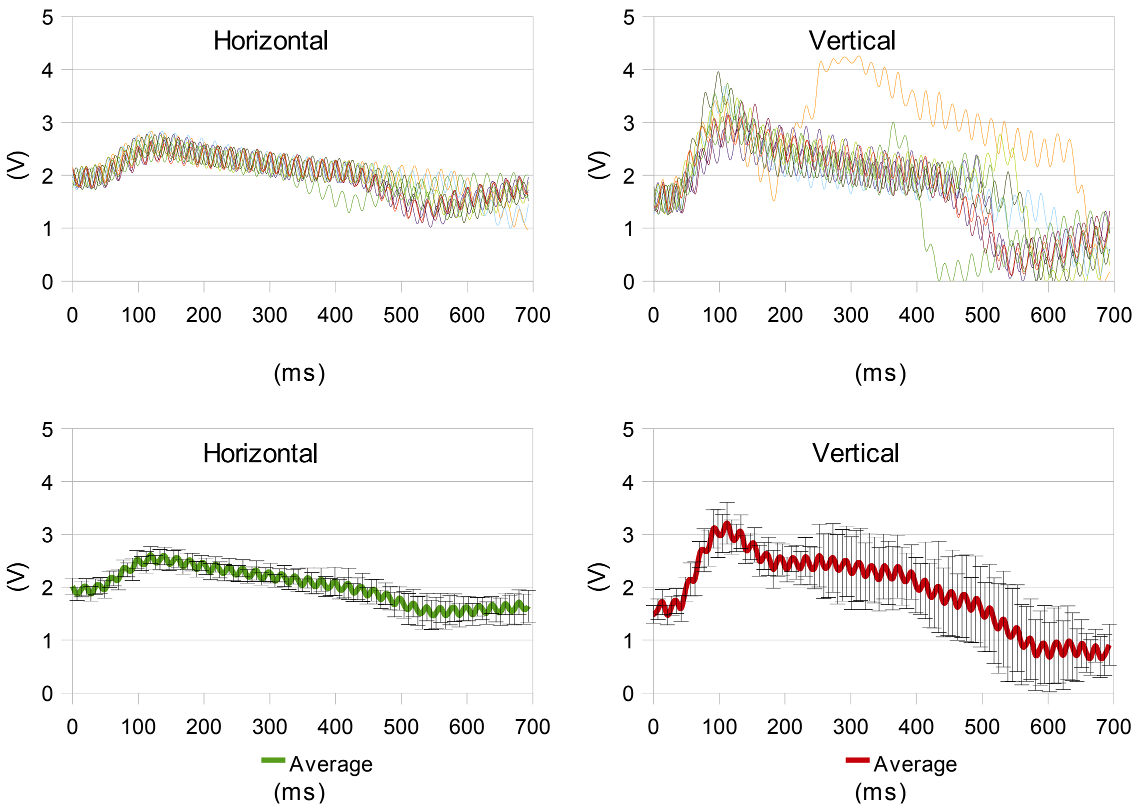


Fig. 5.33: Ten samples and resulting average horizontal and vertical waveforms for the UP-RIGHT movement

Right Pattern

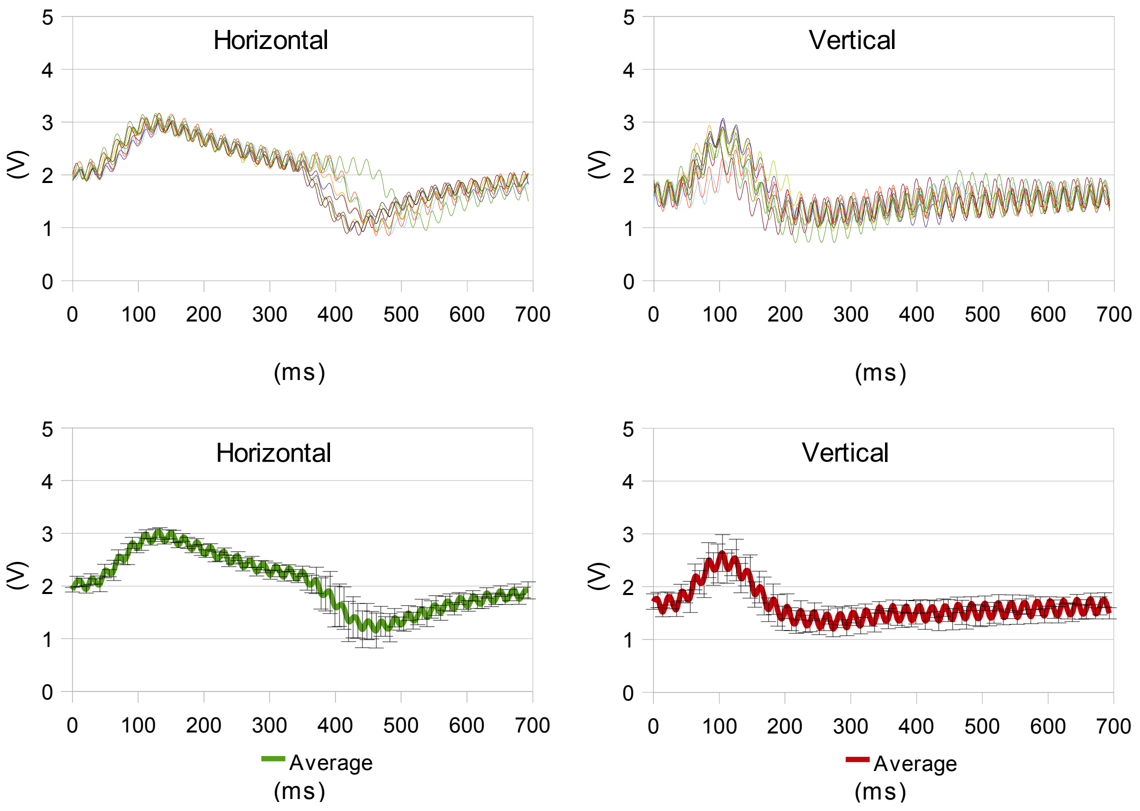


Fig. 5.34: Ten samples and resulting average horizontal and vertical waveforms for the RIGHT movement

Down-Right Pattern

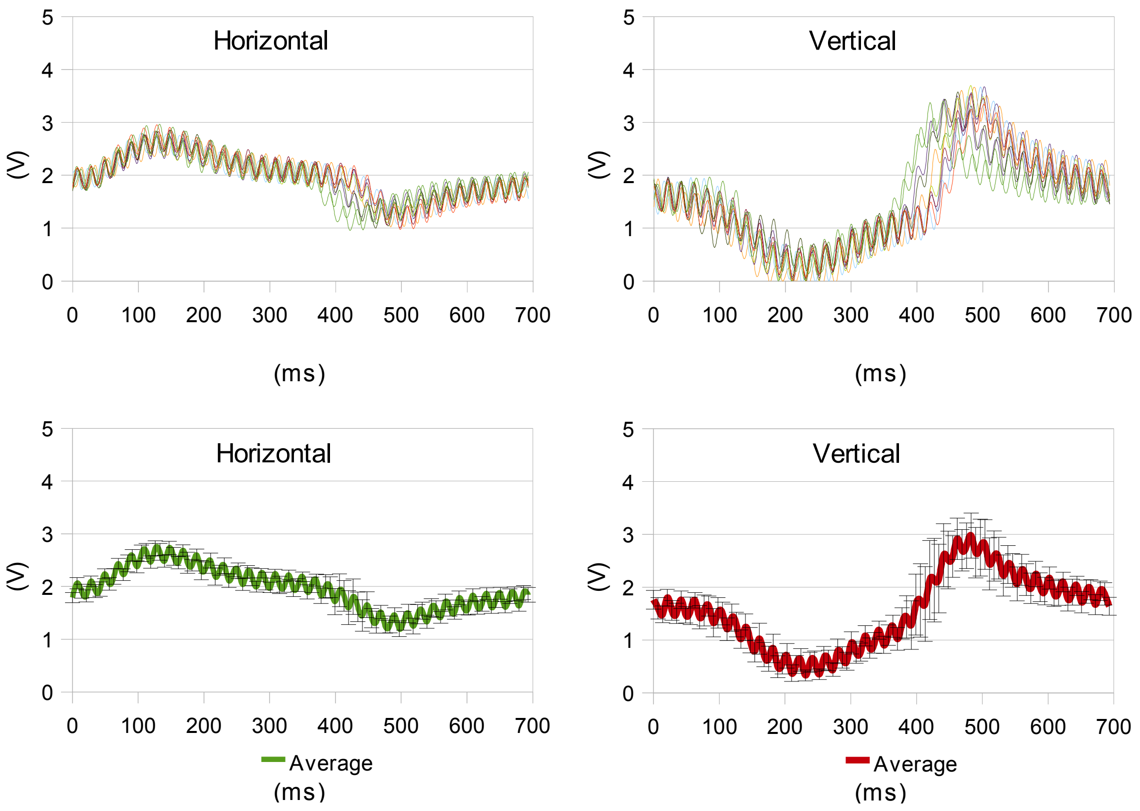


Fig. 5.35: Ten samples and resulting average horizontal and vertical waveforms for the DOWN-RIGHT movement

Down Pattern

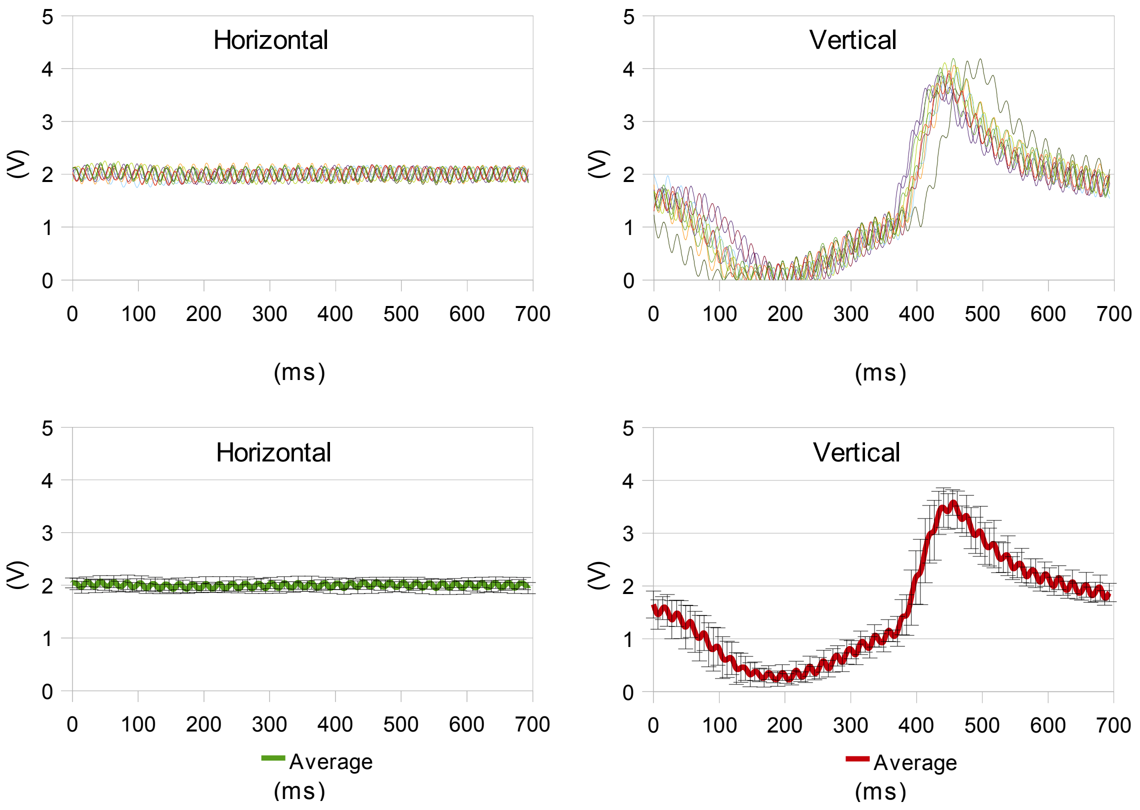


Fig. 5.36: Ten samples and resulting average horizontal and vertical waveforms for the DOWN movement

Down-Left Pattern – Results

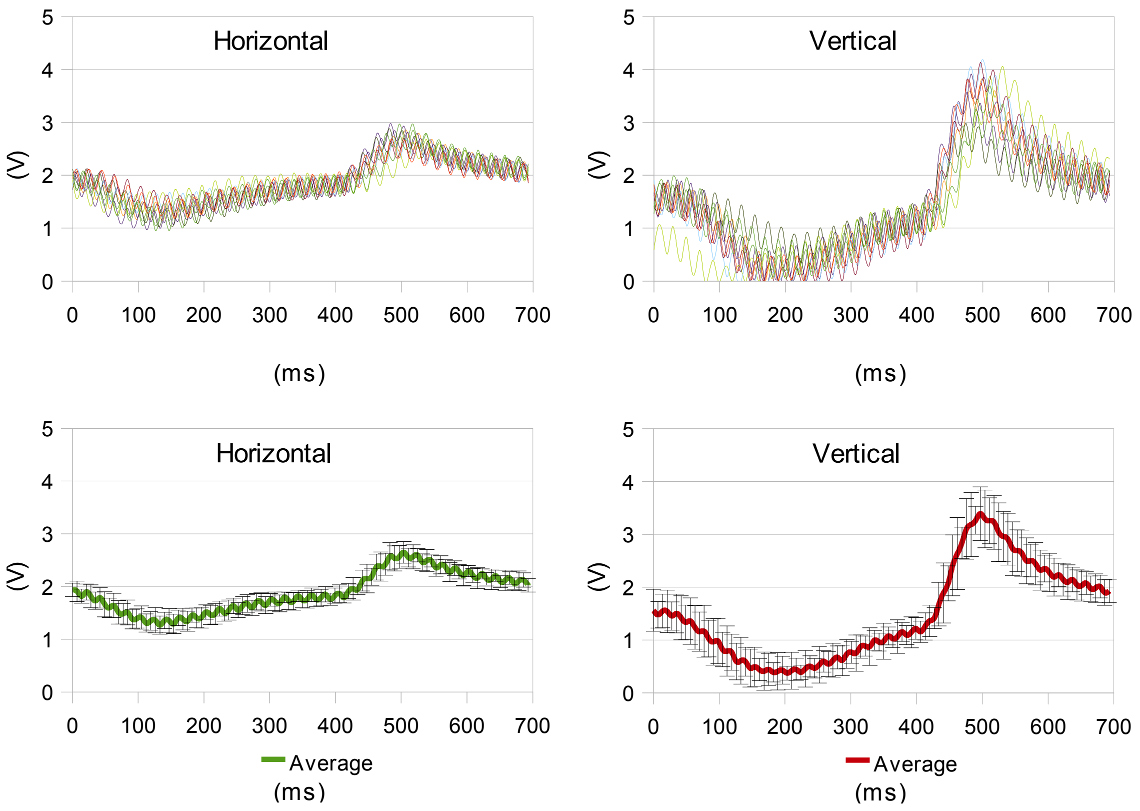


Fig. 5.37: Ten samples and resulting average horizontal and vertical waveforms for the DOWN-LEFT movement

Left Pattern – Results

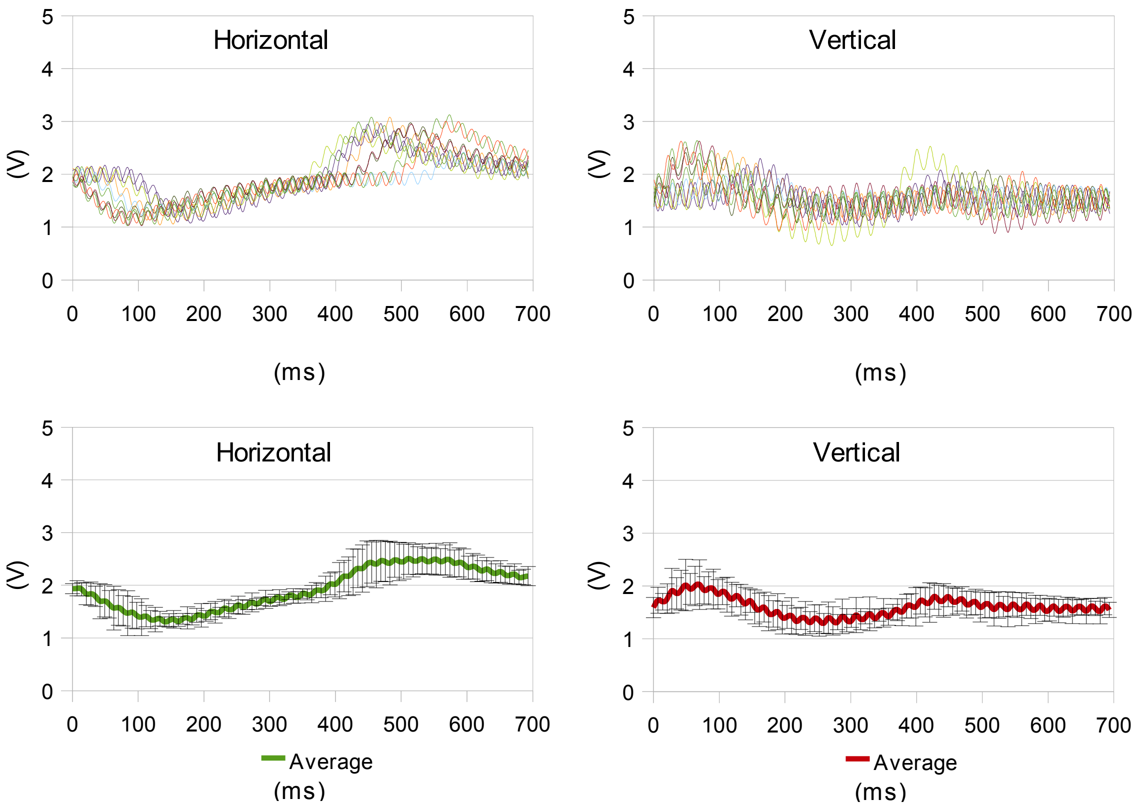


Fig. 5.38: Ten samples and resulting average horizontal and vertical waveforms for the LEFT movement

Up-Left Pattern – Results

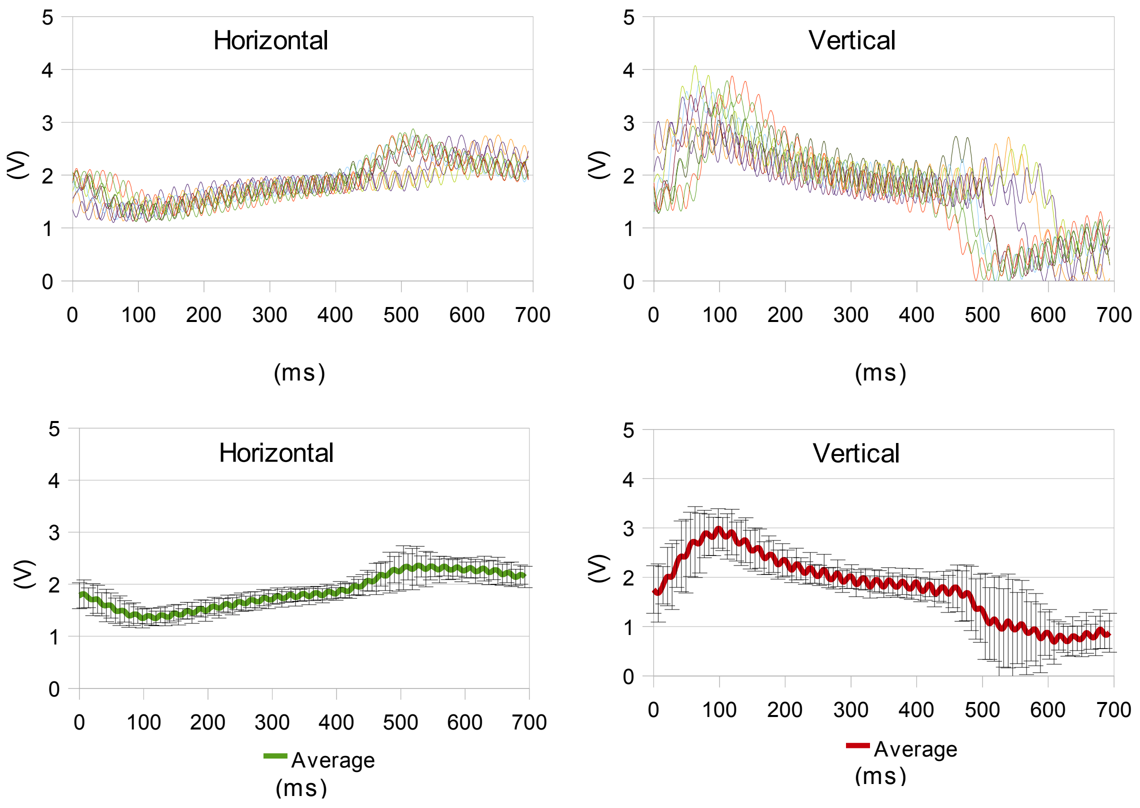


Fig. 5.39: Ten samples and resulting average horizontal and vertical waveforms for the UP-LEFT movement

Considerations on the collected pattern samples

The signals differ from each other in quite a visible manner. The standard deviation values, although generally quite small (usually in the 0,1-0,7 range), are somewhat higher for the vertical channel when the up movement is involved (up, up-right and up-left saccades) reaching values up to 1,06. No particular reason for this can be found, except for inherent variations in the user's eye movement when performing these saccades. The higher deviation presented by the vertical sample sets associated with the up movement, although not as good as the others, is still within the acceptable limits of accuracy.

The most visible criticism that can be given to the previous waveforms is the reduced amplitude of the horizontal channel, probably due to testing errors during the previous stage of this work where the gains where reduced to their half, as explained in chapter 5.1. The maximum amplitude measured in the horizontal channel is 1,5V while the desired value was of 3V. This issue can be solved by replacing R11 and R12 with the previous 1KΩ resistors or replacing R13 and R14 with 150KΩ resistors and C5 and C6 with 33nF capacitors.

After all the signal patterns had been determined it was necessary to implement the pattern detection algorithm in the micro processor.

5.2.2 Algorithm implementation in Arduino

Later during this stage it became visible that the processor could not handle prperly the necessary calculations involved in the proposed algorithm with 100 readings in a 7ms cycle, as was being attempted. In order to preserve the previous work with a 700ms time window pattern collection, half of the values were used to create patterns with 50 readings, each taken every 14ms.

The rest of this chapter will be written according to the work done with the 14ms reading cycle. The first step was to make the calculated average waveforms understandable for the ATMega328 microprocessor. As such, their offset was corrected and now translated from the voltage range values (0-5V) back to their respective 10bit (0-1024) values.

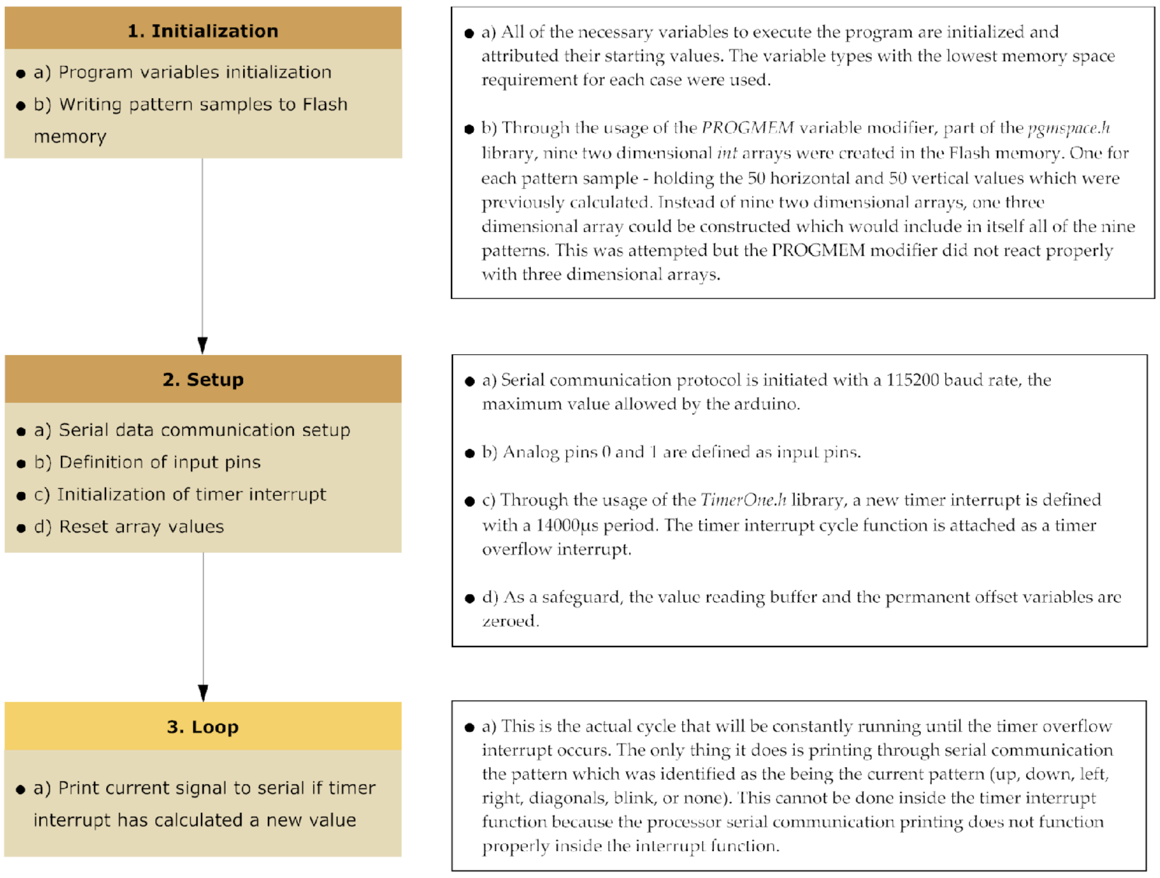
As specified before, the ATMega 328 has only 2KBytes of available RAM memory. This is the type of memory that the processor uses to create and manipulate variables while functioning. Even with only 50 readings it is not possible to store the data related to the 2 channels of the 9 pattern samples. The values are in the 0-1024 range, so they must be stored in *int* variables, which occupy two bytes each given that the *byte* variable type which requires less memory can only store values from 0 to 255. This leads up to a 1800Bytes memory requirement for the pattern detection samples alone.

$$50\,readings \times 9\,patterns \times 2\,byte\;variable = 1800\,bytes \approx 1.76\,Kbytes$$

This is obviously more than the processor would be able to store in its RAM memory, given that all the other variables necessary to execute the program will be forcefully stored and handled in that memory space. When dealing with non-dynamic variables, as is the case given that the pattern samples will not change during the execution of the program, they can instead be stored in the processor's Flash memory and transferred to a RAM memory buffer when necessary.

The complete arduino code is presented in Appendix 1. In this chapter only a simplified pseudo-code approach to the program will be presented. Shown below is the schematic explaining the basic framework of the main program.

Main Program



1. Initialization

- a) Program variables initialization
- b) Writing pattern samples to Flash memory

- a) All of the necessary variables to execute the program are initialized and attributed their starting values. The variable types with the lowest memory space requirement for each case were used.
- b) Through the usage of the *PROGMEM* variable modifier, part of the *pgmspace.h* library, nine two dimensional *int* arrays were created in the Flash memory. One for each pattern sample - holding the 50 horizontal and 50 vertical values which were previously calculated. Instead of nine two dimensional arrays, one three dimensional array could be constructed which would include in itself all of the nine patterns. This was attempted but the PROGMEM modifier did not react properly with three dimensional arrays.

2. Setup

- a) Serial data communication setup
- b) Definition of input pins
- c) Initialization of timer interrupt
- d) Reset array values

- a) Serial communication protocol is initiated with a 115200 baud rate, the maximum value allowed by the arduino.
- b) Analog pins 0 and 1 are defined as input pins.
- c) Through the usage of the *TimerOne.h* library, a new timer interrupt is defined with a 14000µs period. The timer interrupt cycle function is attached as a timer overflow interrupt.
- d) As a safeguard, the value reading buffer and the permanent offset variables are zeroed.

3. Loop

- a) Print current signal to serial if timer interrupt has calculated a new value

- a) This is the actual cycle that will be constantly running until the timer overflow interrupt occurs. The only thing it does is printing through serial communication the pattern which was identified as the being the current pattern (up, down, left, right, diagonals, blink, or none). This cannot be done inside the timer interrupt function because the processor serial communication printing does not function properly inside the interrupt function.

While the loop cycle is running, the processor's timer1 keeps running and the collectData() function is called every 14ms, through a timer overflow interrupt. The basic framework of this function is shown below.

Function called upon timer interrupt

T1. Data collection

- a) Input value from analog pins 0 and 1 is read

- a) Reads the raw values from the 10-bit ADC related to pins 0 and 1

T2. Value offsetting

- a) Difference between average for the last 50 values and 512 is considered to be the immediate offset
- b) Approximation of effective offset to immediate offset
- c) Raw readings saved for future offset calculation
- d) Readings corrected with offset saved for pattern comparison

- a) Due to minor differences in electrode sensitivity from electrodes of the same pair (up-down or left-right), the incoming signal is not perfectly centered around the 2.5V mark (or 512 for a 10-bit ADC working in the 0-5V range). The immediate offset is considered to be the difference between the average value that the signal displays during the past 50 readings and the expected 512 mark.

- b) If the calculated immediate offset is directly applied to the readings, this algorithm would work as a high pass filter and present the same reflux artifact which is present in the hardware filter. Basically, every time that a reading would display a spike, the offset would spike in the opposite direction given the short time-window for the calculation of the average. This was solved by slowly and incrementally changing the value of a new variable - the effective offset.

- c) The raw readings are saved in an two dimensional array with 50 readings and two channels (horizontal and vertical). This array is used for future calculation of the immediate offset.

- d) The current readings are corrected with the effective offset and saved in an two dimensional array with 50 readings and two channels (horizontal and vertical). This array is used for pattern detection.

T3. Distance to pattern sample

- a) Distances from current reading array to pattern samples is reset
- b) Distance between current reading and each pattern sample is calculated for both channels
- c) Horizontal and Vertical distances for each pattern sample are combined

- a) All of the distance variables are zeroed in the beginning of this algorithm.

- b) The following cycle is repeated for each of the 50 readings which constitute the current reading array:

  b.1) Import appropriate horizontal and vertical pattern values from Flash memory to a variable in the RAM memory space, because the processor cannot work with values stored in Flash memory;

  b.2) Calculate distance between horizontal and vertical current readings array values and respective horizontal and vertical imported pattern values;

  b.3) Sum the obtained distance to a total horizontal and a total vertical distance;

  b.4) Repeat this process for each of the nine pattern samples.

- c) For each pattern sample the horizontal and vertical distances are combined, giving and added weight to the horizontal channel, due to its lower gain.

T4. Distance comparison

- a) Current pattern is reset to *none*
- b) Determine if any distance is shorter than a defined threshold and , if so, which one is the shortest to define it as current pattern signal

- a) Define current pattern as none. If the current readings array is not close enough to any pattern sample, the signal is considered as being neutral.

- b) For an optimized version of the hardware the only necessary calculation would be to determine the pattern sample which is closer to the current readings value and check if it is under a certain threshold. Nevertheless, due to the reduced signal amplitude in the horizontal channel, some adjustments had to be made. Basically four different distance thresholds were applied to four different classes - Blink, Basic Directions, Lower Diagonals, Upper Diagonals. After the best values for this classes had been determined, the algorithm presents a very good accuracy.

T5. Finalization

- a) Increase iteration
- b) Inform the loop cycle that a new value was calculated and should be printed

- a) Increase iteration counter. If maximum value has been reached, return to zero

- b) Sets the boolean variable *printInfo* to true, informing the loop cycle that the new calculated values should be printed through serial communication

5.2.3 Considerations

Unfortunately it was not possible to do a thorough testing of the filtering stages and compare real frequency dependent gains against the simulated results. With the signal generators available in the laboratory it was not possible to produce reliable waveforms with hundreds of microvolt even with tension divisors. The final simulated gains obtained were 2170 for the horizontal channel and 4350 for the vertical channel. The simulated pass-band for both channels was 0,949Hz – 33.879Hz with the cut-off frequencies being considered at -3dB. These values are quite close to the 1Hz-35Hz range deemed valuable.

Although the obtained results are quite satisfactory they are not without some flaws. The first one is, once again, the low amplitude of the horizontal signal. The second one is that the pattern samples were obtained from only one user and the system was only tested with that same user. This does not guarantee the system the necessary universality, although no reasons for it not functioning with other users are foreseeable.
Another less positive aspect of this algorithm is that, using the reflux part of the signal to detect the patterns, after the user has finished the eye movement (approximately 350ms) it is still necessary to wait for the hardware module to stabilize the signal (approximately another 350ms) imbuing the system with an inherent delay which would not be necessary if the reflux was not taken in consideration for pattern detection or if it simply did not exist at all.

Given that the pattern detection must run in under 7ms and that the communication between the processor and the computer occurs in a time-scale which can be considered irrelevant, we can consider that the device delivers a delay which is under 357ms.

The final criticism is that the Signal to Noise Ratio (SNR) is not as good as was expected. SNR is the power ratio between the desired part of a signal and the background noise. Given that the signal and the noise are measured with the same impedance, they can be calculated through their amplitude as shown below.

$$SNR=\frac{P_{signal}}{P_{noise}}=\left(\frac{A_{signal}}{A_{noise}}\right)^2 \tag{5.18}$$

For the example case of the blink movement, while on the vertical channel (which is its primary source of signal for this pattern) there is a very good SNR of around 130, the horizontal channel (which presents a much lower desired signal amplitude) displays a SNR of approximately 11.

Although the device already presents an acceptable accuracy with these values, it could be greatly improved by the usage of more appropriate cables and electrodes as well as a higher gain on the horizontal channel. The increase in the gain would be produced by the second stage of the instrumentation amplifier which is the low pass filter. This would grant a lower gain of the noise frequency (50Hz) than the desired signal, providing an increase in SNR.

5.3 Software - Graphic User Interface

As the final stage of human-computer interaction, the main purpose of this GUI was to generate a simple and readable yet robust and complete computer experience to the end user.
The platform chosen to develop the graphical interface was Processing which is a JAVA based open source programming language and environment. Processing was chosen because of the following characteristics:

- Free to download and open source
- Very well documented and has a strong community support
- Designed for interactive graphical applications
- Exports directly to Linux, Mac OS X and Windows as a stand-alone JAVA application
- Great amount of user contributed libraries
- Easy communication with the Arduino platform
- My previous experience with it

As far as interactive visual output design goes Processing can be considered a very low-level developing environment. Unlike other popular commercial platforms, like Adobe's Flash and Microsoft's Silverlight, Processing does not have a graphical interface. It is a pure written language where, for instance, the user must define circles through functions instead of designing them with mouse clicks. This makes it much harder to create visuals but allows for an unparalleled level of configuration and produces extremely lightweight applications.

No particular work has been done on aesthetics, the design efforts have been directed towards proposing a basic interface geometry which can later be a target of a makeover. As we have seen before, the entire system relies on 8-directional input and blinking. As such, the basic principle was that every menu had to be based on a maximum of eight options so that all of the available options would always be immediately reachable with a minimum of input activation.

Graphically, given the nature of the directional input, a circular button arrangement was developed like shown below.

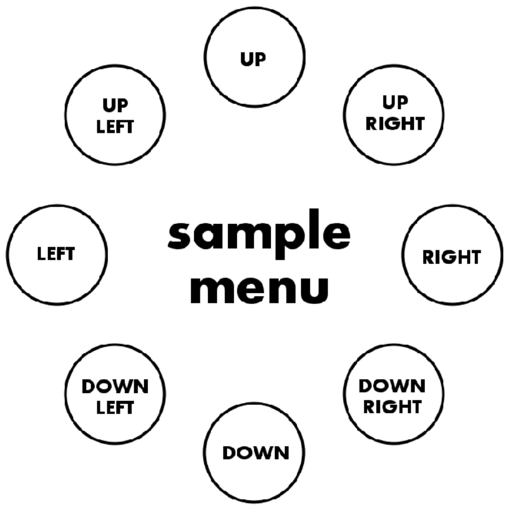


Fig. 5.40: Basic menu geometry sample

Let us imagine the user is in "sample menu" and wishes to enter the "UP" option. He must first select the "UP" button which remains activated for a predetermined amount of time and then click it. The button selection is accomplished by performing a quick eye movement from the computer screen to the far edge of the desired direction and back to the computer screen. The visual queue for this selection is the highlighting of the selected button as shown in Fig. 5.41. While the button is selected clicking is performed by an eye blink.

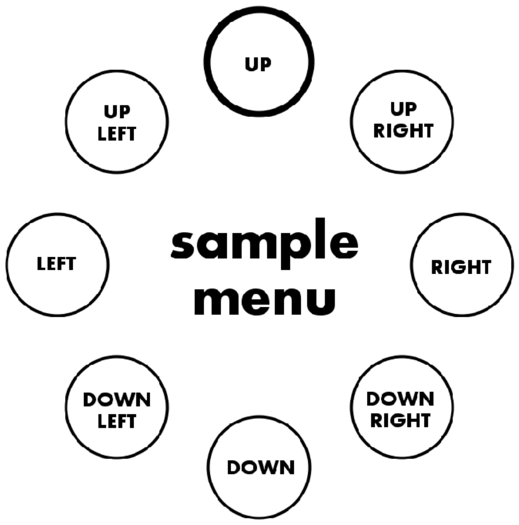


Fig. 5.41: Basic menu geometry sample with selected "UP" button

This sequence ensures that there is a very low probability of a non-desired input order. The user can thus blink freely and without restrictions because, only if he has performed an extreme and infrequent movement just before blinking, will blinking be perceived as an input order. Performing a quick eye movement from the center of your vision field to the far edge and back to the center of your vision field is a movement which has a very low chance of occurring as a natural reflex. This is due to the primitive targeting nature of the saccade, as has been presented in chapter 2.2. The quick eye movement, or saccade, occurs when an object outside your central field of vision requires your attention. When this happens you move your eye gaze into the point of interest and rest your eye there for a minimum of time needed to processes the visual information. Just as when you are copying text from one page to another or someone calls you. The brain needs some time to process the visual stimuli and understand what must be copied or who is calling you and what facial expression they are calling you with.

Ensuing the design of the basic geometry a basic set of applications had to be defined and implemented. This GUI is specifically designed for the needs of a patient who suffers from severe lack of motor and speech coordination thus the three indispensable set of applications was defined as :

- Call for aid
- Predefined messages selector
- Text editor

This set of applications already provides an invaluable help in communicating with those surrounding the patient yet, in order to greatly expand the reach of the GUI, a web browser/media player and mouse controller were projected. Unfortunately, due to time constraints it was not possible to pursue this goal as it would require the whole programming to be translated into pure JAVA as processing would not be able to fulfill this requirements in an elegant way by itself. The best method to execute a web browser as well as the media player inside a stand-alone application would be through the usage of the NativeSwing library which is based on Eclipse's SWT JAVA GUI tool kit. Processing applications do not support this, as they are based on the different AWT (Abstract Windows Toolkit) library and cause severe conflicts when attempted to run with SWT-based libraries. NativeSwing would also allow the seamless implementation of a Flash player and an HTML editor.

The projected application flow was designed as shown below:

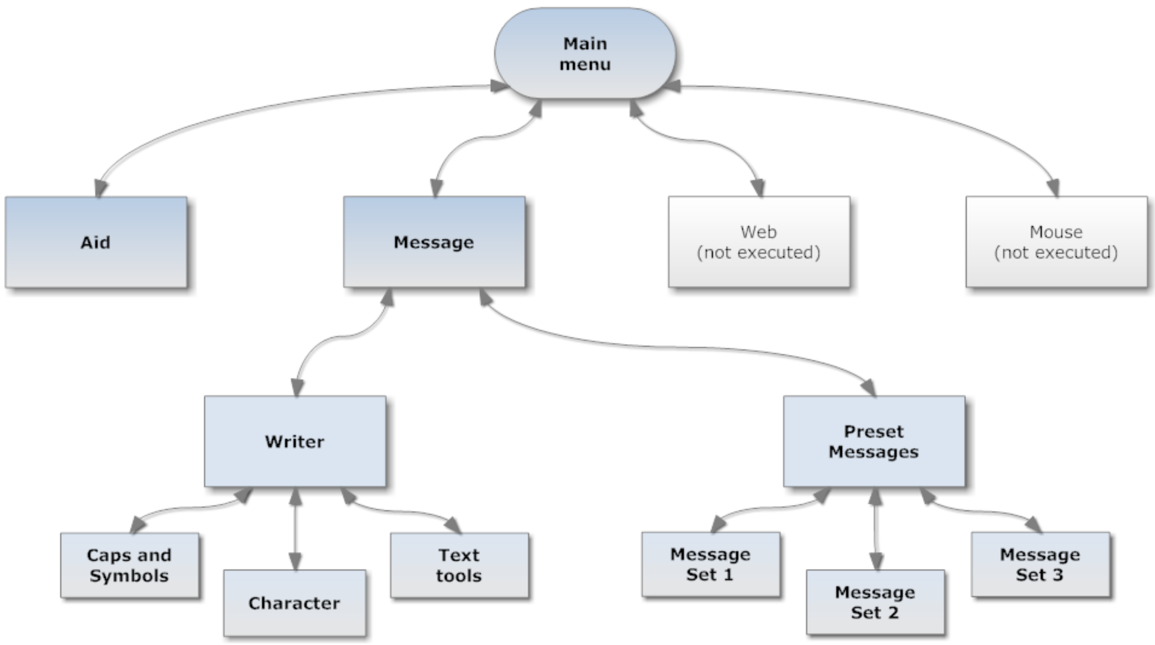


Fig. 5.42: Graphic User Interface application flow

5.3.1 Call for aid

For patients who suffer from severe communication inabilities one of the simplest tools can be the most important - a call for aid. Simply to let a helper or family know that some sort of aid is necessary. It can either be to warn about a basic hygienic or nourishment need or to communicate the existence of a particular pain which may lead to an early diagnosis of a dangerous health condition. It can even be helpful as a simple request to look at the screen in order to show something that has been or will be written..

The first steps through which this can be achieved is to emit a visual and auditory signal in order to alert nearby people that attention is required. This has been achieved by making the screen perform a quickly fading passage between a black and a white background while a warning sound is played in loop. Once again, this was all programmed within a strictly functional frame of mind, no particular attention was devoted to aesthetics. In order to exit the call for aid menu the user must press the left button.

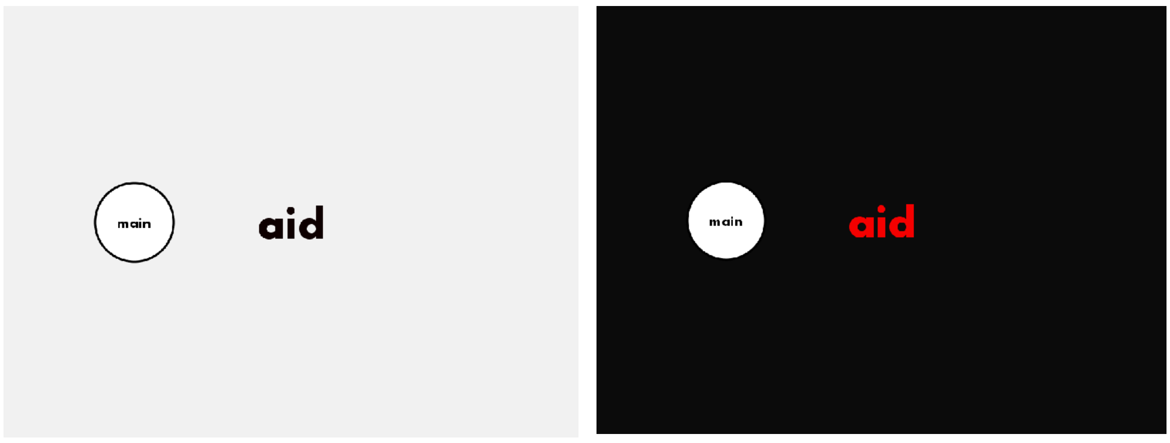


Fig. 5.43: The screen displays a faded blinking in the aid menu

This functionality works very well when help is in close range but lacks in effectiveness if the patient is not in a setting of constant surveillance which is quite common in cases like those that are being addressed. Many different solutions can thus be considered. Placing the audio near the helper is quite probably the simplest and most effective way to solve this problem. For instance connecting the audio output of the computer to speakers and placing them in the bedroom of the patient's parents.

There are many options beyond this one that can easily be achieved through the usage of Processing data communication protocol libraries like XBee, Bluetooth, TCP-IP, UDP, XML-RPC, EEML, RSS feeds, Tweeter, among others. The implementation of these options was not considered to be a priority for the present work but could be a subject of further study. Most of them would require, however, the development of a dedicated client-side hardware and/or software. With enough effort this could lead to very interesting options like having a small ZigBee radio based device that would emit a sound and vibrate or receiving a message on your phone through Bluetooth or SMS in order to warn that the patient is requesting aid or even displaying the text that he is writing at the moment.

5.3.2 Predefined messages selector

The feeling that the patient wishes to express will often be a frequently repeated message like "I am hungry", "Thank you", "I am tired", etc. For these specific cases a predefined message menu was implemented. In order to produce a more flexible interface experience that can be adapted to the needs of each individual user the available messages can be edited in external text files in character strings composed of up to 38 characters. This messages are separated into three sets to allow further organization. The user can define, for instance, a set for "Needs", another for "Conversation" and leave the third one for "Miscellaneous".

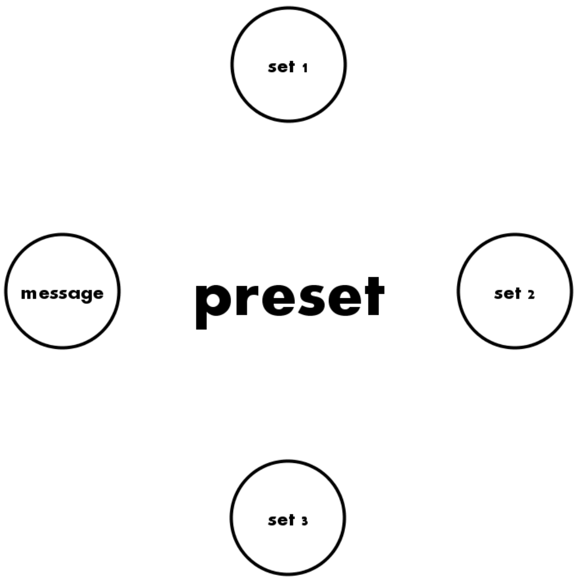


Fig. 5.44: After entering the preset message menu the user can choose which set to access

Ensuing the selection of which set the user wishes to access, the default button menu changes it's size and position in order to allow more space for the rest of the information which now needs a bigger area to be displayed. The button menu is reduced and placed in the top-left part of the screen while the preset messages are displayed in the top center-right part of the screen.

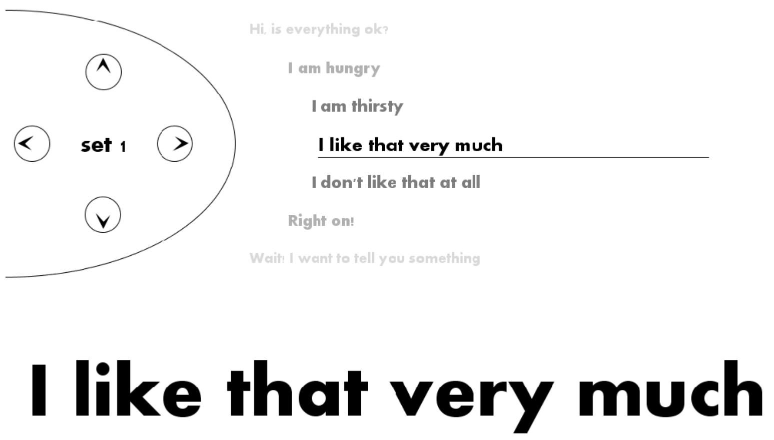


Fig. 5.45: After selecting the sentence set "set 1", the user scrolled down to the desired sentence and chose to display it

When a message is selected through clicking the right button it is then displayed in the bottom part of the screen in a bigger font.

5.3.3 Text editor and key placement

It would be tremendously restrictive if the user would only be allowed to select from a number of predefined messages. As such, a simple text editor was also created. The first concern was how to place the characters in the screen. Instead of immediately adopting the usual QWERTY keyboard-like character placement an alternative keyboard geometry is proposed. The typical QWERTY positioning associates each character to a dedicated key.

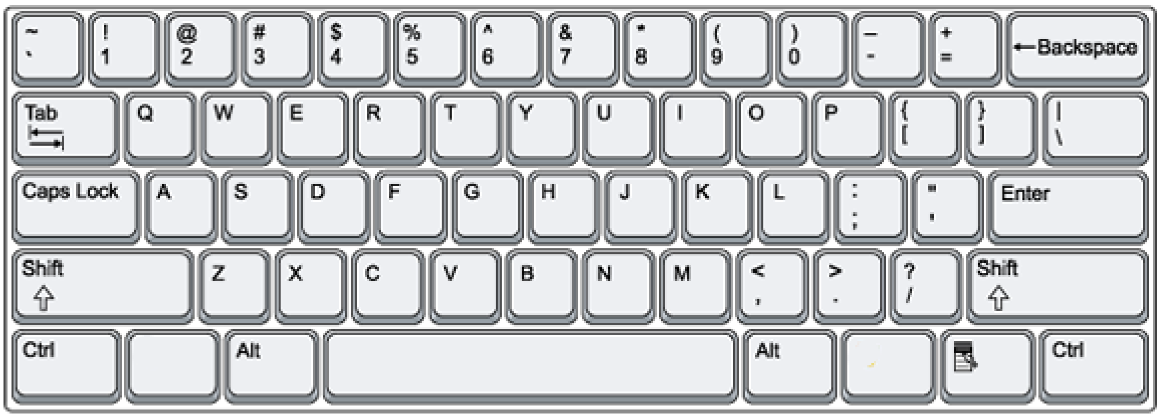


Fig. 5.46: Typical QWERTY layout

This is still, today, the most widely used character layout for this type of devices, not to mention for every other type of device. It does makes perfect sense when you are able to press 26 independent and separate keys - and this is considering alphabet characters only. Nevertheless, due to the inherent design restrictions already mentioned, the proposed device only allows for 8 directional inputs and a blink. Given these limitations let us consider the most common different types of text entry geometries which could be used to improve writing speed.

Optimized QWERTY

When restricted to such a reduced number of inputs, in order to interact with this type of keyboard, the QWERTY is not able to work as a one-key-one-character input. It works as a scrolling QWERTY. The user must scroll through the letters, usually with an UP-DOWN-LEFT-RIGHT set-up, until the desired letter is reached and then press a select key. A very simple way to improve writing time in a QWERTY type keyboard when the number of inputs is reduced to 8 is to first to allow the user to scroll the letters also in a diagonal set-up beyond the basic UP-DOWN-

LEFT-RIGHT. Secondly, after each character is selected the cursor is repositioned in the centre, which is the letter "G". This reduces the maximum number of inputs that can take for the user to scroll from one character to the next. Let us take the example of the most common English word with more than three letters - "that"[51]. The following picture shows how many inputs or steps must be taken in order to write the word "that" using the typical QWERTY (in red), using QWERTY with diagonal input (in yellow) and using QWERTY with cursor repositioning. To even out the competition it was assumed that all of the text type methods started in the letter"G", despite the fact that many QWERTY scrolling keyboards actually start in the letter "Q".

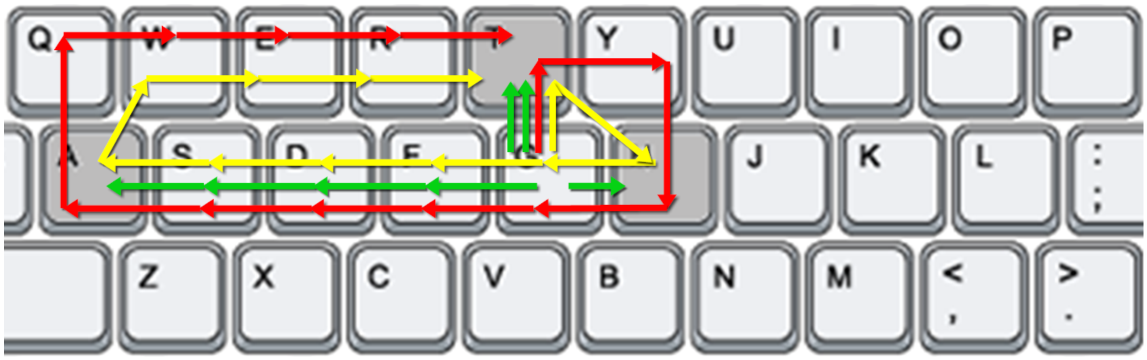


Fig. 5.47: Number of inputs necessary to write "that" in three different scrolling QWERTY setups

In order to write the word "that" the typical QWERTY needs 13 input actions, the QWERTY with diagonal input needs 11 input actions and the optimized QWERTY needs 7 input actions which is almost half of the effort that the first method requires.

This does not necessarily provide better results for every single word. Nevertheless throughout the course of a typical full sentence it does gives this repositioning technique an edge over the standard technique where the cursor remains on the last selected character.

Square keypad

This is the type of keyboard which is most common among P300 BCI (electroencephalogram based) text type applications. Instead of being displayed in three rows, each one with 9 to 10 characters, the letters are displayed in a 6X6 matrix.

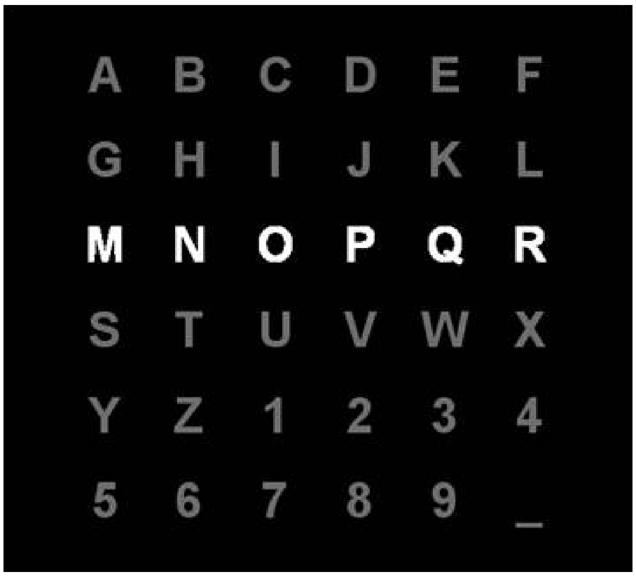


Fig. 5.48: A typical square setup used in P300 applications

This reduces even further the maximum number of input actions that may take from one character to the other given that the maximum distance between letters is much smaller. For instance, to go from one corner to the other it only takes 10 input actions with UP-DOWN-LEFT-RIGHT input and 5 input actions if the user is able to do diagonal scrolling.

Telephone keypad

The past decade and a half has taught us all to use a keyboard which is very reduced – the classic telephone keypad. Although many cellphones are already using QWERTY type keyboards, especially in the touch-screen segment, everyone has used it to some extent. Previously featured in some landline phones it was later re-introduced in cell phones as we know it according to the international standard ITU E.161/ISO 9995-8 and became widely popular due to text messaging.

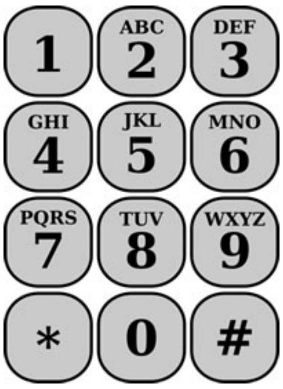


Fig. 5.49: Typical phone keypad

This keyboard also has an alphabet which can be completely accessed through 8 inputs – the keys 1 to 9. Despite the difficulty to reach special characters and text tools this geometry already comes closer to maximizing the input availability of the proposed device. Yet, a new type of character placement was designed.

Concentric keypad

The best key placement is the one which gives the user the ability to reach the maximum number of characters with a minimum number of inputs. Given the 8 input restriction of the device, the best option would be a tree structure design in which each directional input would open 8 new options. This was achieved through a concentric keypad design.

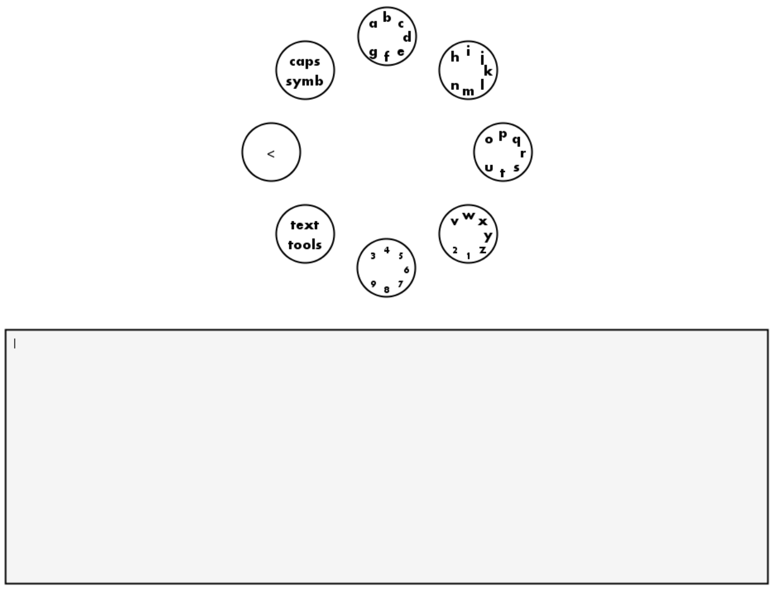
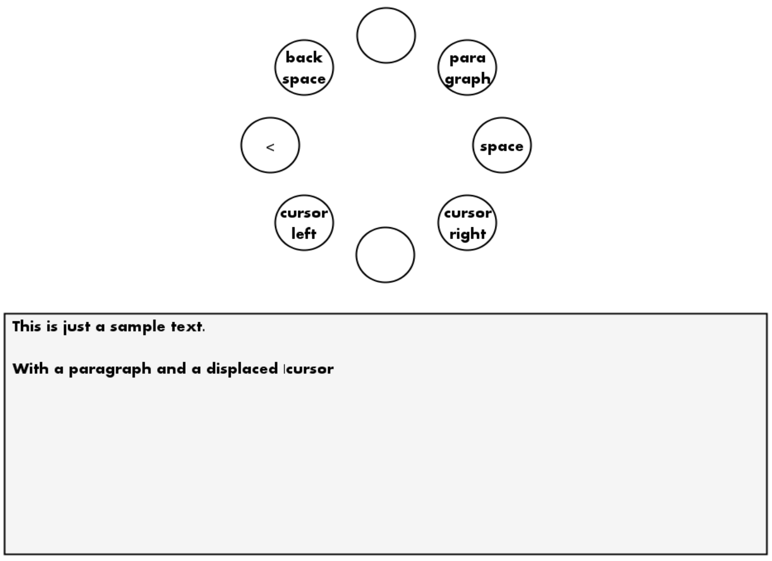


Fig. 5.50: Writer input GUI

One input is already taken for the “back” instruction, which leaves 7 inputs free to choose a character set and to choose, subsequently, a character inside that character set. This allows the user to have access to 49 different characters/instructions using only 2 input actions to reach each one of them.



This is just a sample text.

With a paragraph and a displaced |cursor

Fig. 5.51: Sample text in writer menu

In order to prove the improvement in efficacy this character placement grants the user, an evaluation was performed. The different key placements were tested for the number of input actions that each of following character strings (words, sentences and paragraphs) requires in order to be written.

word	**1**	be
	2	disability
	3	electrooculography
sentence	**1**	The quick brown fox jumped over the lazy dog
	2	The Mars Volta is a Grammy award winning American band from El Paso – Texas, founded in 2001 by guitarist Omar Rodriguez-Lopez and vocalist Cedric Bixler-Zavala
paragraph	**1**	Look again at that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "superstar," every "supreme leader," every saint and sinner in the history of our species lived there – on a mote of dust suspended in a sunbeam. The Earth is a very small stage in a vast cosmic arena. Think of the rivers of blood spilled by all those generals and emperors so that, in glory and triumph, they could become the momentary masters of a fraction of a dot. Think of the endless cruelties visited by the inhabitants of one corner of this pixel on the scarcely distinguishable inhabitants of some other corner, how frequent their misunderstandings, how eager they are to kill one another, how fervent their hatreds. Our posturings, our imagined self-importance, the delusion that we have some privileged position in the Universe, are challenged by this point of pale light. Our planet is a lonely speck in the great enveloping cosmic dark. In our obscurity, in all this vastness, there is no hint that help will come from elsewhere to save us from ourselves.

Table 3: Character strings evaluated for necessary input actions

The square keypad was considered to be optimized as well. Allowing for diagonal movements and repositioning the scroll cursor in the character "O" after each character selection. The results shown below were produced by a custom built Mathematica notebook.

		Optimized QWERTY	Square	Telephone Keypad	Concentric Keypad
word	**1**	3	4	4	4
	2	25	17	24	20
	3	44	28	42	36
sentence	**1**	103	84	96	89
	2	464	367	420	334
par.	**1**	3409	2962	3548	3058

Table 4: Comparison of necessary number of inputs for different keyboard geometries (lower is better)

The QWERTY keyboard layout has a very strong advantage given its widespread status. Most users are already quite used with its key placement. Nevertheless it presents the worst results in the previous test for all of the character strings tested besides the first and shortest one.

In some of tests the square keyboard presents a result similar or even better than the one allowed from the concentric keypad but the edge is not considered to be enough to choose it over the this later one. The proposed concentric keypad integrates much better with the GUI design, providing a more seamless experience and allows for a considerably larger number of characters and/or text tools. The square keyboard, as presented, allows for 36 characters while the concentric one allows 49 different options. Of course, in a future version the best option would be to implement both and let the user select which one he prefers. This keyboard style would also be integrated into further applications that would require written input, like a web browser or to search for media content in a media playlist.

Two very important improvements for the text input are also proposed, although not implemented. The first one would be the possibility for the user to define the preset sentences which can be selected in the preset menu. This could be achieved by having a menu in which the user would have the option to remove old ones, add new ones and reorganize them in the desired order. As of now, they have to be edited through an external editor. The second one would be the implementation of a predictive dictionary like the ones that currently exist in cell phones and major word processors or even in search engines and web browsers. These predictive text tools greatly improve writing speed for disabled people.

5.3.4 Considerations

Despite some issues of the Graphic User Interface which was designed, it has proven to be quite functional and it could provide a solid base in which to build upon a final, more complete, version. In order to reach a definite version it would be essential to do some testing directly with the people to whom this device is addressed. Although conversations were had with representatives from organizations dedicated to assistive technology counselling, unfortunately no significant trial was performed due to changes on the final deadline for the current work because of future professional obligations. Yet, the indispensable importance of having a direct user feedback for the production of a definitive version is completely acknowledged.

5.4 Casing and Hardware

Electrodes

During the course of this work non-branded disposable Ag/AgCl (silver/silver chloride) ECG electrodes were used. The electrodes consist of conducting gel, embedded in the middle of a self-adhesive pad onto which the cables clip. After the skin is cleaned, the release liner is removed and the electrodes are applied directly to the skin surface as an adhesive.

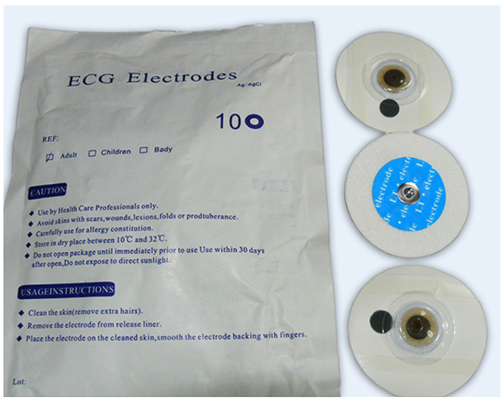

ECG Electrodes
Ag/AgCl
100
REF:
Adult
Children
Baby
CAUTION
Use by Health Care Professionals only.
Avoid skins with scars,wounds,lesions,folds or protuberance.
Carefully use for allergy constitution.
Store in dry place between 10℃ and 32℃.
Do not open package until immediately prior to use Use within 30 days after open,Do not expose to direct sunlight.
USAGEINSTRUCTIONS
Clean the skin(remove extra hairs).
Remove the electrode from release liner.
Place the electrode on the cleaned skin,smooth the electrode backing with fingers.
Lot:

Fig. 5.52: ECG Electrode pack

These are usually sold in packages of 10 or 50 for an extremely low price coming in for under 0,40€ per electrode for a small batch sale. However, they are designed for short term applications and not for long-term usage or testing. The immense amount of error this shortcoming induced in the readings was only properly evaluated towards the conclusion of the building of the device.

Had this flaw been previously diagnosed and a much greater amount of work would have been placed into searching and testing of adequate electrodes. By the end of the work it has been established that proper investment in permanent electrodes should be done despite the large step-up in price.

Electrode-to-electronics cables

The initial skin surface potential variation being read is of an extremely low order of magnitude. The captured signal is, therefore, very sensitive to all types of induced noise. Ideally, in order to prevent all types of noise, a screened shielded twisted pair (S/STP or S/FTP) stranded cable with two pairs should be used. Let us dissect this definition.

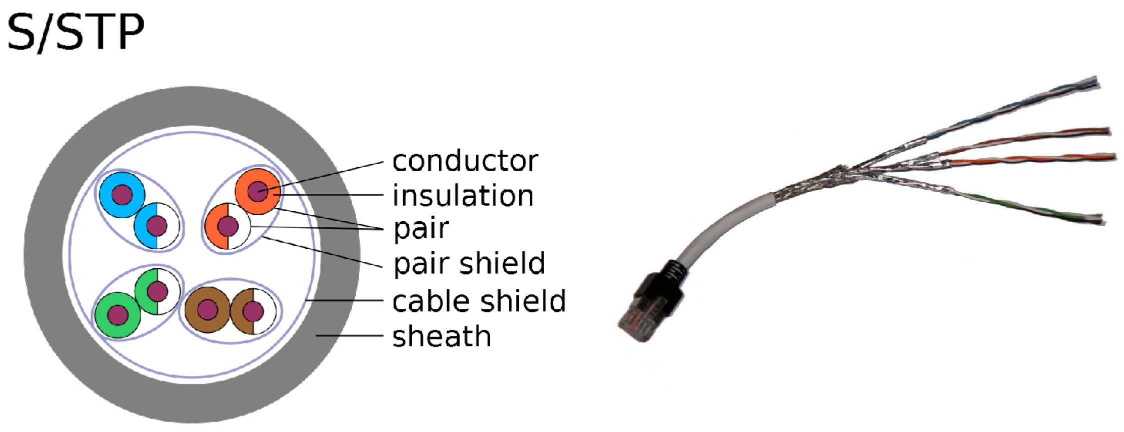


Fig. 5.53: S/STP cable with 4 pairs

Screening provides an external layer of metallic protection which may also be used as a ground, however a special ground wire is usually present as well which allows for the cable to have only two pairs (one pair for the horizontal electrodes and the other for the vertical electrodes) and using this ground wire connected to to the reference electrode. This protection prevents most electromagnetic interference. Shielding is similar to screening but the metallic cover is placed around the cable pairs, instead of externally to all the pairs. This protection eliminates alien crosstalk (crosstalk from one pair to the other) while the twisted pair cabling reduces the crosstalk between individual cables from the same pair. The cable being stranded as opposed to a solid core one allows for increased flexibility which, beyond granting additional cable resistance, provides a more pleasant touch to the user. The only requirement stranded cable demands is the usage of the proper connectors.

A typical Ethernet simple unshielded twisted pair (UTP) solid core cable with 4 pairs was used during the research and development stage due to availability in the laboratory. This cable provided satisfactory results yet, signal stability and SNR would probably improve significantly with the type of cable mentioned before.

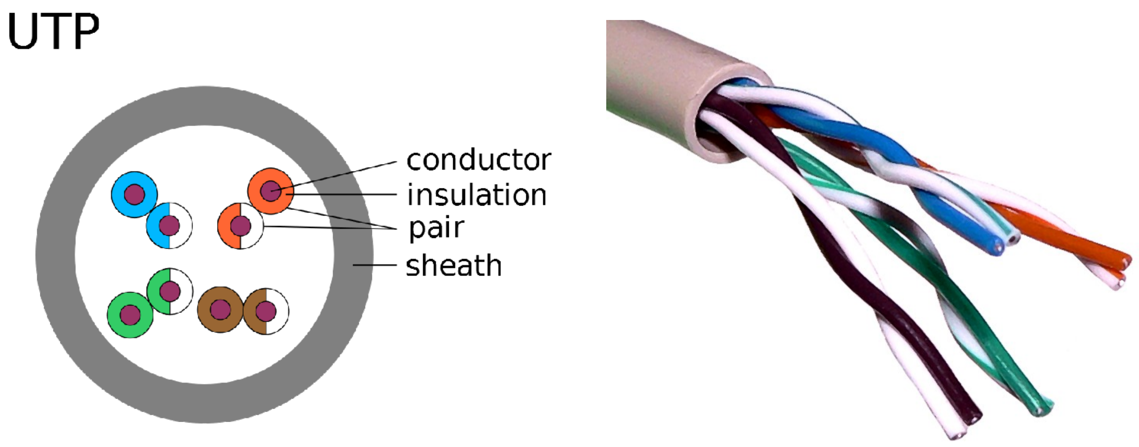


Fig. 5.54: UTP cable used during research and development

For the electrode end of the cable a set of conductive clip-on sockets compatible with the electrodes was used. In the end of the cable which connects to the electronics a simple socket was used to connect to the 5 pins of the PCB. For a final version a sturdier 5-pin connector should be used like a 5-pin DIN. A standard mini/micro USB plug is another option which would be harder and probably slightly more expensive to solder and implement but would allow for an impressively reduced size as well a more solid and modern feeling for the user.

Case

No special casing was designed for the current device as it was considered to be outside of the scope for this work. After the implementation of the electronics as a smaller stand-alone device, a case as well as some more aesthetically pleasing cables and electrodes could be created in partnership with a design team. This has proven to be of much greater importance for disabled users than one would assume. Especially in a young segment, but not exclusively, the users are incredibly more receptive to new technology when it is implemented in a way which is pleasing to the eye and less prone to accentuate their disabilities.

Electronics-to-computer cables

To connect the device to the computer a male USB-A to male USB-B cable is required. Standard USB-A is the most common USB plug which is commonly found in computers and standard USB-B is the more square-like one which can also found in numerous other devices. Three different standard male USB-A to male USB-B cables were tested and all performed according to expectations without any relevant events.

CHAPTER VI

Results and Validation

"Statistics: The only science that enables different experts
using the same figures to draw different conclusions"
Evan Esar, *Esar's Comic Dictionary*

Despite the many proposed improvements which were mentioned during the previous chapter, and will also be further explored during the next chapter, the developed device managed to fulfil the goals which were set in the beginning of the present work - to create a reliable and affordable easy-to-use eye based human-computer interface. Let us delve further into the analysis of the achievement of each of these goals.

6.1 Reliability

The system proved to be capable of delivering a stable and reliable experience, despite the fact that it was only directly tested on one subject. To better illustrate this reliability a number of accuracy tests were made were the user was asked to execute a set of specific eye movement and blinks. The images presented during this chapter were obtained with a custom-built oscilloscope application created in Processing which displays, in its default configuration, the horizontal and vertical signals after being processed by the ATMega328 as well as the detected pattern. The source code is displayed in Appendix 3.

Test 1 – Simple directions

The first test consisted of executing 20 repetitions of one of the predefined movements – UP, UP-RIGHT, RIGHT, DOWN-RIGHT, DOWN, DOWN-LEFT, LEFT, UP-LEFT and BLINK. Five different events can happen when you attempt to perform a movement to be recognized by the device:

- **Success** – The device successfully detects the correct pattern,
- **Mis-match** – The device triggers a pattern which was not the desired one,

- **Correct two signal activation** – The device detects two consecutive patterns in just one eye movement. The last pattern detected is, however, the desired one so the signal detection provides in fact a correct input,
- **Incorrect two signal activation** – The device detects two consecutive patterns in just one eye movement. The last pattern detected is, however, the wrong one so the signal detection provides an incorrect input,
- **No signal** – No signal at all is detected, despite the eye movement.

Some visual translation of what has been explained is given by the following pictures.

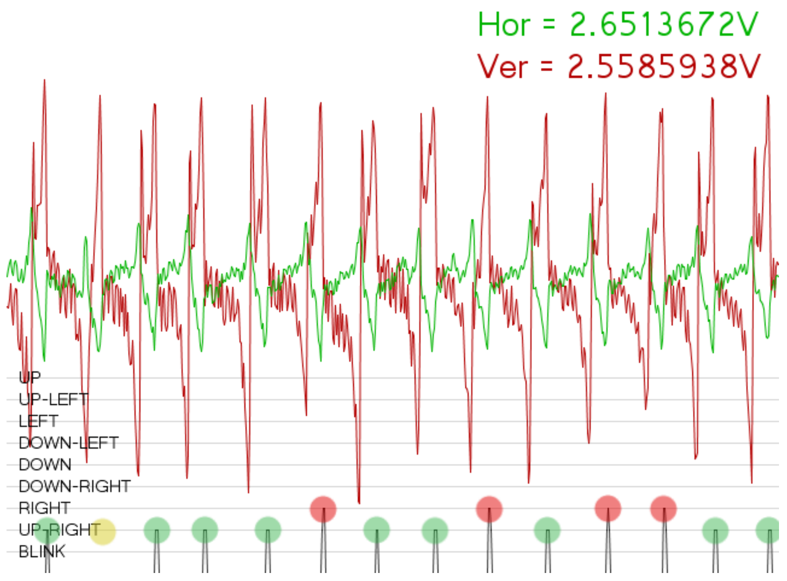


Fig. 5.1: UP-RIGHT repetition sequence: GREEN – success, RED – mismatch, YELLOW – no signal

It is important to notice that the time line should be read the opposite of a usual oscilloscope. The left-most side of the screen is always t=0s while the graphic signal keeps being "pushed" or "shifted" to the right-most side of the screen where t≈15s, unlike the usual oscilloscope method where the t=0s pointer does cyclic sweeps through the screen. This allows for a complete long time window (T≈15s) to be permanently present on-screen and arranged in a more comprehensive order.

The UP-RIGHT signal is one of the most troublesome ones, possibly due to a poor calibration, worsened by the existing problem with the horizontal gain. When Fig. 5.33 is observed and compared to the most other signal patterns, it presents a much higher standard-deviation than usual, which noticeably contributes to a worse performance when attempting to detect this pattern.

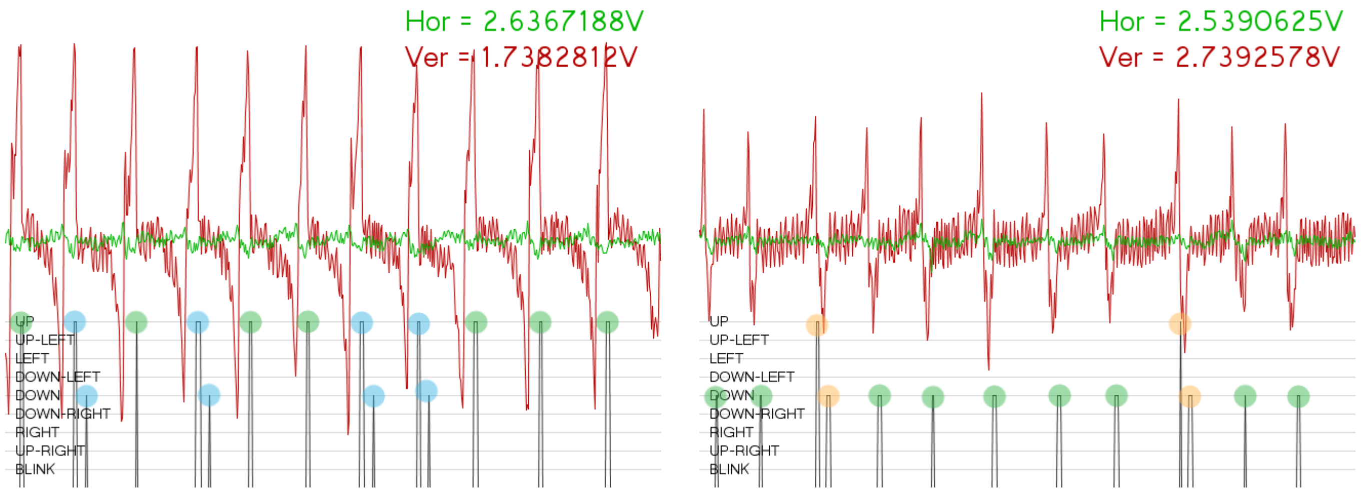


Fig. 5.2: In the left side is represented a repetition of UP movements and on the right side a repetition of DOWN movements: GREEN – success, BLUE – correct two signal, ORANGE – incorrect two signal

A good pair of eye movements to showcase the "two signal activation" type of event is the UP and DOWN movements. These are sometimes mistaken for one another, given the similarity in their signal. This does not present a problem when testing for UP saccades given that the mistake in pattern detection always produces a down before the up. When trying to produce a DOWN input it does sometime present this problem.

The complete obtained results are presented in the table below. "Correct two signal activation" events have been counted as a positive detection for percentage values given that the outcome they produce is the desired one.

	Detection success		**Detection fail**			**Success Rate**
	Success	**Correct two signal**	**Mis-match**	**Incorrect two signal**	**No signal**	
UP	14	6	0	0	0	100,00%
UP-RIGHT	15	0	4	0	1	75,00%
RIGHT	20	0	0	0	0	100,00%
DOWN-RIGHT	18	0	2	0	0	90,00%
DOWN	18	0	0	2	0	90,00%
DOWN-LEFT	20	0	0	0	0	100,00%
LEFT	16	0	0	0	4	80,00%
UP-LEFT	17	0	1	0	2	85,00%
BLINK	20	0	0	0	0	100,00%
TOTAL						**91,11%**

Table 5: System accuracy test

The accuracy achieved with the proposed device is obviously quite good when the low cost of the system along with the lack of optimization is taken into account. It is believed that the improvements proposed in the next chapter will increase this percentage considerably. The system proposed by Usakli et al.[21] is built upon a much more complex set-up of a 6 stage electronic filtering and amplification and running a much more demanding pattern detection algorithm on the computer delivers an accuracy of 95%. Obtaining an accuracy of 91% on an unoptimized system for such a low cost is a very positive result.

Test 2 – Direction followed by blink

The second test was ran in order to prove the solidity that the necessary "saccade + blink" sequence provides. As has been previously stated, the GUI requires that one of the directional buttons is selected through a directional saccade before it can be clicked with a blink. Through this method, executing a sequence of 50 random selections and clicking during which the user could correct the produced input through a new saccade before clicking it, only two errors were produced.

The first one was a non detection of a blink, causing the user to have to repeat the directional saccade and the second was a blink being produced after the system had produced a wrong pattern detection. This only happens when the user is already trusting that the system will produce a positive detection and does not even verify the chosen selection before clicking, which is, in a sense, a positive sign.

Test 3 – Two complete pattern sequences

The last test which was ran involved a simple sequence of movements going through all the defined patterns. It was done three times with great success – there were only two instances of "correct two signal activation" during the execution of the UP saccade.

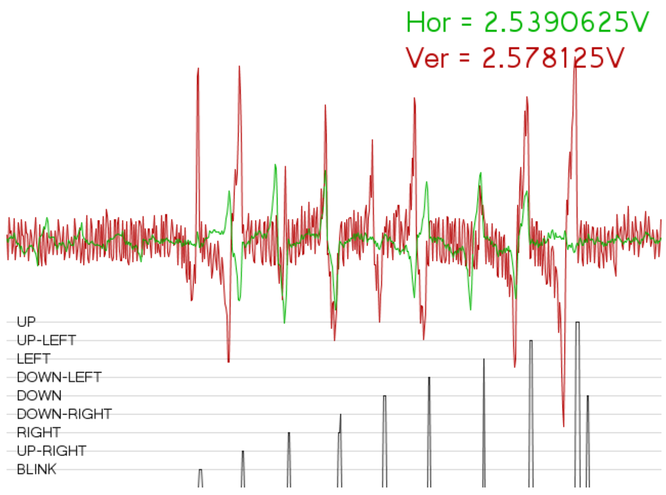


Fig. 5.3: Complete pattern sequence

Above is presented one of the complete sequences of UP, UP-LEFT, LEFT, DOWN-LEFT, DOWN, DOWN-RIGHT, RIGHT, UP-RIGHT and BLINK. This sequences were executed with a great degree of success.

Considerations

It is also important to mention that this values were obtained using the prototype with non-optimized electronics which has a lower than desired horizontal gain value and incomplete noise shielding. After the proposed modifications it is only expected that the device would be able to achieve even higher accuracy rates through more pronounced and cleaner input signals – thus sporting higher SNR.

Another positive feature was that using brand new electrodes, even given the case that they are designed to be temporary and disposable, the captured signal remains perfectly usable for a good amount of time – they have been tested positively for up to six consecutive hours. Only slight DC offset variations are present but these are easily countered by the microprocessor's DC offset correction algorithm.

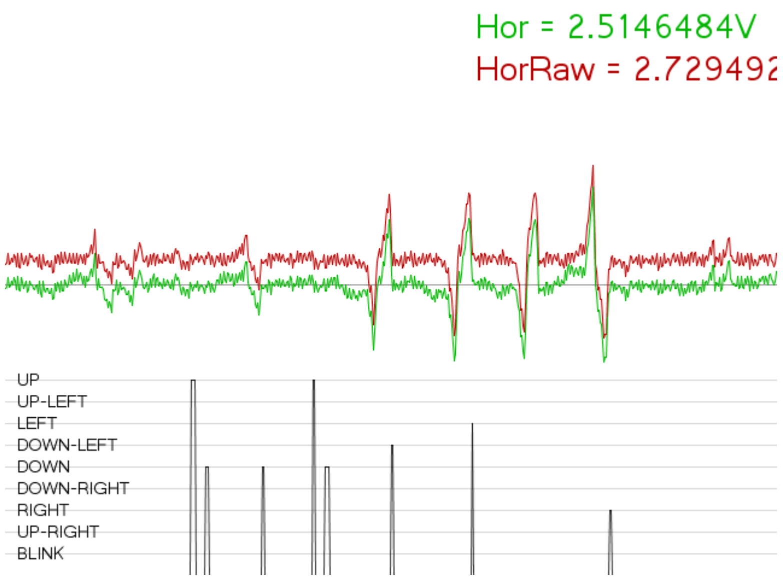


Fig. 5.4: Horizontal signals before and after offset correction

In Fig. 5.4 the horizontal signal transmitted directly from the ADC is shown in read while the signal corrected by the micro processor is shown in green. The corrected signal can be seen around the grey 2.5V reference mark. A has been explained in chapter 5.1.4, offsetting errors might cause the signal to become unreadable if it causes the operational amplifiers to enter saturation, which would only occur with an shift of more than 1V.

The long term signal stability is very good, as can be seen in Fig. 5.5 during an approximately 15s time window when the user attempted to keep the eyes as still as possible. The mains noise is more visible in the vertical channel due to its higher gain.

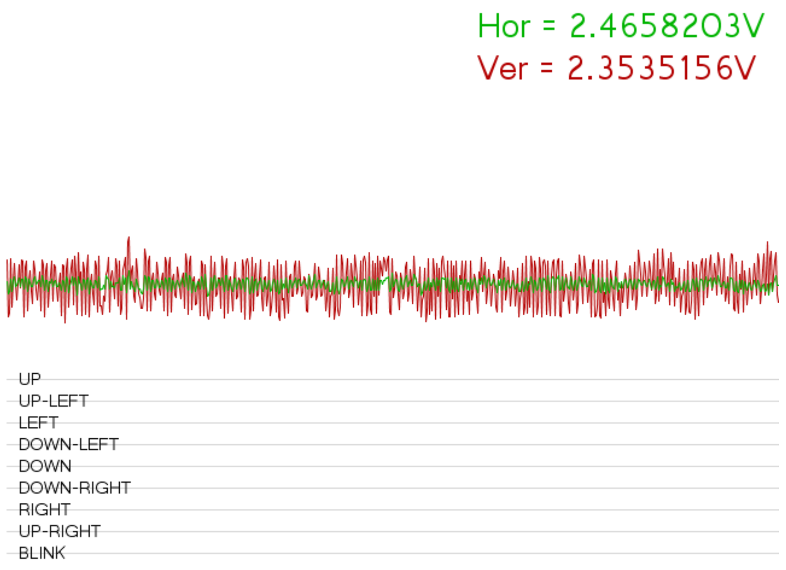


Fig. 5.5: Stable neutral signal

It is also worth mentioning that the calibration was done almost one month prior to these tests. This proves that the signal produced by eye saccades and blinks is, at least for the same user, very stable throughout a long term interval. This was tested more thoroughly by Shackel et al.[25].

6.2 Affordability

The complete device was built for an extremely low cost. Below is presented the total real cost for the one working prototype which was built.

For a pricing reference the electronics distributor website farnell was used including the 23% VAT effective in Portugal with the price list effective in February 2011. The complete amount of electrodes purchased were included simply to determine the full investment which was during the development and testing stage, but they weren't all used, so for the next tables a lower quantity will be taken into consideration. The PCB and soldering was done in the laboratory at no cost and the cable used was a leftover from an Ethernet network cable.

	Item	Quantity	Unitary price	Price
	Arduino board	1	24,350 €	24,350 €
components	470Kohm ±1% 0805 (R1, R3, R8, R10)	4	0,094 €	0,376 €
	16Kohm ±1% 0805 (R2, R9, R15, R16)	4	0,094 €	0,376 €
	2Kohm ±1% 0805 (R4, R5, R11, R12)	4	0,094 €	0,376 €
	150Kohm ±1% 0805 (R6, R7)	2	0,080 €	0,160 €
	75Kohm ±1% 0805 (R13, R14)	2	0,094 €	0,188 €
	680ohm ±1% 0805 (R17)	1	0,094 €	0,094 €
	10microF ±10% 0805 (C1, C4, C7, C8)	4	0,095 €	0,380 €
	33nF ±5% 0805 (C2, C3)	2	0,068 €	0,136 €
	68nF ±10% 0805 (C5, C6)	2	0,127 €	0,254 €
	Opamp TLC274A	2	0,750 €	1,500 €
	Socket DIP-14	2	0,166 €	0,332 €
hardware	PCB and soldering		0,000 €	0,000 €
	Electrodes (pack of 10) + shipping	4	-	13,510 €
	Cable	1,5m	0,000 €	0,000 €
	TOTAL			**42,032 €**

Table 6: Total real cost of the working prototype

In a definitive mass production stage these last two hardware elements would obviously present a production cost, nevertheless the cost for the components would lower greatly as well as the arduino board cost which could be discarded. In order to replace the arduino board with a completely stand-alone PCB, the components presented below would be necessary[41][42][43].

Components required for ATMega328 stand-alone operation:

- ATMega328
- 7805 Voltage regulator
- 2 LEDs
- 2 220 Ohm resistors
- 1 10k Ohm resistor
- 2 10 uF capacitors
- 16 MHz clock crystal
- 2 22 pF capacitors
-

Components required for USB communication (using an Atmega8U2 programmed as a USB-to-serial converter) :

- ATmega8U2
- 2 22Ohm resistor
- 1 uF capacitor
- 16 MHz clock crystal
- 2 22 pF capacitors

			100 Units		1.000 Units		10.000 Units	
	Item	**Quant.**	**Unit**	**Price**	**Unit**	**Price**	**Unit**	**Price**
ATMega stand-alone	ATMega328	1	2,980 €	2,980 €	2,980 €	2,980 €	2,980 €	2,980 €
	7805 Voltage regulator	1	0,710 €	0,710 €	0,660 €	0,660 €	0,660 €	0,660 €
	LEDs	2	0,064 €	0,128 €	0,059 €	0,118 €	0,059 €	0,118 €
	220 Ohm resistors	2	0,005 €	0,010 €	0,003 €	0,006 €	0,003 €	0,006 €
	10k Ohm resistor	1	0,005 €	0,005 €	0,003 €	0,003 €	0,003 €	0,003 €
	10 uF capacitors	2	0,056 €	0,112 €	0,026 €	0,052 €	0,026 €	0,052 €
	16 MHz clock crystal	1	0,320 €	0,320 €	0,270 €	0,270 €	0,270 €	0,270 €
	22 pF capacitors	2	0,012 €	0,024 €	0,006 €	0,012 €	0,005 €	0,010 €
USB comm	ATmega8U2	1	2,210 €	2,210 €	2,210 €	2,210 €	2,210 €	2,210 €
	22Ohm resistor	2	0,005 €	0,010 €	0,003 €	0,006 €	0,003 €	0,006 €
	1 uF capacitor	1	0,026 €	0,026 €	0,022 €	0,022 €	0,022 €	0,022 €
	16 MHz clock crystal	1	0,320 €	0,320 €	0,270 €	0,270 €	0,270 €	0,270 €
	22 pF capacitors	2	0,012 €	0,024 €	0,006 €	0,012 €	0,005 €	0,010 €
components	470Kohm ±1% 0805 (R1, R3, R8, R10)	4	0,005 €	0,020 €	0,003 €	0,012 €	0,003 €	0,012 €
	16Kohm ±1% 0805 (R2, R9, R15, R16)	4	0,005 €	0,020 €	0,003 €	0,012 €	0,003 €	0,012 €
	2Kohm ±1% 0805 (R4, R5, R11, R12)	4	0,005 €	0,020 €	0,003 €	0,012 €	0,003 €	0,012 €
	150Kohm ±1% 0805 (R6, R7)	2	0,005 €	0,010 €	0,003 €	0,006 €	0,003 €	0,006 €
	75Kohm ±1% 0805 (R13, R14)	2	0,005 €	0,010 €	0,003 €	0,006 €	0,003 €	0,006 €
	680ohm ±1% 0805 (R17)	1	0,005 €	0,005 €	0,003 €	0,003 €	0,003 €	0,003 €
	10microF ±10% 0805 (C1, C4, C7, C8)	4	0,026 €	0,104 €	0,022 €	0,088 €	0,022 €	0,088 €
	33nF ±5% 0805 (C2, C3)	2	0,011 €	0,022 €	0,006 €	0,012 €	0,005 €	0,010 €
	68nF ±10% 0805 (C5, C6)	2	0,011 €	0,022 €	0,006 €	0,012 €	0,005 €	0,010 €
	Opamp TLC274A	2	0,440 €	0,880 €	0,440 €	0,880 €	0,440 €	0,880 €
hardware	PCB printing	1	2,920 €	2,920 €	0,530 €	0,530 €	0,300 €	0,300 €
	Soldering	1	8,991 €	8,991 €	6,156 €	6,156 €	4,000 €	4,000 €
	Cable	1,5m	1,234 €	1,234 €	1,234 €	1,234 €	1,234 €	1,234 €
	Electrodes	-	-	-	-	-	-	-
	Case	-	-	-	-	-	-	-
	TOTAL			**21,137 €**		**15,585 €**		**13,190 €**

Table 7: Total production cost preview of a proposed final version

Electronic component prices vary greatly according to order quantities so three different production scales were evaluated – the prices were estimated for batch productions of 100, 1.000 and 10.000. All of the components were considered to be SMD, instead of the previous DIP packages chosen for the TLC274A operational amplifiers. This helps reduce the component cost and PCB size (further reducing cost). The chosen package type was 0805 which is a good all-

around package dimension when comparing price, size and availability but even smaller packages could be used to reduce the device size to a minimum. For the new components no particular attention was paid relatively to tolerance values although most of them were considered to be the same as the ones used for the instrumentation amplifier (±1% for the resistors and ±5% to ±10% for the capacitors) but from a lower cost range.
The prices relatively to PCB printing were obtained from the website MultiCB[52] with 5X5cm and the following saving specifications activated: no Jump scoring or Z-Axis-Milling, Ø Milling / Routing tool > 1mm, Ø Plated Through Slits > 1mm, Chamfered border only 45°.

Soldering prices were obtained from the website Screaming Circuits[53] with the following specifications: 20 total unique parts, 50 total smt parts and smt on both sides with a 20 day waiting period. They do not accept online calculation for orders over 1000, so the value for a 10,000 batch order was merely speculated, based on the price difference which exists from 100 to 1000 orders. Soldering is definitely the most expensive part of fabricating the device.

It was not possible to make a simulation for electrodes and casing pricing. Some more effort should be put into researching other electrode options as the ones tested were not the ones exactly appropriate for the work. Case prices would really depend on size, shape, materials and many other factors which would not make sense to speculate without a specific design proposal.

In the end the approximate production price obtained, which should remain below the 50€ mark even with casing and a decent amount of electrodes, is extremely low when compared to every other similar assistive technology brought into analysis, rendering it much more accessible than the typical options.

6.3 Ease-of-use

The electrodes are definitely the weak point when it comes to evaluate the ease-of-use of the proposed device. In the extreme cases of patient immobility and reduced sensitivity, to whom this work is mainly geared to, it does not pose a big problem to have large electrodes placed on the user's face. Nevertheless a less intrusive electrode system should be designed both to be able to reach a wider user base as well as providing a more comfortable experience to disabled patients.

Regarding the hardware's simplicity, this is always a somewhat subjective question to be analysed without a large number of users testing the device and reviewing it. Nevertheless the set-up process, for instance, is quite basic – simply connect the device with an USB cable to the computer, plug the electrode cables to the device and place the electrodes in the designated positions in the forehead (after proper cleansing, preferably with water, soap and alcohol as is usual in biomedical scenarios). It is yet to be established if it would be possible to do a universal user calibration which is the best scenario or if an individual user calibration will always be necessary. The calibration used for the test user involved 10 samples from 9 different patterns with a duration of 1750ms each (which are latter cropped into the smaller 700ms patterns). The minimum possible duration for the whole sample reading is therefore 2min and 38s. Even if the patient feels the need to rest his or her eyes to avoid eye straining, this process could easily be completed in approximately 5 minutes which is an acceptable time for a long-term calibration.

The graphical user interface, while not having been directly tested with the EOG module, has been shown to a wide number of users who controlled it through the keyboard keys. Reactions have been very positive and the general comment has been that while it does take some time to adapt to the eight-directional paradigm, it gets very comfortable and easy to interact with.

One test which is usually performed with similar technologies is the time which the user takes to write a specific character string.

	Time 1 (s)	Time 2 (s)	Time 3 (s)	Time 4 (s)	Time 5 (s)	Average (s)
crazy	21,13	21,57	30,87	23,22	28,28	25,01
water	18,48	21,36	22,1	21,57	20,46	20,79
hello	24,72	25,6	23,16	21,33	24,24	23,81
matter	32,94	21,5	25,29	32,36	27,21	27,86
choco	20,59	25,59	29,61	25,65	26,37	25,56
TOTAL						**24,61** ± 3,89

Table 8: Time elapsed during the writing of a 5 character string

The typical BCI through EEG[21] allows the user to write a five letter word in 105 seconds while the typical EOG allows for 25 seconds. This is very similar to the results which were obtained by the proposed device, although they were stated to be produced using electronics and processing power of much higher cost and complexity. Both the typical EOG and the one presented here presented a 100% rate of successful character selection. Therefore this system, although unoptimized obtains better results in reliability, affordability and ease-of-use when compared to equivalent EEG technologies and a better result in affordability when compared to equivalent EOG techniques.

CHAPTER VII

Conclusions and Prospects

"The problem with the future is that it keeps turning into the present"
Bill Watterson, Calvin and Hobbes

As has been mentioned and detailed in the previous chapter the main challenges taken during the initial stages of the work have been concluded with success. Despite this fact, there are still many areas where the current work could be improved. This will be addressed in this chapter alongside the possible options and prospects which lie in the future path for the proposed work.

7.1 Improvements

Any device can be permanently tweaked, nevertheless a few priority key areas should be addressed first in order to improve the quality and range of use of the proposed device. The following steps are proposed:

Electronics

- **Increasing horizontal gain** – The horizontal gain should be increased at least to the double of its current value to match the gain of the vertical channel (4350). This can be done through the replacement of the appropriate resistors and capacitors.
- **Test better filter tuning values** – It is possible that with some alternative filter cut-off frequencies and component tuning no signal reflux (see page 44) is present. This would allow for the analysis time-window to have half the duration, increasing system response time.
- **Stand-alone ATMega328** – Use the same microprocessor, which provided very good results, without the need for the Arduino board which entails a high cost. The process through which this can be achieved is mentioned in the previous chapter.
- **Design a new PCB** – Design a new PCB to correct the flaws of the prototype which have been mentioned in chapter 5.1 as well as to completely integrate the ATMega328

microprocessor. This method would provide a complete stand-alone independent PCB instead of a piggy-back.

Microcontroller

- **Gather a large database of user signals –** In order to improve accuracy ratings in pattern detection as well as to allow the device to be used by a large range of users a much more in-depth study of signal patterns should be performed. It would be desirable that a group of at least 50 people from varied ages, sex and ethnic groups would provide 20 saccade and blink repetitions to a database. This database should subsequently be analysed to determine if a global pattern collection could be created and if any necessary changes to the proposed algorithm would be necessary. This would enable the creation of a true plug-and-play device with no need for a specific individual user training program,
- or **Create a training module –** In the possible scenario that a global pattern definition is not feasible a simple individual user training program and its graphical interface should be created.

Software

- **Create mouse input and web browser –** This two tools have an incomparable importance in providing a complete computer experience as an assistive technology. The mouse allows the user to make use of every normal computer tool. It even allows the user to interact with the usual web browser, nevertheless a dedicated interface for the input paradigm that the device presents would immensely facilitate the user's web browsing experience.
- **Improve text input** – Many improvements could be done to the text input like auto-completion dictionary, exporting from "writer" menu to "pre-defined" menu, wider range of text tools (text size, color, ident, underline, bold, etc.), export to external text, and so on.
- **Create dedicated media centre** – A very useful tool which would be quite easy to implement in the graphic user interface would be a media centre integrated with the built user input paradigm. This could be achieved by allowing the control of audiovisual playlists and playback commands to be interacted with based on the 8-directional method.
- **Create a specific option for severe LIS** – Some severe LIS patients are only able to perform vertical eye movements. A new interface could be designed based on this constraint.

Hardware

- **Do an extensive electrode testing –** One of the weakest parts of the device were, without a doubt, the electrodes used. This could be solved by testing many different electrode types and configurations and determining the best ones.
- **Testing a shielded cable** – By using a shielded cable mains noise could probably be considerably reduced, improving SNR and, therefore, pattern detection accuracy.
- **Design and build of a casing and dedicated cables** – As mentioned before, an aesthetically pleasing design could go along way in facilitating a product acceptance near the big public. Not to mention that any electronic device needs a proper cover to insure its safety and its proper lifetime.

7.2 Prospects

Any new device which is researched can usually follow three paths: to be developed commercially, to be released into the open source community and to be forgotten into oblivion.

Both from an open source and a commercial point of view the device has the competence to be successful. As an open source project it is cheap to buy, relatively simple to assemble, powerful and would probably arouse some interest and curiosity, one of the biggest propellents of the open source community. As a commercial device it is enough to say that it provides the same capabilities that some other devices provide, although at an extremely low production cost which gives it a safe margin of profit.

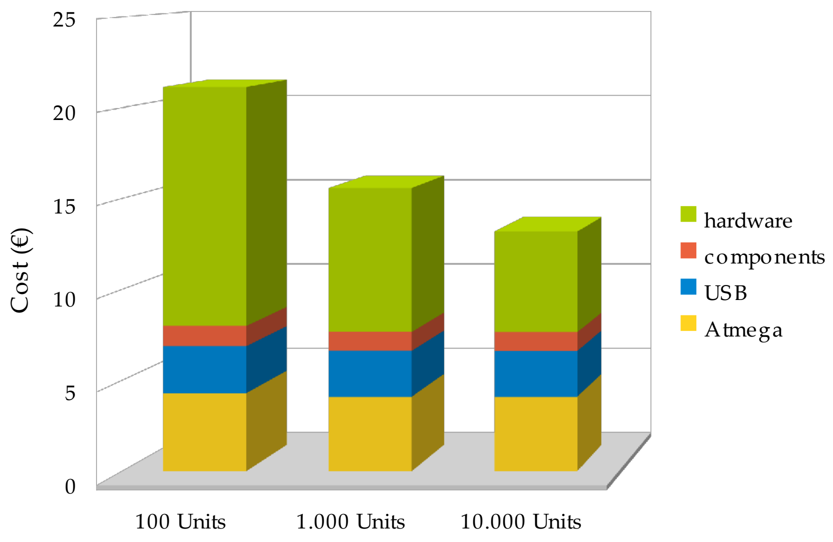


Fig. 7.1: Stacked production costs for various batch quantities

The picture above represents the different cost contributions for each part of the device in different batch production amounts, as has been explained in the previous chapter (table 8). The hardware part is obviously the one with the higher costs as well as the largest margin for price dropping in a hypothetical large scale production. This quantities are probably not realistic for the device to be sold in its full three module setup (hardware + processor + software) as an assistive technology given the relatively low amount of people who suffer from such extreme motor disabilities. It is, nevertheless, realistic to envision the device being mass produced simply as an "eye joystick" for one of the many other applications that it could be of use as mentioned in the final section of Chapter 3 – Going Beyond.

7.3 Final considerations

Given the positive results obtained throughout the course of this thesis it is believed that the proposed work has the possibility of having some impact not only from a technological standpoint, but also in effectively improving the living conditions of disabled patients.

Bibliography

Books:

[1] Emil Du Bois-Reymond. *Untersuchungen Uber Thierishce Elektricität*. Verlag Von G. Reimer (1849).

[2] Neil A. Campbell, Jane B. Reece. *Biology, 8th Edition*. Pearson Education (2009).

[3] Arthur C. Guyton MD, John E. Hall PhD. *Textbook of Medical Physiology, 11th Edition*. Saunders (2005).

[4] Alliance for Technology Access. *Computer Resources For People With Disabilities, 4th Edition*. Hunter House (2004).

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer (2006)

[6] Jerome B. Posner, Clifford B. Saper, Nicholas Schiff, Fred Plum. *Plum and Posner's Diagnosis of Stupor and Coma, 4th Edition*. OUP USA (2007).

[7] Joshua Noble. *Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFrameworks*. O'Reilly Media (2009).

[8] Joe Clark. *Building Accessible Websites*. Joe Clark (2008) [Online Edition].

Papers:

[9] B. Shackel. *Pilot Study In Electro-oculography*. EMI Electronics Ltd. (1959).

[10] Chang-Chia Liu, W. Art Chaovalitwongse, Panos M. Pardalos, Onur Seref, Petros Xanthopoulos, J.C. Sackellares, Frank M. Skidmore. *Quantitative Analysis On Electrooculography (Eog) For Neurodegenerative Disease*. Optima Neuroscience, Inc.

[11] F.C.C. Riemslag, H.F.E. Verduyn Lunel & H. Spekreijse. *The Electrooculogram: A Refinement Of The Method*. Documenta Ophthalmologica (1990).

[12] Duane Denney And Colin Denney. *The eye blink electro-oculogram*. Department of Psychiatry, Oregon Health Sciences University (1984).

[13] Peter Barfoot. *Electrooculography*. BioPortfolio (2010).

[14] Toral Zaveri, Jason Winters, Mamta Wankhede, Il Park. *A fast and accurate method for discriminating five choices with EOG.*

[15] M. Trikha, A. Bhandari, T. Gandhi. *Automatic Electrooculogram Classification for Microcontroller Based Interface Design*. (2007).

[16] Arie E. Kaufman, Amit Bandopadhay, Bernard D. Shaviv. *An Eye Tracking Computer User Interface.* State University of New York at Stony Brook.

[17] David Grant. *An Integrated Human Computer Interface using Eye Gaze Tracking and Facial Feature Recognition*. University of Plymouth (1998).

[18] Rafael Barea, Luciano Boquete, Manuel Mazo, Elena López, L.M. Bergasa. Electrooculographic Guidance Of A Wheelchair Using Eye Movements Codification. Electronics Department, University of Alcala.

[19] Andreas Bulling, Daniel Roggen and Gerhard Tröster. *EyeMote - Towards Context-Aware Gaming Using Eye Movements Recorded From Wearable Electrooculography*. Wearable Computing Laboratory, ETH Zurich.

[20] Andreas Bulling, Daniel Roggen and Gerhard Tröster. *It's in Your Eyes - Towards Context-Awareness and Mobile HCI Using Wearable EOG Goggles*. Wearable Computing Laboratory, ETH Zurich (2008).

[21] A. B. Usakli, S. Gurkan, F. Aloise, G. Vecchiato, F. Babiloni. *On the Use of Electrooculogram for Efficient Human Computer Interfaces*. Hindawi Publishing Corporation (2010).

[22] G. B. Arden, J. H. Kelsey. *Changes Produced By Light In The Standing Potential Of The Human Eye*. Medical Research Council Ophthalmological Research Unit, Institute of Ophthalmology (1961).

[23] Louis D. Homer, Hansjorg E. J. W. Kolder. *The Oscillation Of The Human Corneoretinal Potential At Different Light Densities*. Department Of Physiology, Emory University (1967).

[24] B. Estrany, P. Fuster, A. Garcia, Y. Luo. *Human Computer Interface by EOG tracking*. PETRA'08 (2008)

[25] B. Shackel, J. R. Davis. *A Second Survey With Electro-oculography*. EMI Electronics Ltd. (1959).

[26] Arne John Glenstrup, Theo Engell-Nielsen. *Eye Controlled Media: Present and Future State.* University of Copenhagen (1995).

[27] Bernardo Dal Seno, Matteo Matteucci, LucaMainardi. *Online Detection of P300 and Error Potentials in a BCI Speller*. Hindawi Publishing Corporation (2010)

[28] Armando B. Barreto, PhD ; Scott D . Scargle, MSEE; Malek Adjouadi, PhD. *A Practical EMG-Based Human-Computer Interface For Users With Motor Disabilities*. Department of Electrical and Computer Engineering, Florida International University (2000).

[29] David W. Patmore, R. Benjamin Knapp. *Towards an EOG-Based Eye Tracker for Computer Control*. ACM Inc. (1998).

[30] *Op Amps for Everyone*. Texas Instruments (2008)

[31] Anil K. Jain, Robert P.W. Duin, Jianchang Mao, Senior Member. *Statistical Pattern Recognition: A Review. Transactions On Pattern Analysis And Machine Intelligence* (2000)

[32] Heiko Drewes. *Eye Gaze Tracking for Human Computer Interaction.* Ludwig-Maximilians-Universität München (2010).

[33] Robert J.K. Jacob. *Eye Movement-Based Human-Computer Interaction Techniques: Toward Non-Command Interfaces*. Human-Computer Interaction Lab, Naval Research Laboratory.

[34] *Ideal Op-Amp Circuits.*

[35] Robert J.K. Jacob, Manuel A. Perez-Quinones, Linda E. Sibert, Vildan Tanriverdi, James N. Templeman. *What you look at is what you get: Eye movement-based object selection*. Naval Research Laboratory (2004).

[36] Gerwin Schalk, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer, Jonathan R.Wolpaw. BCI2000: *A General-Purpose Brain-Computer Interface (BCI) System*. Transactions On Biomedical Engineering (2004).

Datasheets:

[37] *LM124, LM224, LM324 - Low Power Quad Operational Amplifiers*. STMicroelectronics (2010).

[38] *TL081, TL081A, TL081B, TL082, TL082A, TL082B, TL084, TL084A, TL084B – JFET-Input Operational Amplifiers.* Texas Instruments (2004).

[39] *TLC274, TLC274A, TLC274B, TLC274Y, TLC279 – LinCMOS Precision Quad Operational Amplifiers*. Texas Instruments (2001).

[40] *TLC225x, TLC225xA Advanced LinCMOS - RAIL-TO-RAIL VERY LOW-POWER OPERATIONAL AMPLIFIER*. Texas Instruments (2001).

[41] *Arduino UNO Reference Design*. Arduino (2010)

[42] *8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash – Atmega48A, Atmega48PA, Atmega88A, Atmega88PA, Atmega168A, Atmega168PA, Atmega328, ATmega328P*. ATMEL (2010)

[43] *8-bit AVR Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller – Atmega8U2, Atmega16U2, Atmega32U2*. ATMEL (2010)

Online:

[44] Epoc Emotiv Review -
http://www.joystiq.com/2010/01/27/review-emotiv-epoc-tough-thoughts-on-the-new-mind-reading-cont/

[45] Brain-Fed switch a world first-
http://www.focusonals.com/brain-fed_switch_a_world_first.htm

[46] Magic Eye - http://www.magickey.ipg.pt/magic_eye.asp

[47] Inclusive Technology - http://www.inclusive.co.uk

[48] BioControl Systems - http://www.biocontrol.com/

[49] Makezine Blog - http://blog.makezine.com/archive/2010/12/how-to-eyewriter-20.html

[50] Farnell - http://farnell.com/

[51] Wikipedia Article: Most Common English Words -
http://en.wikipedia.org/wiki/Most_common_words_in_English

[52] MultiCB - http://www.multi-circuit-boards.eu

[53] Screaming Circuits - http://www.screamingcircuits.com/

Appendixes

Appendix 1 – ATMega328 Arduino source code

```
///////////////////////////////////////////
//
//
//     Electrooculogram signal interpreter
//                    by
//                João Bárcia
//
//
//
//             26 December 2010
//
//
//
//////////////////////////////////////////

// For ease of read define Horizontal and Vertical
#define HORIZONTAL 0
#define VERTICAL 1

// Include Timer One interrupt helper library
#include "TimerOne.h"
// Include PROGMEM library to store comparison matrixes
#include <avr/pgmspace.h>

byte const nMax=50;              // The maximum index value for the Value Buffer
int valBuffer[2][nMax];          // Value Buffer where current readings are stored for average
int valBufferOffseted[2][nMax];  // Value Buffer where offseted current readings are stored for analysis
int currentSignal=0;             // What signal is going on now? (0=neutral)
long dcOffset[2];                // Stores current dcOffset
int n=0;                         // Current index
int shiftedBufferIndex=0;        // Index for comparison

int minThreshold=550;
int minDistance=minThreshold;

boolean printInfo=false;         // Flag which defines if info should be printed or not
boolean firstAverage=true;       // Flag which defines if the average has not yet been calculated, in order to
store the 50 (nMax) first values

long getHorFromFlash;            // Get the Horizontal values from Flash to RAM
long getVerFromFlash;            // Get the Vertical values from Flash to RAM

// Distance values
long distanceBlink[2], distanceUp[2], distanceUpRight[2], distanceRight[2], distanceDownRight[2],
distanceDown[2], distanceDownLeft[2], distanceLeft[2], distanceUpLeft[2];           // Euclidean Hor and
Ver distances being calculated
long calculatedDistances[9];            // Euclidean total distances to each signal

float horBoost = 4;              // Enhances the importance the horizontal distance has in the calculated
distance, due to the low amplitude of the si
al

long averageForOffset[2];

// Comparison Matrixes
prog_uint16_t comparisonBlink[2][nMax] PROGMEM = {
  {
    524,514,503,527,536,524,537,546,522,513,523,505,490,508,509,492,507,521,506,507,525,513,500,5
19,519,499,511,523,504,505,522,511,500,519,519,500,512,524,505,505,523,512,499,518,519,499,511,
522,503,504          }
  ,
  {
    496,555,582,639,813,933,924,920,934,859,774,750,662,552,526,497,410,388,413,364,341,393,386,3
51,393,420,381,397,449,416,403,460,454,419,461,485,441,455,504,469,452,507,497,457,498,519,471,
484,531,494          }
};
prog_uint16_t comparisonUp[2][nMax] PROGMEM = {
  {
    514,516,516,516,520,524,522,524,523,520,520,521,520,520,519,521,518,519,520,518,517,518,517,5
16,517,517,515,515,514,511,509,509,508,505,506,508,506,508,510,509,510,512,511,510,512,513,509,
511,513,512          }
   ,
  {
    518,524,540,609,739,832,871,885,880,855,843,828,805,797,789,773,762,757,739,719,704,691,660,6
31,600,572,544,530,503,464,410,347,299,275,284,286,284,305,314,317,331,350,355,366,382,384,388,
404,410,413          }
};
prog_uint16_t comparisonUpRight[2][nMax] PROGMEM = {
  {
    524,513,494,527,551,550,588,628,617,617,642,626,600,617,615,581,588,602,573,563,585,569,545,5
63,562,528,536,550,521,512,533,514,487,503,495,453,452,459,422,413,441,431,412,438,445,420,434,
456,434,435          }
   ,
  {
    497,544,530,520,632,729,754,810,851,793,750,766,723,672,710,712,676,686,724,687,672,703,685,6
44,669,682,637,645,661,596,572,603,567,525,560,557,487,473,479,417,380,401,367,331,380,388,339,
348,375,340          }
};
prog_uint16_t comparisonRight[2][nMax] PROGMEM = {
  {
    489,518,531,512,554,618,628,651,706,702,676,697,693,646,644,657,617,596,619,595,559,578,580,5
40,546,561,511,475,479,429,366,369,366,331,346,381,366,369,414,413,399,439,455,426,444,477,454,
452,489,478          }
   ,
  {
    528,487,530,563,546,603,688,676,658,689,644,555,550,537,463,460,494,451,430,476,465,426,469,4
93,447,467,512,471,456,508,493,453,493,515,465,477,522,484,469,520,509,470,510,528,477,492,537,
499,483,535          }
};
prog_uint16_t comparisonDownRight[2][nMax] PROGMEM = {
  {
    483,508,547,517,536,615,616,603,665,675,625,643,667,609,590,627,590,549,587,584,525,548,576,5
25,518,564,533,490,524,512,444,447,455,388,370,415,393,373,432,444,406,443,482,441,445,499,475,
446,498,503          }
   ,
  {
    551,495,510,551,508,484,527,501,437,455,445,353,339,363,305,267,318,309,263,316,352,313,348,4
12,384,377,449,460,453,549,628,634,706,788,761,734,766,721,645,665,668,596,600,636,581,554,605,
583,533,574          }
};
prog_uint16_t comparisonDown[2][nMax] PROGMEM = {
  {
    519,502,514,526,509,507,522,511,497,513,513,495,505,516,499,499,517,507,496,514,514,498,508,5
20,503,503,520,510,499,515,518,502,512,523,506,505,521,511,499,516,517,499,509,521,504,503,520,
511,499,517          }
   ,
  {
    530,516,471,472,464,400,374,367,318,282,282,271,245,246,266,246,251,291,284,282,331,346,337,3
75,409,388,409,481,557,657,803,885,891,898,874,799,767,764,712,672,682,661,618,626,634,588,583,
607,579,556          }
};
prog_uint16_t comparisonDownLeft[2][nMax] PROGMEM = {
  {
    516,491,490,485,446,428,428,404,386,398,397,383,401,417,407,418,443,437,436,462,469,456,471,4
86,471,475,496,485,479,501,511,517,559,604,615,638,661,644,623,628,613,584,586,588,562,558,570,
552,540,554          }
   ,
  {
    513,510,491,484,471,430,403,394,355,314,314,294,277,272,281,279,273,296,302,304,330,343,347,3
74,397,393,402,429,427,429,472,536,617,732,828,861,867,854,799,751,734,705,668,659,650,622,611,
618,599,581          }
};
prog_uint16_t comparisonLeft[2][nMax] PROGMEM = {
  {
    505,503,485,457,444,432,410,400,399,385,376,385,388,386,402,416,414,424,441,441,445,463,468,4
64,477,486,484,497,523,537,554,584,601,600,609,617,611,614,621,614,610,617,607,587,584,581,564,
559,562,551          }
   ,
  {
    520,543,586,588,592,609,594,572,570,567,535,523,522,496,482,489,479,462,471,481,464,471,492,4
81,478,500,504,503,527,548,540,548,563,545,532,542,533,513,524,533,515,517,529,512,507,524,522,
507,517,526          }
};
prog_uint16_t comparisonUpLeft[2][nMax] PROGMEM = {
  {
    485,477,472,447,426,424,412,396,403,409,402,410,427,425,426,444,448,444,460,469,463,470,486,4
81,479,493,493,485,499,509,506,520,545,552,563,588,595,590,600,602,589,590,598,587,581,592,585,
568,570,568          }
  ,
  {
    558,575,612,696,745,751,785,811,778,757,762,724,688,695,679,639,644,648,616,606,625,603,579,6
00,595,569,579,592,563,557,580,554,539,562,544,477,447,435,398,396,415,384,355,372,365,342,355,
378,365,370          }
};



void setup() {
  // Initialize Serial communication
  Serial.begin(115200);
  // Set pins 0 and 1 as input
  pinMode(HORIZONTAL, INPUT);
  pinMode(VERTICAL, INPUT);

  Timer1.initialize(14000);          // initialize timer1, and set a 14ms period
  Timer1.attachInterrupt(collectData);  // attaches collectData() as a timer overflow interrupt

  // Reset Value Buffer
  for(int i=0; i<nMax; i++){
    valBuffer[HORIZONTAL]=0;
    valBuffer[VERTICAL]=0;
  }
  // Reset dcOffset
  dcOffset[HORIZONTAL]=0;
  dcOffset[VERTICAL]=0;

}

void collectData() {
  // Read input values
  int valHor=analogRead(HORIZONTAL);
  int valVer=analogRead(VERTICAL);

  averageForOffset[HORIZONTAL]=0;
  averageForOffset[VERTICAL]=0;

  //Average algorithm
  //if(!firstAverage){
  for(int i=0; i<nMax; i++){
    averageForOffset[HORIZONTAL]+=valBuffer[HORIZONTAL][n];
    averageForOffset[VERTICAL]+=valBuffer[VERTICAL][n];
  }
  averageForOffset[HORIZONTAL]=512-averageForOffset[HORIZONTAL]/nMax;
  averageForOffset[VERTICAL]=512-averageForOffset[VERTICAL]/nMax;

  if(averageForOffset[HORIZONTAL]>dcOffset[HORIZONTAL]) dcOffset[HORIZONTAL]+=1;
  else if(averageForOffset[HORIZONTAL]<dcOffset[HORIZONTAL]) dcOffset[HORIZONTAL]-=1;
  if(averageForOffset[VERTICAL]>dcOffset[VERTICAL]) dcOffset[VERTICAL]+=1;
  else if(averageForOffset[VERTICAL]<dcOffset[VERTICAL]) dcOffset[VERTICAL]-=1;
  //}

  // Send raw captures values to valBuffer to calculate averages
  valBuffer[HORIZONTAL][n]=valHor;
  valBuffer[VERTICAL][n]=valVer;

  // Send to Value Buffer the value of the reading, with respective offset
  valBufferOffseted[HORIZONTAL][n]=valHor+dcOffset[HORIZONTAL];
  valBufferOffseted[VERTICAL][n]=valVer+dcOffset[VERTICAL];

  // Test value buffer for perfect neutral
  //  valBufferOffseted[HORIZONTAL][n]=512;
  //  valBufferOffseted[VERTICAL][n]=512;

  // Reset distances
  distanceBlink[0]=0;
  distanceBlink[1]=0;
  distanceUp[0]=0;
  distanceUp[1]=0;
  distanceUpRight[0]=0;
  distanceUpRight[1]=0;
  distanceRight[0]=0;
  distanceRight[1]=0;
  distanceDownRight[0]=0;
  distanceDownRight[1]=0;
  distanceDown[0]=0;
  distanceDown[1]=0;
  distanceDownLeft[0]=0;
  distanceDownLeft[1]=0;
  distanceLeft[0]=0;
  distanceLeft[1]=0;
  distanceUpLeft[0]=0;
  distanceUpLeft[1]=0;

  for (int i=0; i<50; i++) {
    shiftedBufferIndex=n+i;
    if(i+n>nMax-1) shiftedBufferIndex=n+i-nMax;
    getHorFromFlash = pgm_read_word_near(&(comparisonBlink[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonBlink[VERTICAL]));
    distanceBlink[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceBlink[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonUp[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonUp[VERTICAL]));
    distanceUp[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-getHorFromFlash);
    distanceUp[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonUpRight[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonUpRight[VERTICAL]));
    distanceUpRight[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceUpRight[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonRight[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonRight[VERTICAL]));
    distanceRight[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceRight[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonDownRight[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonDownRight[VERTICAL]));
    distanceDownRight[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceDownRight[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-
getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonDown[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonDown[VERTICAL]));
    distanceDown[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceDown[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonDownLeft[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonDownLeft[VERTICAL]));
    distanceDownLeft[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceDownLeft[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);

    getHorFromFlash = pgm_read_word_near(&(comparisonLeft[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonLeft[VERTICAL]));
    distanceLeft[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceLeft[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);
    getHorFromFlash = pgm_read_word_near(&(comparisonUpLeft[HORIZONTAL]));
    getVerFromFlash = pgm_read_word_near(&(comparisonUpLeft[VERTICAL]));
    distanceUpLeft[HORIZONTAL]+=sq(valBufferOffseted[HORIZONTAL][shiftedBufferIndex]-
getHorFromFlash);
    distanceUpLeft[VERTICAL]+=sq(valBufferOffseted[VERTICAL][shiftedBufferIndex]-getVerFromFlash);
  }

  // AQUI CALCULAR AS DISTANCIAS SOMADAS DE TODOS
  calculatedDistances[0]=sqrt(distanceUp[HORIZONTAL]*horBoost+distanceUp[VERTICAL]/horBoost);
  calculatedDistances[1]=sqrt(distanceUpRight[HORIZONTAL]*horBoost+distanceUpRight[VERTICAL]/horB
oost);
  calculatedDistances[2]=sqrt(distanceRight[HORIZONTAL]*horBoost+distanceRight[VERTICAL]/horBoost);
  calculatedDistances[3]=sqrt(distanceDownRight[HORIZONTAL]*horBoost+distanceDownRight[VERTICAL]
/horBoost);
  calculatedDistances[4]=sqrt(distanceDown[HORIZONTAL]*horBoost+distanceDown[VERTICAL]/horBoost)
;
  calculatedDistances[5]=sqrt(distanceDownLeft[HORIZONTAL]*horBoost+distanceDownLeft[VERTICAL]/ho
rBoost);
  calculatedDistances[6]=sqrt(distanceLeft[HORIZONTAL]*horBoost+distanceLeft[VERTICAL]/horBoost);
  calculatedDistances[7]=sqrt(distanceUpLeft[HORIZONTAL]*horBoost+distanceUpLeft[VERTICAL]/horBoos
t);
  calculatedDistances[8]=sqrt(distanceBlink[HORIZONTAL]*horBoost+distanceBlink[VERTICAL]/horBoost);

  //  // AQUI CALCULAR QUAL O MENOR
  currentSignal=9;
  minDistance=minThreshold;


  if(calculatedDistances[8]<420) currentSignal=8;

  minDistance=580;
  for (int i=0; i<8; i=i+4){
    if (minDistance>calculatedDistances){
      minDistance = calculatedDistances;
      currentSignal = i;
    }
  }

  minDistance=700;
  for (int i=1; i<8; i=i+2){
    if (minDistance>calculatedDistances){
      minDistance = calculatedDistances;
      currentSignal = i;
    }
  }
  minDistance=780;
  for (int i=2; i<8; i=i+4){
    if (minDistance>calculatedDistances){
      minDistance = calculatedDistances;
      currentSignal = i;
    }
  }


  // increment and loop if the end is reached
  if(n<nMax-1) n+=1;
  else n=0;

  // Set print flag to true
  printInfo=true;
}

void loop() {
  // Print each data collection information
  if(printInfo){
    //  Serial.print("blah");
    //  Serial.print(",");
    //  Serial.print(valBufferOffseted[HORIZONTAL][n]);
    //  Serial.print(",");
    //  Serial.print(valBufferOffseted[VERTICAL][n]);
    //  Serial.print(",");
    //  Serial.print(currentSignal);
    //  Serial.print(",");
    //  Serial.println("|");
    if(currentSignal!=9){
      Serial.print(currentSignal);
      Serial.println(" ");
    }
    printInfo=false;
  }
}
```

Appendix 2 – Graphic User Interface Processing source code

Main

```
//////////////////////////////////////////////
//
//          Eight-directional
//         eye based Interface
//
//           by  João Bárcia
//        joaobarcia@gmail.com
//
//
/////////////////////////////////////////////

// imports serial lib
import processing.serial.*;
Serial arduinoPort;

// imports sound lib
import ddf.minim.*;
AudioPlayer aid;
Minim minim;

// font settings
PFont myFont;
PFont symbolFont;

// defines if input is through serial or keyboard
boolean serialInput=true;

// which button is currently selected - 8 is none for now
int selectedButton=8;
//which menu is currently selected and where it's buttons lead to. 0 is not used
int currentMenu=1;

boolean pushButton=false;

// declaration of variables related to aid menu
boolean aidStart = true, flashCounterUp=true;
int aidFlashCounter=0;

// chosen preset sentence, preset sentence being shown and imported preset strings
int currentPreset=1;
int showPreset=0;
String[] presetText1, presetText2, presetText3;

// writer settings
char alphabet[] =
{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','1','2','3','4','5','6','7','8','9'};
char alphabetCaps[] =
{'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','1','2','3','4','5','6','7','8',
'9'};
char symbols[] =
{'.',',',';',':','-','+','!','?','(',')','*','<','>','@','"','#','%','&','/','$','«','»','=','º','ª','{','1','2','3','4','5','6','7','8','9'}
;
int currentAlphabetType=0;
int textCursor=0;
int writerPanel=0; //writerPanel: 0-main, 1-abcde, 2-...
StringBuilder userText = new StringBuilder("");
char alphabetFull[][]= new char[3][26];

//visual settings
int buttonRadius=140, buttonScattering=280;

//defines which number is attributed to which menu, for ease of programme read
// Everytime you add a new menu, you must change menuDefinitions and menuTitle instancing and definiti
```

```
ons
int menuMain=1, menuHelp=2, menuMessage=3, menuPredf=4, menuPredf1=5, menuPredf2=6,
menuPredf3=7, menuWriter=8, menuWeb=9, menuMouse=10;

// the array where each menu's defns and titles are stored
int[][] menuDefinitions = new int[11][8]; //[number of menus][numer of choices in each menu]
String[] menuTitle = new String[11]; //[number of menus]

//the counter that specifies when selection is nullified and the
//number of loops it will hold the selection
int loopCounter=0, holdTime=80;

void setup() {
  // Visual definitions
  size(1024,768);
  smooth();

  // Serial communication definitions
  println(Serial.list());
  arduinoPort = new Serial(this, Serial.list()[1], 115200);

  // audio definitions
  minim = new Minim(this);
  aid = minim.loadFile("7NL10002.mp3", 2048);
  //aid.loop();
  //aid.pause();

  myFont = createFont("Aharoni Negrito", 80, true, symbols);
  symbolFont = loadFont("LaGirouette-70.vlw");
  //symbolFont = loadFont("Aharoni-Bold-48.vlw");
  textFont(myFont);
  textAlign(CENTER, CENTER);

  //temp test setups | MUST CREATE EXTERNAL EDITOR IMPORT - not
  menuTitle[1]="main";
  menuTitle[2]="aid";
  menuTitle[3]="message";
  menuTitle[4]="preset";
  menuTitle[5]="set 1";
  menuTitle[6]="set 2";
  menuTitle[7]="set 3";
  menuTitle[8]="writer";
  menuTitle[9]="web";
  menuTitle[10]="mouse";


  //GLOBAL MENU NAVIGATION
  menuDefinitions[menuMain][0]=menuHelp;
  menuDefinitions[menuMain][2]=menuMessage;
  menuDefinitions[menuMain][4]=menuMouse;
  menuDefinitions[menuMain][6]=menuWeb;

  menuDefinitions[menuHelp][6]=menuMain;

  menuDefinitions[menuMessage][0]=menuWriter;
  menuDefinitions[menuMessage][2]=menuPredf;
  menuDefinitions[menuMessage][6]=menuMain;

  menuDefinitions[menuPredf][0]=menuPredf1;
  menuDefinitions[menuPredf][2]=menuPredf2;
  menuDefinitions[menuPredf][4]=menuPredf3;
  menuDefinitions[menuPredf][6]=menuMessage;

  menuDefinitions[menuPredf1][0]=1;
  menuDefinitions[menuPredf1][2]=1;
  menuDefinitions[menuPredf1][4]=1;
  menuDefinitions[menuPredf1][6]=menuPredf;

  menuDefinitions[menuPredf2][0]=1;
  menuDefinitions[menuPredf2][2]=1;
  menuDefinitions[menuPredf2][4]=1;
  menuDefinitions[menuPredf2][6]=menuPredf;
  menuDefinitions[menuPredf3][0]=1;
  menuDefinitions[menuPredf3][2]=1;
  menuDefinitions[menuPredf3][4]=1;
  menuDefinitions[menuPredf3][6]=menuPredf;

  menuDefinitions[menuWriter][0]=1;
  menuDefinitions[menuWriter][1]=1;
  menuDefinitions[menuWriter][2]=1;
  menuDefinitions[menuWriter][3]=1;
  menuDefinitions[menuWriter][4]=1;
  menuDefinitions[menuWriter][5]=1;
  menuDefinitions[menuWriter][6]=1;
  menuDefinitions[menuWriter][7]=1;

  menuDefinitions[menuWeb][6]=menuMain;

  menuDefinitions[menuMouse][6]=menuMain;

  presetText1 = loadStrings("preset1.txt");
  presetText2 = loadStrings("preset2.txt");
  presetText3 = loadStrings("preset3.txt");

  alphabetFull[0] = alphabet;
  alphabetFull[1] = alphabetCaps;
  alphabetFull[2] = symbols;
}


void draw() {
  background(255);
  menuSwitch();

  if (serialInput) {
    callSerial();
  }
  else numLockInput();

  loopCounterIncrm();
}


// always close Minim audio classes when you are done with them
void stop()
{
  aid.close();
  minim.stop();

  super.stop();
}
```

Input

```
// serial communication protocol
void callSerial() {
  String myString = arduinoPort.readStringUntil(' '); //the ascii value of the "|" character
  if(myString != null ){
    if(selectedButton!=int(trim(myString)) && int(trim(myString)) <8){
      pushButton=false;
      selectedButton=int(trim(myString));
      loopCounter=0;
      println(selectedButton);
    } else if (int(trim(myString)) ==8){
      pushButton=true;
      println("les go");
  }
}
}
```

```
// keyboard input -  whenever a key is pressed, if the the respective button is not null,
// it is selected and the loop counter is reset
void numLockInput() {
  if(keyPressed) {
    if (key == '1' && menuDefinitions[currentMenu][5]!=0) {
      selectedButton=5;
      loopCounter=0;
    }
    if (key == '2' && menuDefinitions[currentMenu][4]!=0) {
      selectedButton=4;
      loopCounter=0;
    }
    if (key == '3' && menuDefinitions[currentMenu][3]!=0) {
      selectedButton=3;
      loopCounter=0;
    }
    if (key == '4' && menuDefinitions[currentMenu][6]!=0) {
      selectedButton=6;
      loopCounter=0;
    }
    if (key == '6' && menuDefinitions[currentMenu][2]!=0) {
      selectedButton=2;
      loopCounter=0;
    }
    if (key == '7' && menuDefinitions[currentMenu][7]!=0) {
      selectedButton=7;
      loopCounter=0;
    }
    if (key == '8' && menuDefinitions[currentMenu][0]!=0) {
      selectedButton=0;
      loopCounter=0;
    }
    if (key == '9' && menuDefinitions[currentMenu][1]!=0) {
      selectedButton=1;
      loopCounter=0;
    }

    if (key == 'p') {
      saveFrame("testPrints-####.png");
    }
  }
}
```

Large GUI

```
void large_GUI() {
  // menu title
  fill(0);
  textSize(80);
  text(menuTitle[currentMenu], width/2, height/2);

  // if a button is selected draw it in bold
  if(selectedButton<8) {
    strokeWeight(14);
    ellipse(width/2+cos(3*PI/2+(selectedButton)*PI/4)*buttonScattering,height/2+sin(3*PI/2+
(selectedButton)*PI/4)*buttonScattering,buttonRadius,buttonRadius);
  }

  // draw the circle buttons and the text inside
  int n;
  for (n=0; n<8; n++) {
    if( menuDefinitions[currentMenu][n]!=0 ) {
      fill(255);
      strokeWeight(4);
      ellipse(width/2+cos(3*PI/2+n*PI/4)*buttonScattering,height/2+sin(3*PI/2+n*PI/4)*buttonScattering,
buttonRadius,buttonRadius);
      fill(0);
      textSize(25);
      text(menuTitle[menuDefinitions[currentMenu][n]], width/2+cos(3*PI/2+n*PI/4)*buttonScattering, hei
ght/2+sin(3*PI/2+n*PI/4)*buttonScattering);
```

```
    }
  }

  //what happens when enter is pressed - REPLACE BY BLINK - JUNTAR COM FUNCAO DE BAIXO
  if(keyPressed) {
    if (key == '5') {
      if(selectedButton<8) {
        currentMenu=menuDefinitions[currentMenu][selectedButton];
        loopCounter=0;
        selectedButton=8;
      }
    }
  }

  if(pushButton && selectedButton!=8) {
    pushButton=false;
    currentMenu=menuDefinitions[currentMenu][selectedButton];
    loopCounter=0;
    selectedButton=8;
  }
}
```

Menus

```
void menuSwitch() {
  switch (currentMenu) {
    //if emergency menu
  case 2:
    if(aidStart) {
      aid.loop();
      aidStart=false;
    }
    aidTitleFlash();
    break;

    //preset menu
  case 5:
    small_GUI();
    presets_GUI(presetText1);
    break;
  case 6:
    small_GUI();
    presets_GUI(presetText2);
    break;
  case 7:
    small_GUI();
    presets_GUI(presetText3);
    break;

    // writer menu
  case 8:
    writer_GUI();
    break;

    //else is Large GUI
  default:
    stopAidSound();
    large_GUI();
    break;
  }
}


//loop counter that runs in the end of every cycle to determine how long a button stays toggled
void loopCounterIncrm() {
  loopCounter+=1;
  if (loopCounter>holdTime) {
    loopCounter=0;
    selectedButton=8;
  }
}
```

```
void aidTitleFlash(){
 aidFlashCounter = (flashCounterUp) ? aidFlashCounter+2 : aidFlashCounter-2;
 if(aidFlashCounter>=254) flashCounterUp=false;
 if(aidFlashCounter<=0) flashCounterUp=true;
  background(255-aidFlashCounter);
 large_GUI();
 textSize(80);
 fill(255,0,0,aidFlashCounter);
 text(menuTitle[currentMenu], width/2, height/2);

}

//stop the aid Sound when leaving aid request
void stopAidSound() {
 aid.pause();
 aid.rewind();
 aidStart=true;
 aidFlashCounter=0;
}
```

smallGUI_presets

```
void small_GUI() {
  // GUI border
  strokeWeight(1);
  fill(255);
  ellipse(0,height/3,600,350);

  // menu title
  //pushStyle();
  fill(0);
  textSize(30);
  text(menuTitle[currentMenu], width/8, height/3);
  //popStyle();

  // if a button is selected draw it in bold
  if(selectedButton!=8) {
    //pushStyle();
    strokeWeight(5);
    ellipse(width/8+cos(3*PI/2+(selectedButton)*PI/4)*buttonScattering/3,height/3+sin(3*PI/2+
(selectedButton)*PI/4)*buttonScattering/3,buttonRadius/3,buttonRadius/3);
    //popStyle();
  }

  // draw the circle buttons and the text inside
  for (int n=0; n<8; n++) {
    if( menuDefinitions[currentMenu][n]!=0 ) {
      fill(255);
      strokeWeight(1);
      ellipse(width/8+cos(3*PI/2+n*PI/4)*buttonScattering/3,height/3+sin(3*PI/2+n*PI/4)*buttonScatteri
ng/3,buttonRadius/3,buttonRadius/3);
      fill(0);
      //      textSize(25);
      //      text(menuTitle[menuDefinitions[currentMenu][n]],
width/2+cos(3*PI/2+n*PI/4)*buttonScattering, height/2+sin(3*PI/2+n*PI/4)*buttonScattering);

      pushMatrix();
      translate(width/8,height/3);
      textFont(symbolFont);
      rotate(PI*n/4);
      text("X", 0, -buttonScattering/3-10);
      popMatrix();
      textFont(myFont);
    }
  }


}

void presets_GUI(String[] presetText) {
  //what happens when enter is pressed - REPLACE BY BLINK
  if(keyPressed) {
    if (key == '5') {
      if(selectedButton==0) {
        if(currentPreset>1) {
          currentPreset -= 1;
        }
        loopCounter=0;
        selectedButton=8;
      }

      if(selectedButton==2) {
        showPreset=currentPreset;
        loopCounter=0;
        selectedButton=8;
      }

      if(selectedButton==4) {
        if(currentPreset<presetText1.length-1) {
          currentPreset += 1;
        }
        loopCounter=0;
        selectedButton=8;
      }

      if(selectedButton==6) {
        currentMenu=menuDefinitions[currentMenu][selectedButton];
        currentPreset=1;
        showPreset=0;
        loopCounter=0;
        selectedButton=8;
      }
    }
  }

  if(selectedButton==2) {
    strokeWeight(3);
    line(width/2.5, height/3+18, width-width/10, height/3+18);
  }

  // Writes the presents options
  pushStyle();
  textAlign(LEFT,CENTER);
  for(int n=-3; n<=3; n++) {
    if(currentPreset+n >= 0 && currentPreset+n <= presetText1.length-1) {
      textSize(25-2*abs(n));
      fill(0,255-sqrt(abs(n))*125);
      text(presetText[currentPreset+n], width/2.5-sq(abs(n))*10, height/3+n*50);
    }
  }
  strokeWeight(1);
  line(width/2.5, height/3+18, width-width/10, height/3+18);
  popStyle();
  textSize(100);
  rectMode(CENTER);
  text(presetText[showPreset],width/2,height*.75,width-width/20,height/3);
}
```

writerGUI

```
void writer_GUI() {
  // if a button is selected draw it in bold
  if(selectedButton!=8) {
    strokeWeight(5);
    ellipse(width/2+cos(3*PI/2+(selectedButton)*PI/4)*buttonScattering/2,height*.27+sin(3*PI/2+
(selectedButton)*PI/4)*buttonScattering/2,buttonRadius/2,buttonRadius/2);
  }

  // draw the circle buttons
  for (int n=0; n<8; n++) {
    fill(255);
    strokeWeight(2);
    ellipse(width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2,height*.27+sin(3*PI/2+n*PI/4)*buttonScatteri
ng/2,buttonRadius/2,buttonRadius/2);
    fill(0);
  }

  // draw the text inside the buttons
  for (int n=0; n<8; n++) {
    if(writerPanel==0) {
      textSize(18);
      if(n>=0 && n<=4)
        for(int i=0; i<7; i++) {
          text(alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+i]
+ "",width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2+cos(i*2*PI/8+10*PI/8)*24, height*.27+sin(3*PI/2
+n*PI/4)*buttonScattering/2+sin(i*2*PI/8+10*PI/8)*24);
        }

      //if(n==4) text(alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n] +
"" + alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+1] + "" +
alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+2] + "\n" +
alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+3]+ "" +
alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+4]+ "" +
alphabetFull[currentAlphabetType][alphabetFull[currentAlphabetType].length/5*n+5],
width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2+n*PI/4)*buttonScattering/2);
      if(n==5) text("text\ntools", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2
+n*PI/4)*buttonScattering/2);
      if(n==7) text("caps\nsymb", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2
+n*PI/4)*buttonScattering/2);
    }
    else if(writerPanel>0 && writerPanel<=5) {
      if(n!=7) {
        textSize(20);
        text(alphabetFull[currentAlphabetType][n+(writerPanel-1)*7], widt
h/2+cos(10*PI/8+n*PI/4)*buttonScattering/2,height*.27+sin(10*PI/8+n*PI/4)*buttonScattering/2);
      }
    }
    if(writerPanel==6){
      if(n==1) text("para\ngraph", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2
+n*PI/4)*buttonScattering/2);
      if(n==2) text("space", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2+n*PI
/4)*buttonScattering/2);
      if(n==3) text("cursor\nright", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/
2+n*PI/4)*buttonScattering/2);
      if(n==5) text("cursor\nleft", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2
+n*PI/4)*buttonScattering/2);
      if(n==7) text("back\nspace", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/
2+n*PI/4)*buttonScattering/2);
    }
    if(n==6) text("<", width/2+cos(3*PI/2+n*PI/4)*buttonScattering/2, height*.27+sin(3*PI/2+n*PI/4)*b
uttonScattering/2);
  }



  //what happens when enter is pressed - REPLACE BY BLINK
  if(pushButton || (keyPressed && key == '5' && selectedButton!=8)) {

    //If the main panel is active
    if(writerPanel==0) {
      //If letter or text tool buttons are selected
      if(selectedButton>=0 && selectedButton<=5) {
        writerPanel=selectedButton+1;
        loopCounter=0;
        selectedButton=8;
      }
      if(selectedButton==6) {
        currentMenu=menuMessage;
        loopCounter=0;
        selectedButton=8;
      }
      if(selectedButton==7) {
        currentAlphabetType += 1;
        if(currentAlphabetType > 2) currentAlphabetType =0;
        loopCounter=0;
        selectedButton=8;
      }
    }
    //if the letter panels are activated
    else if(writerPanel>0 && writerPanel<=5) {
      //if the letter buttons are selected
      if(selectedButton>=0 && selectedButton<=5) {
        userText.insert(textCursor,alphabetFull[currentAlphabetType][selectedButton+1+(writerPanel-
1)*7]);
        textCursor +=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
      }
      if(selectedButton==6) {
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
      }
      if(selectedButton==7) {
        userText.insert(textCursor,alphabetFull[currentAlphabetType][(writerPanel-1)*7]);
        textCursor +=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
      }
    }

    //if the text tool panel is activated
    else if(writerPanel==6) {
      switch(selectedButton) {
      //paragraph
      case 1:
        userText.insert(textCursor,'\n');
        textCursor+=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        break;
      //space
      case 2:
        userText.insert(textCursor,' ');
        textCursor+=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        break;
        //cursorRight
      case 3:
        if(textCursor+1<userText.length()){
        textCursor+=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        }
        break;
        //cursorLeft
      case 5:
        if(textCursor-1>=0){
        textCursor-=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        }
        break;
      //back to menu
      case 6:
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        break;
      case 7:
        if(textCursor-1>=0){
        userText.deleteCharAt(textCursor-1);
        textCursor-=1;
        writerPanel=0;
        loopCounter=0;
        selectedButton=8;
        }
        break;

      }
    }
  }

  //text box
  strokeWeight(2);
  rectMode(CORNER);
  fill(245);
  //  rect(width/2,height*.75,width*.8,height*.4);
  rect(width*.05,height*.55,width*.9,height*.40);
  fill(0);
  textSize(20);
  //cursorTimer+=1;
  //if(cursorTimer>100) cursorTimer=0;
  //if(cursorTimer>50) userText.insert(textCursor,'|');
  //else userText.insert(textCursor," ");
  userText.insert(textCursor,'|');
  textAlign(LEFT,TOP);
  text(userText.toString(),width*.06,height*.56,width*.88,height*.38);
  textAlign(CENTER,CENTER);
  userText.deleteCharAt(textCursor);
}
```

Appendix 3 – EOG Oscilloscope

```
/*
 *    Oscilloscope for EOG
 *
 *
 *      by João Bárcia
 *
 *   built upon an oscilloscope proposal by
 * (c) 2008 Sofian Audry (info@sofianaudry.com)
 *
 *
 *
 */
import processing.serial.*;

Serial port;  // Create object from Serial class
float val, val2;      // Data received from the serial port
float[] values, values2, values3;
float threshold = 512;
float distance;
float spread=1;

void setup()
{
  size(640, 480);
  // Open the port that the board is connected to and use the same speed (9600 bps)
  println(Serial.list());
  port = new Serial(this, Serial.list()[1], 115200);
  values = new float[width];
  values2 = new float[width];
  values3 = new float[width];
  smooth();

  PFont font;
  font = loadFont("Gautami-30.vlw");
  textFont(font);
}

void serialEvent(Serial myPort){
  String myString = myPort.readStringUntil('|'); //the ascii value of the "|" character
  if(myString != null ){
    myString = trim(myString); //remove whitespace around our values
    int inputs[] = int(split(myString, ','));
    //now assign your values in processing
    if(inputs.length == 5){
      val = inputs[1];
      val2 = inputs[2];
      distance = inputs[3];
      println(inputs[3]);
    }
  }
}


float getY(float val) {
  return (float)(val / 1023.0f * height) - 1;
}

void draw(){
  strokeWeight(1);
  scale(spread,1);
  for (int i=0; i<width-1; i++){
    values = values[i+1];
    values2 = values2[i+1];
    values3 = values3[i+1];
  }
//  values[width-1] = val;
  values[width-1] = val;
  values2[width-1] = val2;
  values3[width-1] = distance;
  background(255);

  stroke(255);
  //line(0,480-threshold/1023*height,width,480-threshold/1023*height);

  for (int x=1; x<width; x++) {
    stroke(180,0,0);
    line(width-x,   height-1-getY(values[x-1]),
    width-1-x, height-1-getY(values
        •   ));
   stroke(0,180,0);
    line(width-x,   height-1-getY(values2[x-1]),
    width-1-x, height-1-getY(values2
        •   ));
   stroke(0,200);
//    line(width-x,   height-1-getY(values3[x-1])*100,
//    width-1-x, height-1-getY(values3
        •   )*100);[/color]
              line(width-x,   height-(9-values3[x-1])*18,
              width-1-x, height-(9-values3
        •   )*18);
  }

  for (int i=0; i<10; i+=1) {
    stroke(0,40);
    line(0, height-18*i, width, height-18*i);
  }

  textSize(30);
  textAlign(LEFT);
  scale((1/spread),1);
  fill(180,0,0);
  text("Ver = " + val * 5/1024 + "V", width-250,70);
  fill(0,180,0);
  text("Hor = " + val2 * 5/1024 + "V", width-250,35);
  fill(255);
  //text("Tresh = " + threshold, width-200,height-115);
  fill(0);
  //text("Signal = " + distance, width-200,150);
  textSize(15);
  textAlign(LEFT,CENTER);
  text("BLINK", 10,height-18*1);
  text("UP-RIGHT", 10,height-18*2);
  text("RIGHT", 10,height-18*3);
  text("DOWN-RIGHT", 10,height-18*4);
  text("DOWN", 10,height-18*5);
  text("DOWN-LEFT", 10,height-18*6);
  text("LEFT", 10,height-18*7);
  text("UP-LEFT", 10,height-18*8);
  text("UP", 10,height-18*9);
}

void keyPressed() {
  if (key == '+') {
threshold+=1;
  }
  if (key == '-') {
threshold-=1;
  }
  if (key == 'p') {
saveFrame("finals/osciloTestPrints-####.png");;
  }
}
```