"Design of a BMI for tetraplegic patients"

Delhaye, Benjamin ; Schramme, Maxime

Abstract

Tetraplegia is a paralysis of the four limbs, which leaves attained people without their full autonomy and freedom. Indeed, tetraplegia not only affects the movement capabilities of the patients, but also their ability to communicate with others. Needless to say that tetraplegic patients' everyday life is not easy at all and that they constantly suffer. In this context, a very interesting and exciting prospect is the field of Brain-Machine Interfaces (BMIs). BMIs, also called Brain-Computer Interfaces (BCIs) or neuroprosthetics, can be defined as systems that use brain signals as inputs to produce a control signal as output. This output can be used in various kinds of applications: to control a wheelchair, to move a mouse cursor, to control a voice synthesizer and communicate with other people. With a BMI, tetraplegic patients could move or communicate in an autonomous way, just by thinking about the action they would like to perform. Obviously, BMIs have the potential to change th...

Document type : *Mémoire (Thesis)*

UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Design of a BMI for tetraplegic patients

Dissertation presented by
Benjamin DELHAYE , Maxime SCHRAMME

for obtaining the Master's degree respectively in
Electro-mechanical and Electrical Engineering

Supervisors
Pascal DOGUET , Jean-Didier LEGAT

Reader
François-Xavier STANDAERT

Academic year 2015-2016

Acknowledgements

First and foremost, we would like to thank our master thesis directors Jean-Didier Legat and Pascal Doguet, for their availability, support, help and good advice throughout the duration of this work. The subject they proposed was very exciting and we enjoyed discovering it and working on it during this academic year. The world of Brain-Machine Interfaces has captivated us. We thank them for that too.

This work would not have been possible without the support of Synergia Medical. All needed resources, including the electronic components, tools, machines and material, were provided by Synergia Medical. We would like to thank the entire team of Synergia Medical for their welcome. Special thanks go to Gregory and Carmen for their advice during the PCB assembly. They also have printed in 3D the helmet, used for this work to position the EEG electrodes. Many thanks also to Pascal Doguet again, this time for his help with the assembly of several critical components.

On a personal level, we would like to thank our parents for the numerous rereadings of this text. This is not always easy to dive into a scientific report, in an unknown research field, and correct the mistakes. Our parents did it! Many thanks also to our friends and our beloved for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is estimated that more than 50000 people suffer from spinal cord injuries nowadays in France, with 1200 new trauma every year. Per million inhabitants, this corresponds to 20 new trauma every year. In Europe, from 10 to 30 new medullary injuries per million inhabitants are detected annually. This number goes even higher for the United States, where 11000 people are affected each year (40 new cases per million inhabitants). Several factors, which differ from one place to the other, can explain these dissimilar numbers [51] [47].

Spinal cord injuries are mainly caused by physical trauma, such as road accidents (50%), falls (22%), suicide attempts (10%) or firearm injuries (1%). Young population is more affected, especially men. Approximatively one third of the medullary lesions causes tetraplegia, while the remaining part is at the origin of paraplegia [51]. Other typical causes of spinal cord injuries are diseases, such as degenerative (e.g. Amyotrophic Lateral Sclerosis), inflammatory (e.g. Multiple Sclerosis), vascular or tumoral ones. Malformation of the spinal cord can also cause medullary paralysis (e.g. Spina Bifida) [16].

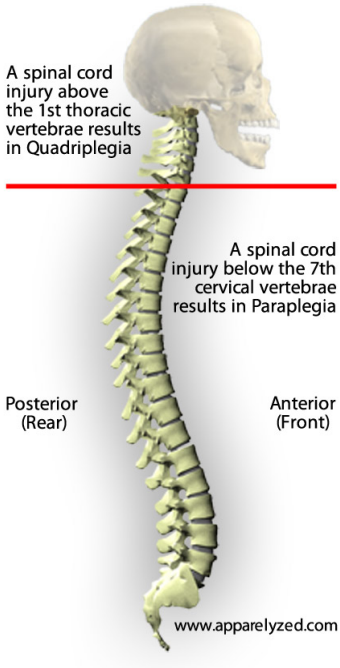


Figure 1.1: Spinal cord injuries causing tetraplegia and paraplegia [4] [5]

Tetraplegia (less often called quadriplegia) results from lesions to cervical spinal cord segments at levels C1-C8 and is a paralysis of the four limbs. Paraplegia (less often called hemiplegia) is caused by injuries to the spinal cord at thoracic, lumbar or sacral levels and affects only the legs. Figure 1.1 illustrates this. Lesions to the spinal cord typically result in motor or sensory impairment to the limbs, as the commands from the brain do not reach the targeted muscles. Depending on which spinal cord segment is injured, the patient can also suffer from additional complications. The lesions can lead to loss or impairment of autonomic functions, such as sexual functions, digestion or breathing for instance [5] [4].

The drama with spinal cord injuries is that there is no real cure: the spinal cord cells are destroyed and the human body is not able to replace them. When the affected spinal cord cells die, their function dies with them [15]. In addition to the fact that their life expectancy is highly reduced and that their pathology impacts their social situation [51] [47], tetraplegic people suffer from pain and from their (partial or total) inability to move or communicate. The simplest action becomes a nightmare, as the patient can not be autonomous any more. Numerous testimonies can be found on the internet, such as [33], [14] or [45]. More and more movies give an idea about the everyday life of this kind of disabled people. Recent movies like *Intouchables* (2011) or *The theory of everything* (2014) are good examples.

New technologies carry a lot of hope for tetraplegic people, because they have the potential to help them move, communicate and retrieve some kind of autonomy. A good example is the brain-body actuated system used by famous physicist Stephen Hawking, who was diagnosed with Amyotrophic Lateral Sclerosis when he was 21 years old. With his system, Stephen Hawking is able to control a computer, has access to the internet and interacts with others. By moving his cheek, he selects letters from a screen and forms words and sentences. A speech synthesiser is then used to formulate the written sentences [24]. Similar systems exist and allow disabled patients to control their wheelchair or to grab things by commanding robots. New technologies can also play a role in the rehabilitation of the patients, with the help of rehabilitation robots or virtual reality for instance [46]. In this context, one very exciting prospect is the field of brain-machine interfaces (BMIs) [31].

1.1 What is a BMI?

A *brain-machine interface* (BMI), also called *brain-computer interface* (BCI) or *neuroprosthetics*, is a system that links individuals' electrophysiological signals with a machine or a computer [8]. Brain signals are decoded and are used to control external devices, enable movements or communications with others, or facilitate rehabilitation of the patients [23]. The design of a BMI requires knowledge in several research fields, such as neurology, electronics, computer science, machine learning, signal processing or even robotics.

A healthy person does not need a BMI, as his brain is able to communicate with his body and transmit commands to perform an action. If the spinal cord is injured, the link between the brain and the body is broken, and a BMI can be useful to bypass the lesion. This situation is shown in Figure 1.2. A BMI is composed of several blocks [31]:

1. sensors are needed to acquire brain signals, in which are hidden information about patient's intention;
2. acquired signals must be processed (e.g. filtering, artefacts removal, feature extraction, classification) in order to decode them, retrieve patient's intentions and transform them into a control signal;

3. then, the control signal has to be transmitted to an effector, via either a wired link or a wireless one;

4. based on the command it receives, the effector performs the desired action (e.g. movement, locomotion, communication) for the patient (see arrow B in Figure 1.2).

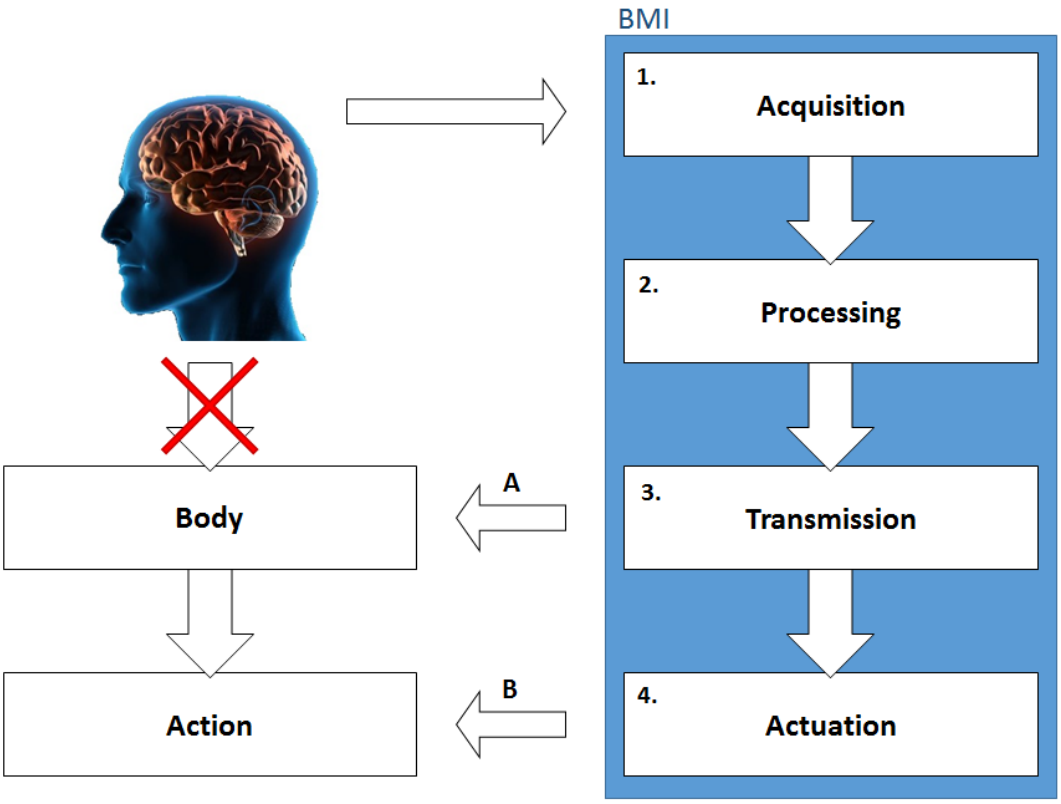


Figure 1.2: Conceptual description of a BMI [31]

As a remark, if patient's body and muscles are undamaged, the command can directly be used to stimulate the limbs instead of an external effector (see arrow A in Figure 1.2). This technique is called *Functional Electrical Stimulation* (FES) [13]. Also, even if it is not visible in the presented scenario, a feedback loop exists in the system, as the patient usually sees what movement the effector does.

1.2 Objectives of this work

This Master's thesis is done in collaboration with *Synergia Medical*, which is a company based in Mont St-Guibert working in the field of neural implants and new generation medical devices. Synergia Medical develops active implantable medical devices and biosensors in order to treat neurological disorders [35]. They are interested by the field of BMIs and would like to explore it, in order to develop interesting solutions for tetraplegic patients.

Therefore, this work has the following objectives, which correspond to the different steps that have been followed:

1. learn what has been done previously in the field, evaluate the possibilities (e.g. electrodes to use, signal processing steps, classification algorithms), discover the trends, see what are the applications;
2. imagine and design a flexible and programmable experimentation system, dedicated to the development and testing of BMIs based on *Electroencephalogram* (EEG) signals; this objective concerns only the acquisition, processing and transmission blocks in Figure 1.2 (the actuation is not covered in this work) and is composed of the following steps:
   (a) choice of a smart solution to acquire the brain signals properly (rigid structure to position correctly the EEG electrodes);
   (b) choice of the electronic components, including a *Field-Programmable Gate Array* (FPGA), which is the central component of the development board;
   (c) design and assembly of a *Printed Circuit Board* (PCB);
3. demonstrate that the development board is working well and develop a small application based on Motor Imagery (when the patient thinks about performing a movement, without doing it necessarily); the goal is to classify EEG patterns produced by imagined hand movements in real-time, by lighting the leftmost/rightmost LED of the development board when "Left"/"Right" is thought by the user; this implementation phase goes through several successive steps:
   (a) choice of an acquisition procedure to record samples for imagined left/right hand movements;
   (b) offline learning of classification algorithms, using MATLAB;
   (c) real-time classification;
4. identify possible areas of improvements and future steps.

Each objective corresponds roughly to a chapter. Therefore, this document is structured as follows.

Chapter 2 gives the background about BMIs. Several aspects are presented, such as brain signals or the classification algorithms that can be used to control a BMI. Then, the design of an experimentation system, done in collaboration with Synergia Medical, is explained in Chapter 3. Specifications are given and every design choice is justified. The assembled and fully functional system is finally illustrated at the end of the chapter.

The application developed in this work is implemented on FPGA and classifies in real time the thoughts of the user. It proves that the system developed here works perfectly and can be used to develop BMIs. The implementation details are given in Chapter 4. Chapter 5 discusses the obtained results. Characterisation of some parts of the system and an evaluation of some key performances are given.

The last chapter summarises the entire work and discusses the numerous possibilities that exist to improve the system and the application.

Chapter 2

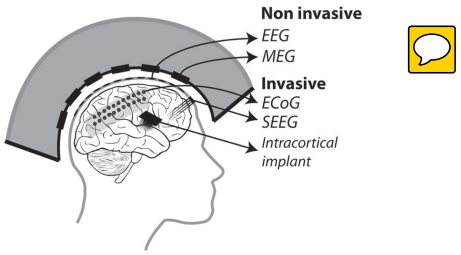
Background

This chapter gives background about BMIs and is composed of three parts. First, Section 2.1 describes the brain signals that can be used to control a neuroprosthetics and introduces electrodes positioning systems. Then, Section 2.2 gives a brief history of BMIs, explaining what has been done until now. Finally, Section 2.3 summarises the different classification algorithms that have been used in the field of BMIs.

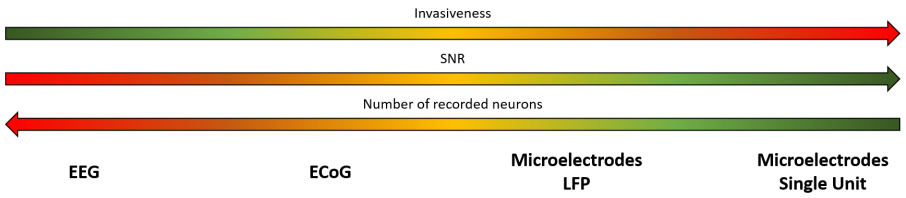
2.1 Neurophysiological signals

Voltage measures of the electrical activity of neurons are usually used to control BMIs and are called neural signals or brain signals [38]. They can be recorded at different distances from the neurons [31], as it is partly shown in Figure 2.1(a): either from the scalp, and in this case brain signals are called *Electroencephalogram* (EEG); either from the surface of the cortex, and in this case they are referred to as *Electrocorticogram* (ECoG); or directly near individual neurons in the brain using *microelectrodes*.

The deeper in the brain, the closer to the neurons and the higher the quality of the signal (or equivalently, the *Signal-to-Noise Ratio* (SNR)), at the cost of more invasiveness [31]. This is represented by Figure 2.1(b).



(a) Different recording methods [6]



(b) Characteristics of different recording methods

Figure 2.1: Illustration of the several recording methods that can be used to control BMIs

This section is structured as follows. EEG, ECoG and signals recorded with microelectrodes are described. The emphasis is put on the EEG part, because they are non invasive and they are used for this work. The positioning system for EEG is also described, together with the needed electrodes.

As a remark, the activity of neurons does not produce only an electrical response. Magnetic and metabolic responses (resulting in changes of blood flow) are also produced and could be used in a BMI [38]. Magnetic fields resulting from neural activity can be recorded using *Magnetoencephalography* (MEG). Brain metabolic activity can be measured with different techniques, such as *Positron Emission Tomography* (PET) or *Functional Magnetic Resonance Imaging* (fMRI).

2.1.1 EEG

EEG signals are brain potentials recorded from the scalp. They are thus non invasive, which is a great advantage of EEG. As EEG electrodes are far away from the neural activity source, they record the influence of thousands of neurons. As a consequence, the spatial resolution and the SNR are poor and the response is slow. EEG signals recorded from the scalp are in the microvolt range and the signal bandwidth is usually located below 35 Hz [31] [8]. Main characteristics are summarised in Table 2.1.

EEG	
Amplitude	10-100 μV
Bandwidth	0.1-35 Hz
Spatial resolution	3 cm

Table 2.1: EEG characteristics [31] [8]

EEG signals are very susceptible to ambient electrical noise, but also to artefacts [31] [38] [52]. Several physiological activities can cause artefacts: eye movements or blinking, muscular tension, heart beat, breathing, etc. Artefacts can also be caused by electrical interferences from power lines or electronic systems around. These artefacts impact the recorded signal and can lead to a wrong detection or to a wrong interpretation.

EEG signals have been studied for several decades. As a consequence, the equipment to record them is well-established and standardised. Initially, EEG signals were recorded with

(a) Pen recorder from 1958 for 8 EEG channels [39]

(b) Examples of 14 EEG waveforms [40]

Figure 2.2: Examples of EEG acquisition

EEG pen recorders, as the one shown in Figure 2.2(a). Nowadays, signals are digitised and can be displayed on a computer screen. Figure 2.2(b) shows examples of recorded EEG waveforms.

As the SNR of EEG signals is poor, it is important to have good electrodes in order to avoid degrading it even further. That is why electrodes are constructed from very conductive materials such as gold or silver chloride (AgCl). In addition, conductive gels or pastes are used to lower even further the impedance between the electrodes and the scalp [8]. However, these gels can be avoided nowadays by using dry electrodes [23] [38].

Two EEG positioning systems are widely used and are depicted on Figure 2.3 [8] [38] [30]. The oldest one contains 21 electrodes and is named *10-20 international system* (in orange on the picture). A newer system is composed of 76 electrodes, to offer a better spatial resolution (black circles on the picture) and is called *5-10 international system*.

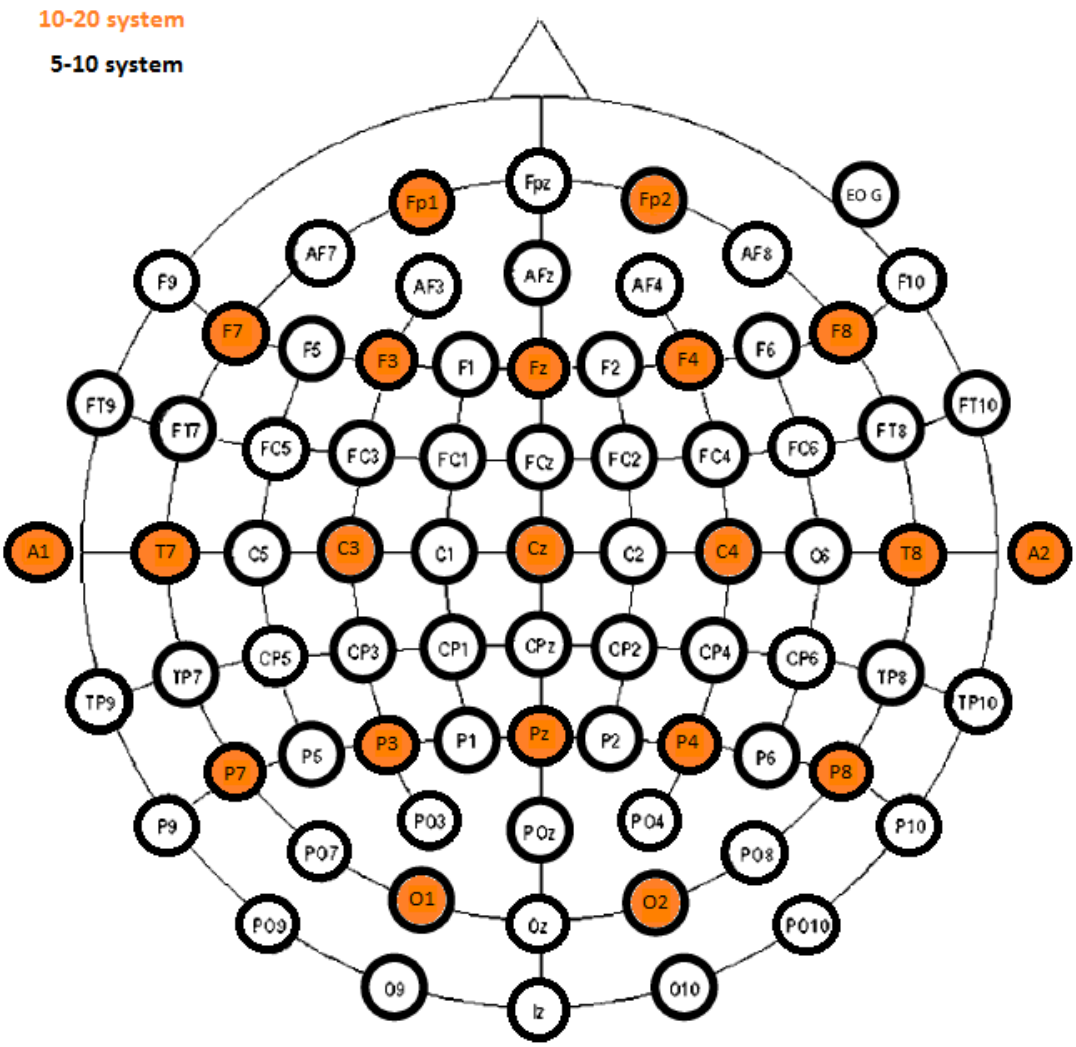


Figure 2.3: Representation of two standard EEG positioning systems; even/odd numbers correspond to electrodes located on the right/left side of the head; letters correspond to the general area of the cortex the electrode is located above; A stands for auricular, C for central, Fp for prefrontal, F for frontal, P for parietal, O for Occipital, T for temporal; EOG electrodes can be added to record vertical and horizontal electro-oculographic movements.

As the amplitude range of recorded EEG signals is low, these are usually amplified, sometimes by a factor 10000. There exist various ways to connect the electrodes to the amplifiers [8]: either with a *referential montage*; either with a *bipolar montage*.

With referential montages, a reference electrode is placed in an electrically quiet area and all other electrode potentials are computed with respect to the reference. In other words, all amplifiers have an input connected to the shared reference while other electrodes are individually connected to their remaining input. The reference is often put on an ear (electrode A1 or A2). As it is almost impossible to have a single reference electrode entirely inactive, several reference electrodes are used and the actual reference is computed as the average of the reference electrodes potentials. Usually, two electrodes put on the ears (electrodes A1 and A2) are used and this configuration is known as linked-ears reference [8] [38]. As a remark, several processing techniques and signal transformations can get rid of the reference. *Surface Laplacian* (SL) derivation, *Common Average Reference* (CAR) or *Common Spatial Patterns* (CSP) are good examples [38].

Bipolar montages connect pairs of electrodes to the inputs of amplifiers. The amplifiers are not all connected to a shared reference. The advantage of this montage is that it can distinguish local activity much more clearly. The drawback is that it may also cause distortion. This is probably the reason why they are less often used than referential configuration [8].

Depending on the position of the electrodes that are used, different "types of thoughts" are recorded. As an example, electrodes situated close to the motor areas of the cortex (e.g. C5, C3, C1, Cz, C2, C4, C6, CP3, CP1, CPz, CP2, CP4, FC3, FC1, FCz, FC2, FC4) will mainly record neural activity related to (real or imagined) movements [30]. EEG signals can thus be categorised as a function of the position of the source of neural activity. Different responses can also exist in different frequency bands. In fact, EEG signals can be categorised in many different manners. Figure 2.4 shows the usual way to classify EEG signals. The different categories are explained hereafter.

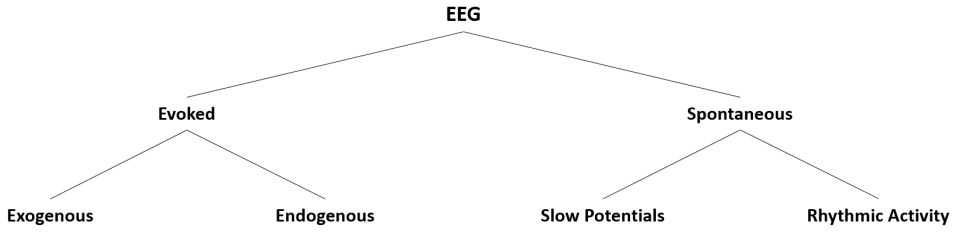


Figure 2.4: Categories of EEG signals

2.1.1.1 Evoked potentials

The first category of EEG signals is composed of *evoked potentials* (EPs) and appears in Figure 2.4. EPs are responses of the brain to the occurrence of some external sensory stimuli, such as flashed images and lights [38]. They can be used to control a BMI when the latter produces the appropriate stimuli. The benefit is that it requires little training. However, users have to wait for the corresponding stimulus [8]. One important factor that influences EPs response is the frequency of stimulus occurrence: the less frequent the stimulus, the larger the response.

There are two types of EPs' components [8]: either *exogenous* or *endogenous*. Components influenced primarily by physical stimulus properties are called exogenous. They generally in-

clude a negative waveform N1 around 100 ms and a positive waveform P2 around 200 ms after stimulus onset. *Visual Evoked Potentials* (VEPs) are exogenous components. They are related to electrophysiological mechanisms taking place when the brain processes visual information. *Steady-State Visual Evoked Potentials* (SSVEPs) are induced by stimuli repeated at a rate higher than 6 Hz [38].

Components influenced primarily by cognitive factors are called endogenous and take place after exogenous components [8]. The well-known P3 component is an EP that appears in response to an awaited infrequent event. It is composed of a positive wave around 300 ms after stimulus onset [38]. As it is related to cognitive factors, the P3 component is known to change in response to subject fatigue [8].

2.1.1.2 Spontaneous signals

The second kind of EEG is composed of *spontaneous signals*, which are no more linked to external stimuli but are carried out by the subject at his own will. This is also represented in Figure 2.4. As they are spontaneous, they can be linked to any kind of thoughts and are less easily used. As a consequence, they require more training in order to be used to control a BMI. When subjects are trained to learn to self-control their spontaneous brain signals, this is called *Operant Conditioning*. This learning can be achieved by giving a visual or auditory feedback of the signal that the subjects try to control [38].

Several categories of spontaneous EEG signals exist. A first category is composed of *Slow Potentials* which are nonoscillatory event-related signals. Two examples fall in this category [38]: *Slow Cortical Potentials* (SCPs), which are related to the overall preparatory excitation level of a given cortical network; and *Bereitschaftspotential* (BP), which is related to slow pre-movement potentials.

The second category of spontaneous brain signals is named *Rhythmic Activity* and designates oscillatory activities generated by complex networks of neurons with feedback loops. This oscillatory activity is generally analysed in the frequency domain [38]. For instance, the μ rhythm corresponds to a frequency band between 8 and 12 Hz. It is associated with motor activities and is usually recorded over the sensorimotor cortex [8]. Other rhythms exist, associated with other frequency bands and location over the cortex, such as the α rhythm (8-12 Hz over the occipital cortex) or the β rhythm (13-28 Hz). *Event-Related Synchronization/Event-Related Desynchronization* (ERS/ERD) are movement-related increases/decreases for specific rhythms and can also be used to control BMIs, for instance by imagination of hand movements [8] [38].

2.1.2 ECoG

Brain potentials recorded from the surface of the cortex (below the scalp) are called ECoG. Corresponding electrodes are invasive, but they remain outside of the brain and do not hurt it. In addition, compared to EEG, they benefit from better SNR and spatial resolution. ECoG signals are presented as a good compromise between high-resolution and highly invasive microelectrodes and low-resolution and non-invasive EEG electrodes [31].

2.1.3 Microelectrodes

Microelectrodes are put into the cortex to record smaller population of neurons. As a consequence, they are highly invasive and can cause tissue reaction and inflammation [31]. Figure 2.5 shows a microelectrode array.

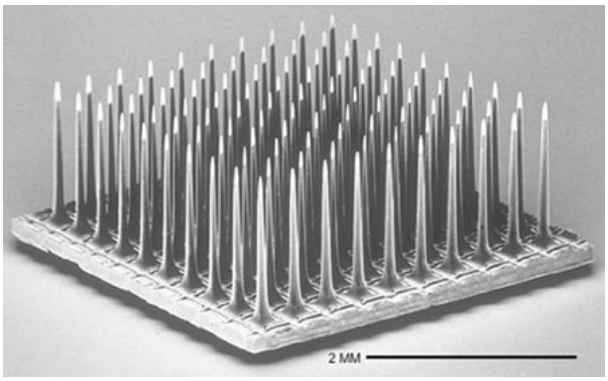


Figure 2.5: Microelectrode array [25]

Depending on the distance existing between the microelectrode and the neural population, two types of signals can be recorded [31]: either the microelectrode is very close to a *single neuron* (within 150 μm distance) and records only its activity; either the microelectrode is further away (within 250 μm distance) and records *Local Field Potentials* (LFPs), resulting from the summation of electrical activity of small populations of neurons.

Microelectrodes that record the activity of single neurons benefit from a higher SNR as they are closer to the neural activity source. However, they are extremely susceptible to motion artefacts. The main challenge is to put the electrode near the neuron, but not too close in order to avoid hurting or killing the neuron cell, neither too far away in order to keep the signal quality high. In addition, neural tissues react to the presence of the electrode and can degrade the signal over time.

LFP microelectrodes have the advantage to be more robust against movement artefacts and more resistant to tissue reaction. As they are further away, signal quality is of course lower.

2.2 History of BMIs

In this section, the main events in BMIs history are summarised and examples of applications are given in order to identify what has been done in the past. This enables also to sense what is feasible and what are the trends in the field. Events are sorted out chronologically.

2.2.1 From EEG invention to the first descriptions of BMIs

The history of BMIs begins with the first record of a human Electroencephalogram (EEG) by Hans Berger around 1924 [8]. Berger successfully recorded human brain activity from the scalp, opening the way to new research areas and enabling devices to be controlled by mind.

Around 1964, the P3 component was discovered independently by two research groups, conducted by Chapman and Bragdon on one hand, and by Sutton on the other hand [8].

At the same time, the first description of a BMI was made by Grey Walter. Walter surgically placed electrodes inside patient's brain. Therefore, a direct link between motor regions of the patient's brain and a slide projector was created, enabling the patient to control the position of the slide projector by thoughts. Walter never published his results [22].

In 1973, Jacques Vidal published the first article about non-invasive BMI, which was based

on EEG signals. Vidal mainly described the EEG signal acquisition and processing steps. He also mentioned which brain signals could be used to control BMIs and their properties. Many of the concepts described by Vidal in his papers are still used today in the field of BMIs [8] [23].

2.2.2 The first BMIs

From that time and up to the 2000s, there was no major progress in the field. Only a few groups of researchers were working on BMIs and the number of different applications was limited [23]. Here are listed a few examples of research applications, which are described by Bayliss in [8] and summarised in Table 2.2.

1. Around 1988, a group led by Farwell and Donchin showed that the evoked P3 component could successfully be used to control a neuroprosthetics. Farwell and Donchin demonstrated that with a *P3 Character Recognition System*. Their BMI is composed of a screen that displays a 6-by-6 grid containing letters. The principle is simple. Every 100 ms, a column or a row of letters flashes. Users can select a letter by counting the number of times their letter flashed during a trial. Each time the desired letter flashes leads to a P3 response, which is detected by the system. This character recognition BMI is very accurate and requires a very short training time. However, the chosen flashing rate of 10 Hz may cause epilepsy. In addition, the system speed was quite low.

2. Another group conducted by Wolpaw designed in 1991 a μ *Rhythm Cursor Control*. Wolpaw trained subjects to modify their μ rhythm amplitude in order to move a cursor up or down on a screen. Compared to the work of Farwell and Donchin, the speed of Wolpaw's BMI is faster as users do not have to wait external stimuli (μ rhythm is spontaneous). However, the training time is longer and, more importantly, not all subjects were able to control the system.

3. A BMI described by Sutter's group in 1992 was called *Brain Response Interface*. It shares similarities with the P3 Character Recognition System described earlier. The system uses VEPs acquired with implanted electrodes and is composed of a screen that displays a grid of 64 symbols. Periodically, a subgroup of symbols undergoes a color alteration. Users concentrate on a symbol, and their EEG response is stored. The desired symbol can then be found by the BMI and the desired symbol is picked. An important fact is that this BMI was tested with severely disabled people, which is not often the case. The main advantage is its speed, even if it is far lower than the speed of a good typist.

4. In 1996, Pfurtscheller and his colleagues tested an *ERS/ERD Cursor Control*. A cursor displayed on a screen can be moved either to the right or left direction using ERS/ERD signals, extracted from EEG electrodes placed over the sensorimotor cortex. Trials with eye or muscle artefacts are automatically rejected, in order to improve classification accuracy. However, this could lower the speed. The required training time is short again, but not all subjects were able to learn how to control the system again.

5. Middendorf's group trained subjects to use a small *SSVEP Flight Simulator* in 1999. Subjects control the amplitude of SSVEP to flashing florescent tubes to command an aeroplane displayed on a screen. When their VEP amplitude goes below/above a threshold for a specific period of time, a plane rolls to the left/right. In addition to this experiment, Middendorf integrated also Functional Electrical Stimulation (FES) to his BMI in order to provide direct control for healthy muscles. When VEP amplitude goes above the threshold for one second, FES turns on and muscle activation begins (current increases

and muscle contraction activates). When VEP amplitude goes below the threshold for one second, FES turns off and stimulation stops (current decreases and muscle contraction deactivates). As with the P3 Character Recognition System described earlier, the chosen flashing rate may cause epilepsy in some subjects.

6. Around the same period of time, Kennedy and colleagues designed an *Implanted BCI*, which has been implanted into two patients. The application is again related to cursor control. Based on the neural firing rate, the vertical and horizontal position of a computer mouse cursor is computed. In addition to that, a mouse click or enter signal is also extracted from brain waves. Locked-in patients were trained to control the cursor and were able to communicate by sending emails and control a computer for instance. Kennedy used neurotrophic electrodes, which contain growth factors in order to encourage brain tissues to grow on them. Thanks to these special electrodes, at least two subjects have been able to use the system for more than 426 days.

7. Also in 1999, another application called the *Thought Translation Device*, designed by Birbaumer's group, was used by Amyotrophic Lateral Sclerosis patients. The system enabled subjects without any form of motor control to communicate with the outside world. The experiments revealed that not all subjects were able to use the system. Also, motivation is a key feature to learn how to control the BMI.

In her PhD dissertation published in 2006, Bayliss analyses the seven systems described above [8]. Table 2.2 gives a comparison of these examples. Several observations can be made and summarise very well the BMIs characteristics of the 20th century:

1. the above examples show that various kinds of brain signals can be used to successfully control a BMI; the signal choice has an influence on the learning rate, the accuracy and the retention; as a consequence, there are also various kinds of algorithms that can be used, for instance to pre-process acquired data (e.g. artefacts removal) or to decode brain signals;

2. BMIs' applications are mainly related to control and to communication tasks (in particular spelling), motivated by the wish to help disabled people retrieve their autonomy; only a few exceptions are taken out of the labs and can be used in reality by disabled people; furthermore, most BMIs are not very flexible and have a specific hardware architecture, which make changes or updates not easy to do without adapting the entire system;

3. human factors are not taken into account in the research, but have a big influence on subject's performances; for instance, it appears that experiments done with disabled people usually require a longer learning time, due to fatigue, stress or medication for instance; moreover, user satisfaction plays a role and has also an impact on the learning rate and the accuracy; finally, not all individuals are able to learn how to control a BMI, either because of physiological differences (every brain is structured differently), of brain damages or because of a loss of motivation and satisfaction.

2.2.3 BMIs of the 21st century

The 21st century has seen the emergence of more than 300 groups of researchers working in the BMI field. Many factors can explain this tremendous growth [8] [23]: scientists understand better how a normal or abnormal brain works; human factors are better taken into account by the researchers; electronic chips and instrumentation are cheaper, smaller and faster; computation power has increased and algorithms have advanced, enabling to work in real-time; sensors

Systems	**P3 Character Recognition**	**μ Rhythm Cursor Control**	**Brain Response Interface**	**ERS/ERD Cursor Control**	**SSVEP Flight Simulator**	**Implanted BCI**	**Thought Translation Device**
Application type	Communication	Control	Communication and control	Control	Control	Communication and control	Communication
Signal type	P3	μ Rhythm	VEPs	ERS/ERD	SSVEPs	Neural firing rate	Slow cortical potential
Invasive?	No	No	Yes	No	/	Yes	No
Training time	Minutes	15-20 sessions	10-60 minutes	2-2.5 hours	6 hours	Months	Months
Volunteers	Healthy	Healthy	Disabled	Healthy	/	Disabled	Disabled
Number of choices	36	2	64	2	/	/	27
Average speed (items/min)	4	20	30	/	/	2	2
Accuracy	95%	90%	90%	>89%	>80%	/	70-90%
Retention	Excellent	/	Excellent	/	/	Excellent	Not good
Subjective satisfaction	/	/	Considered	/	/	Considered	Discussed
Additional comments	May cause epilepsy	/	/	Artefacts removal	Also used with FES; may cause epilepsy	/	Artefacts removal

Table 2.2: Comparison of several BMIs (data coming from [8])

are improved, for instance with active and dry non-invasive electrodes and better invasive ones; etc.

As a consequence of this growth, performances and usability of BMIs have advanced a lot and motivations and trends have evolved [23]. More and more real-time BMIs are imagined and designed, showing that these systems go step by step out of the labs. The majority of current neuroprosthetics still use EEG, because of its simplicity, including P3 evoked potential, SSVEP or even Auditory Steady-State Response (ASSR). Most common applications remain related to control and spelling, but new applications appear such as monitoring, speech, coma, authentication, mechanical ventilation, learning, sensation, etc. BMIs could also be used by soldiers in order to augment their capabilities and speed up their response (e.g. hands-off control for fighter pilots) [8]. Space applications could require BMIs in the future, to control robotic devices in hostile and dangerous environments for instance [38]. Gaming industry and automotive are also sectors where BMI's technologies could play a role. A last example is related to radiotelegraphy, where two people can communicate only by thoughts thanks to BMIs [54].

To highlight these major directions in the BMI field and to promote excellence in this domain, an annual *BCI award* has been created. Ten nominees are chosen among all the candidatures and their work is summarised each year in a book [23]. This enables to track year after year the constantly evolving trends followed by BMI researchers.

As a remark, with the development of the internet, more and more information about BMIs can also be found on open source websites dedicated to them. As an example, *OpenBCI* [42] describes with a lot of details how to build a BMI, including a helmet to position EEG electrodes and a PCB with electronic components. Another open source website, *OpenVibe* [43], gives access to software and databases that can be used with BMIs.

2.2.4 What makes a good BMI today?

Nowadays, what are the essential characteristics that make a good BMI for tetraplegic patients? The answer to this question is not unique, complete nor definitive. However, important factors can be pointed out. Today, the *ideal BMI* should not only operate in real-time, be accurate and be learned easily and quickly. It should also be cheap, in order for more patients to be able to buy it and use it. A lot of tetraplegic patients just cannot afford the price of current systems.

In addition, flexibility, portability, reliability, usability and usefulness of a BMI are characteristics of primordial importance [8]. Flexibility is essential as it enables maintenance and augmentation of the BMI, which are absolutely needed in a constantly changing research field. It also includes the fact that the BMI has to adapt itself to its user. This can be done with Machine Learning algorithms for instance. Portability and reliability imply that BMIs can go out of the lab and be used anywhere, without being too much visible and annoying for the user. Usability is often linked to speed and signal recognition, but is also impacted by human factors, such as satisfaction and motivation. Finally, a good BMI should be useful for a wide variety of tasks, as it is intended to be used almost constantly by disabled people.

In particular for invasive BMIs, miniaturisation, power dissipation and safety are unavoidable features [36]. It is needless to say that implanted devices which dissipate too much power in a human brain can disturb or even destroy brain cells and be dangerous for the patient. Long-term biocompatibility (linked to signal longevity) is also required, to avoid inflammation, injuries and complications due to the contact between the implanted device and the brain tissues [10] [9] [31].

Obviously, the list of important factors to take into account when designing a BMI is very long and not yet complete. It includes both technical (e.g. speed, accuracy, signal, cost, brain signal, algorithms, adaptability to the user, etc.) and human features (e.g. retention, health status, learning time, satisfaction, motivation, adaptability to the BMI, etc.).

2.3 Classification algorithms

BMIs are controlled by brain signals, which need to be decoded. In most cases, this decoding step relies on classification algorithms. A good review of classification algorithms used with BMIs is given by Lotte's group in [32] and this section is highly inspired by it.

This part is structured as follows. First, the definition and properties of a classifier are given. Secondly, features extracted from EEG signals are briefly discussed. Then, two classification algorithms used with BMIs are described. Finally, a comparison of several classification algorithms is given in order to make a choice for the work presented in this document.

2.3.1 What is a classifier?

A binary classifier $\mathcal{C}$ can be defined as being a function $f : \mathbf{x} \rightarrow y$ that maps an input feature vector $\mathbf{x} \in \mathbb{R}^N$ to a class label $y \in \{-1, +1\}$. The goal of classification is to find the true label y^* corresponding to the input feature vector $\mathbf{x}$ [32]. Two steps are required in order to use a classifier:

1. *Learning phase*: based on a known set of input-output pairs $\mathcal{S}_L = \{(\mathbf{x}_i, y_i^*) \mid i = 1, 2, ..., N_L\}$, function f can be *learned* to fit these data as much as possible according to a chosen criterion; $\mathcal{S}_L$ is called the *learning* or *training* set;
2. *Classification*: then, the classifier can be used to classify a new feature vector $\mathbf{x}_{new}$ and to give it a label y_{new} (which is ideally equal to the unknown true label y_{new}^*).

As a remark, the goal is not to fit perfectly the learning set (in which case, this is called *overfitting*). The real objective is to get a model that performs well on new samples. In other words, the classifier has to have good *generalisation* properties. A second set of known input-output pairs $\mathcal{S}_V = \left\{(\mathbf{x}_j, y_j^*) \mid j = 1, 2, ..., N_V\right\}$, called *validation* set, is usually used to evaluate the properties of a classifier when dealing with new data. This is done by feeding the classifier with feature vectors $\mathbf{x}_j$ from the validation set, and by comparing the corresponding classifier outputs $y_j = f(\mathbf{x}_j)$ with the true labels y_j^*.

There are a lot of classification algorithms, and several of them have been used in the BMI field. Figure 2.6 shows the most common classification algorithms used with EEG-based BMIs. They can be compared thanks to various properties:

1. *generative/discriminative*: generative classifiers learn the class model and classify a feature vector by choosing the class in which it most likely belongs; discriminative classifiers learn the way of discriminating and classify the input feature vector directly;
2. *static/dynamic*: static classifiers do not take into account temporal information and classify a single feature vector at a time; dynamic classifiers classify a stream of feature vectors and are able to catch temporal variations;
3. *stable/unstable*: the performances of stable classifiers are not affected when there are small variations in the learning set, while it is the case with unstable ones; this is also linked to the classifier complexity;

4. *regularised*: regularisation limits the complexity of a classifier to prevent overfitting and gives good generalisation performances and robustness against outliers.

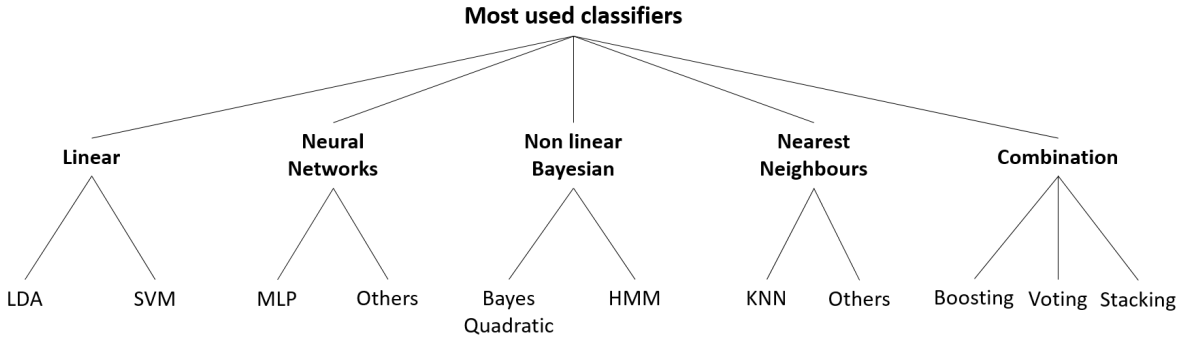


Figure 2.6: Most common classification algorithms used with BMIs [32]

One very important problem with classification algorithms is named the *curse of dimensionality* [32]. In order to describe properly the different classes, the size of the training set has to increase exponentially with the dimension of the feature vectors. However, this is very difficult to achieve in practice, hence the name "curse".

2.3.2 EEG-based features

The choice or the generation of relevant features is critical in order to obtain good classification performances. Several types of features have been used in BMI research [32]: *amplitude values* of EEG signals, *Band Powers* (BPs) [48], *Power Spectral Density* (PSD) values [7] [30], *Autoregressive* (AR) coefficients [48] [21] [19] [30], *Adaptive Autoregressive* (AAR) coefficients, *time-frequency* features, etc.

Features extracted from EEG signals have some critical properties it is useful to consider [32]. First of all, they are *non-stationary* as they can vary rapidly over time and over sessions. In addition, because of their poor SNR (see Section 2.1), EEG signals are *noisy*. This can introduce *outliers* in the features. Also, learning sets are often *small*, since data acquisition takes time. Finally, feature vectors can also contain a lot of features and be of *high dimensionality*.

As a remark, *feature selection* algorithms exist in order to find the most relevant features for a given application. This kind of algorithms will not be discussed in this document, but can also be useful with BMIs.

2.3.3 Classification algorithms in the BMI field

As shown in Figure 2.6, several types of classification algorithms have been used with BMIs [32]: *Linear* classifiers use linear functions to distinguish classes; *Neural Networks* are composed of several artificial neurons to produce nonlinear decision boundaries; *Bayesian* classifiers are generative and produce also nonlinear decision boundaries; *Nearest Neighbours* classifiers assign a feature vector to a class according to its nearest neighbours; finally, *Combination* of several simple classifiers can be used and results in a more powerful classification.

In this part, only two algorithms are described: *Linear Discriminant Analysis* (LDA) and *Support Vector Machine* (SVM). Multi-class SVMs and combination of algorithms are also briefly discussed.

Other classification algorithms, such as *Multilayer Perceptron* (MLP), *Bayes Quadratic*, *Hidden Markov Model* (HMM) or *K-Nearest Neighbours* (KNN), are not detailed here but are reviewed in [32].

2.3.3.1 LDA

LDA is probably the simplest classification algorithm. It separates two classes using a hyperplane. Therefore, the mapping function can be defined as

$$y = f(\mathbf{x}) = \text{sign}\left(\mathbf{x}^T\mathbf{w} + b\right) = \text{sign}\left(\sum_{j=1}^{N} x_j w_j + b\right), \tag{2.1}$$

where $\mathbf{w}$ is the weight vector and b is an offset [1]. These parameters are determined during the learning phase, by seeking the hyperplane that maximises the distance between the classes means and minimises the interclass variance [32]. Figure 2.7 illustrates linear classification of two classes using only two features.

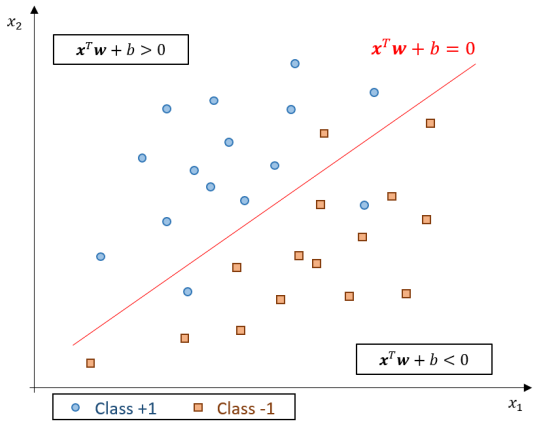


Figure 2.7: Linear classification of two classes using two features

2.3.3.2 SVM

SVM also uses a hyperplane to separate two classes, but the selected hyperplane is the one that maximises the margins [32]. This is illustrated with Figure 2.8. Therefore, the mapping function stays the same as the one defined in (2.1) for the LDA algorithm and is recalled here:

$$y = f(\mathbf{x}) = \text{sign}\left(\mathbf{x}^T\mathbf{w} + b\right) = \text{sign}\left(\sum_{j=1}^{N} x_j w_j + b\right), \tag{2.2}$$

where $\mathbf{w}$ is the weight vector and b is an offset.

The margin is inversely proportional to $\|\mathbf{w}\|$. Maximising the margin is thus equivalent to minimising $\|\mathbf{w}\|$ [1]. There exist different formulations in order to determine the optimal hyperplane. These are presented here from the simplest to the most complex:

1. *Hard-margin SVM formulation*: the hard-margin formulation assumes that the data are linearly separable. Using a training set $\mathcal{S}_L$, the goal is to find $\mathbf{w}$ and b that solves

$$\min_{\mathbf{w},b} \ \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i^* \left(\mathbf{x}_i^T\mathbf{w} + b\right) \geq 1, \quad \forall(\mathbf{x}_i, y_i^*) \in \mathcal{S}_L; \tag{2.3}$$

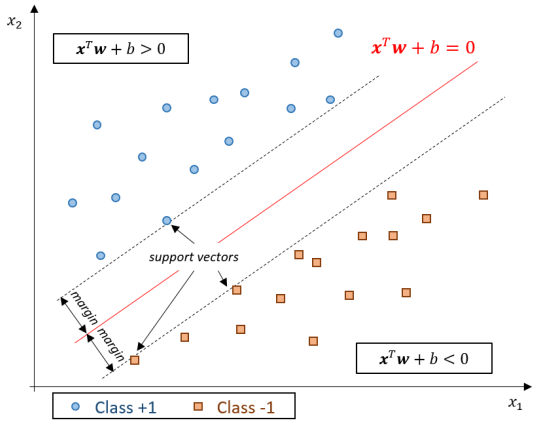


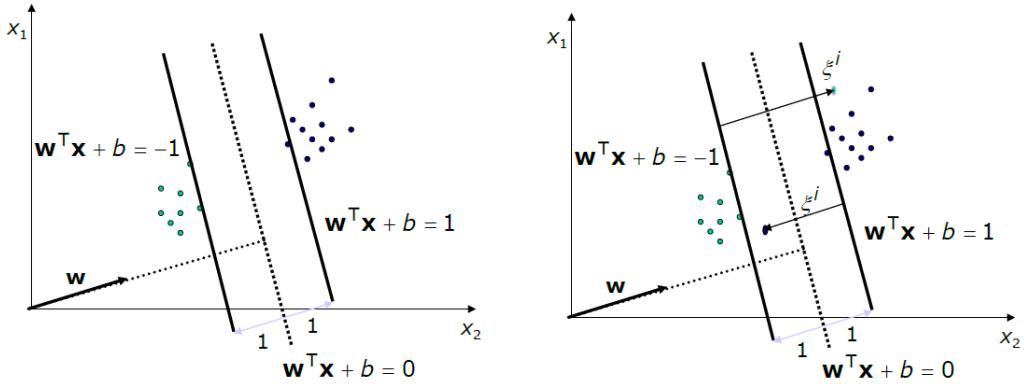
Figure 2.8: SVM classification of two classes using two features

If data are not linearly separable, the hard-margin formulation does not find a solution. Figure 2.9(a) illustrates the hard-margin formulation.

2. *Soft-margin SVM formulation*: the soft-margin formulation introduces slack variables ξ_i allowing some data to be misclassified. Using a training set $\mathcal{S}_L$, the goal is to find again $\mathbf{w}$ and b that solves

$$\min_{\mathbf{w},b} \ \|\mathbf{w}\|^2 + C\sum_i \xi_i \quad \text{s.t.} \quad \begin{cases} y_i^* \left(\mathbf{x}_i^T \mathbf{w} + b\right) \geq 1 - \xi_i, & \forall (\mathbf{x}_i, y_i^*) \in \mathcal{S}_L; \\ \xi_i \geq 0, \ \forall i. \end{cases} \tag{2.4}$$

The soft-margin formulation always finds a solution. Figure 2.9(b) represents the soft-margin formulation. There is obviously a trade-off between the margin width and the number of misclassifications. This trade-off is controlled by the parameter C. Notice that if C tends to infinity, the soft-margin formulation gets back to the hard-margin one.



(a) Hard-margin formulation (b) Soft-margin formulation

Figure 2.9: Two SVM formulations [1]

3. *Mapping to a higher dimensional space*: nonlinear decision boundaries can be obtained with SVM using the *kernel trick* [1]. The data are assumed to be linearly separable in a higher dimensional space and are projected into this new space using a function $\mathbf{\Phi}(\mathbf{x})$. This situation is illustrated with Figure 2.10. The SVM problem statement becomes then:

using a training set $\mathcal{S}_L$, find $\mathbf{w}$ and b that solves

$$\min_{\mathbf{w},b} \ \|\mathbf{w}\|^2 + C\sum_i \xi_i \quad \text{s.t.} \quad \begin{cases} y_i^* \left(\mathbf{\Phi}\left(\mathbf{x}_i\right)^T \mathbf{w} + b\right) \geq 1 - \xi_i, & \forall (\mathbf{x}_i, y_i^*) \in \mathcal{S}_L; \\ \xi_i \geq 0, \ \forall i. \end{cases} \tag{2.5}$$

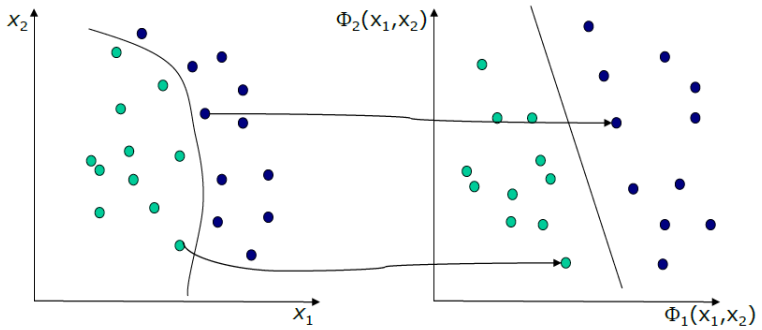


Figure 2.10: Mapping in a higher dimensional space using a function $\mathbf{\Phi}(\mathbf{x})$ [1]

The higher the mapping dimensionality, the higher the chance to be able to separate the data in a linearly manner in the new space. An explicit mapping using function $\mathbf{\Phi}(\mathbf{x})$ can however be very computationally expensive [1].

4. *The (Wolfe) dual problem*: an equivalent problem formulation exists and is defined as follows. Using a training set $\mathcal{S}_L$, find the Lagrange multipliers α_i such that:

$$\min_{\alpha_i} \ \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \ y_i^* y_j^* \ K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i \quad \text{s.t.} \quad \begin{cases} 0 \leq \alpha_i \leq C, \ \forall i; \\ \sum_i \alpha_i y_i^* = 0; \\ (\mathbf{x}_i, y_i^*) \in \mathcal{S}_L; \end{cases} \tag{2.6}$$

where $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j)$ is called the kernel, which avoids an explicit and costly mapping of the data. It can be shown also that the classifier function of (2.2) is equivalent to

$$y = f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i^* \ K\left(\mathbf{x}_i, \mathbf{x}\right) + b\right). \tag{2.7}$$

As a reminder, the data pairs $(\mathbf{x}_i, y_i^*)$ are known as they are contained in the training set $\mathcal{S}_L$. Feature vectors $\mathbf{x}_i$ for which corresponding α_i are non zero are called *Support Vectors* (SVs).

Several types of kernels exist [1]. The more common kernels are the following ones: linear, polynomial and gaussian. A hardware-friendly kernel has also been proposed in [49]. These four examples are shown in Table 2.3.

Kernel $K(\mathbf{x}, \mathbf{z})$	
linear	$\mathbf{x}^T\mathbf{z}$
polynomial (order p)	$\left(\mathbf{x}^T\mathbf{z} + 1\right)^p$
gaussian	$\exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}\right)$
hardware-friendly	$2^{-\gamma\|\mathbf{x}-\mathbf{z}\|_1}$

Table 2.3: Several common kernels

2.3.3.3 Multi-class SVMs

When dealing with more than two classes, several approaches are possible [1]. The first one is to generalise the SVM classification to multiple classes. This is not discussed here.

Two other methods exist:

1. *One-Versus-All strategy*: also called *One-Versus-the-Rest*, this strategy trains n binary classifiers to separate n classes. Each classifier separates one class against all other classes. The predicted class is the class of the most confident classifier (confidence can be given by the argument of the sign function in equations (2.2) and (2.7)).

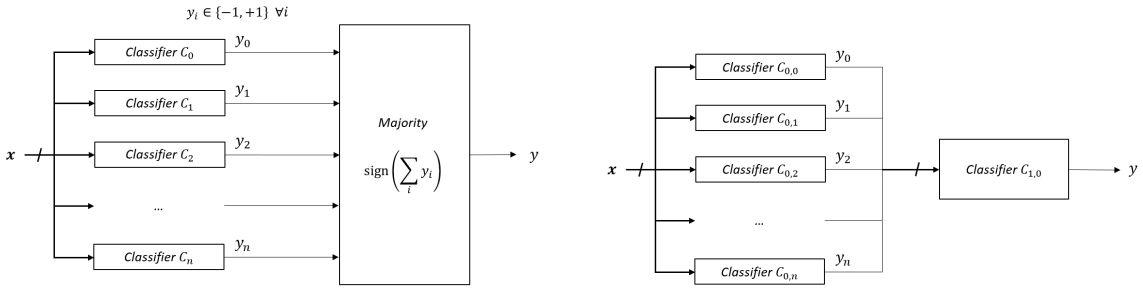
2. *One-Versus-One strategy*: this method trains $n(n-1)/2$ binary classifiers to separate n classes. Each classifier separates two classes. Several strategies exist to select the final class. They are not discussed here neither.

2.3.3.4 Combination of classifiers

Combination of classifiers can be very useful to improve the performances of a single classifier and has proved to be efficient in the field of BMIs [32]. Several types of combinations are possible. The two most popular classifier combinations used in BMIs applications are shown in Figure 2.11.

Voting technique uses several classifiers to classify the input feature vector $\mathbf{x}$. Each classifier can be different (one can be SVM, another can use LDA, another MLP, etc.) or can use different features as well. Each of them assigns an output label to the input feature vector. The final class is the one chosen by the majority. Figure 2.11(a) illustrates this situation. Voting is the most popular, probably because of its efficiency and its simplicity.

Stacking technique uses two levels of classifiers. Classifiers at level 0 classify the input feature vector $\mathbf{x}$ and produce an output label. Again, each of these classifiers can be different. Then, a meta-classifier takes as features the output of previous classifiers and assigns the final label based on that. Stacking is represented by Figure 2.11(b).



(a) Voting with binary classifiers (b) Stacking of classifiers

Figure 2.11: Two possible combinations of classifiers

2.3.4 Choice of the classification algorithm

How to choose a classification algorithm for EEG-based BMI research? Which classifier is chosen for the work presented in this document? This part aims to answer these questions. Table 2.4 gives a comparison of several algorithms that are earlier described or mentioned.

Algorithms	Linear Nonlinear	Generative Discriminative	Static Dynamic	Stable Unstable	Regularised
LDA	Linear	Discriminative	Static	Stable	No
SVM	Both	Discriminative	Static	Stable	Yes
MLP	Nonlinear	Discriminative	Static	Unstable	No
Bayes Quadratic	Nonlinear	Generative	Static	Unstable	No
HMM	Nonlinear	Generative	Dynamic	Unstable	No
KNN	Nonlinear	Discriminative	Static	Unstable	No

Table 2.4: Comparison of several classification algorithms (data coming from [32])

The choice of a classification algorithm has to take into account the nature of the features it will use [32]. As a reminder, features related to EEG signals were described in Section 2.3.2.

SVM has been frequently used with BMIs and has proved to be very efficient [32]. It has indeed very good properties. Its stability is needed as EEG data are non-stationary. Its simplicity makes it ideal for small training sets. Its regularisation properties are important against noise and outliers, together with the fact that it is discriminative. SVM has also an immunity to the curse-of dimensionality. Also, in its initial form, SVM is linear, but it can be made nonlinear quite easily using the kernel trick. For all these reasons, SVM is chosen for this work.

2.4 Conclusion

A lot of concepts and information are given in this chapter, which is divided into three main parts.

Various types of neural signals are first described with Section 2.1. As EEG is non invasive and easier to use for development, it is chosen for this work. That is the reason more details are given about it, such as its signal characteristics and the standard electrodes positioning systems for instance.

Then, a brief history of BMIs is given in Section 2.2, in order to identify the challenges and the current trends in the field. BMIs are at the crossroad of many disciplines, which makes them difficult to design because of the huge number of degrees of freedom.

Finally, Section 2.3 is devoted exclusively to classification algorithms. Their role is to decode brain thoughts based on input feature vectors. Therefore, they are crucial and the choice of the algorithm has to be made carefully. SVMs is chosen for this work.

A lot of aspects are covered in this chapter. These are used in the following chapters. Chapter 3 is about the experimentation system developed for this work and describes additional choices that are made to build a real BMI, based on the background given here. Chapter 4 is related to the implementation of a small BMI application and reuses the concepts linked to classification algorithms explained here.

Chapter 3

Experimentation System

Generally speaking, a BMI can be described by four blocks [31], as shown in Section 1.1 with Figure 1.2. These four blocks form the signal chain of the BMI and are recalled here with Figure 3.1.

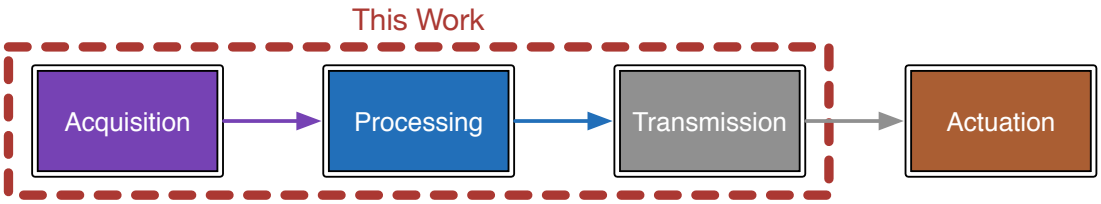


Figure 3.1: Conceptual block diagram of a BMI

Therefore, the four steps through which the useful signal goes are:

1. Acquisition: thanks to EEG electrodes and *Analog-to-Digital Converters* (ADCs), brain signals can be recorded and digitized in order to be processed;
2. Processing: brain signals are then processed to decode patient's intentions; an adequate control signal is generated; filtering, artefact removal, feature generation, and classification are examples of processing phases that can be performed;
3. Transmission: the transfer of the control signal to an effector can then be done, either with a wired or a wireless link;
4. Actuation: the desired action is finally performed by the effector, which can be a robotic limb, a computer screen or a wheelchair motor for instance.

The development of such a design is quite complex and can last for years. In this document, an experimentation system able to execute the first three steps is imagined and described. The objective is to be able to develop real-time EEG-based BMIs to help tetraplegic patients retrieve, at least partially, their movement autonomy. Therefore, the experimentation system is specially designed for development. As a consequence, it has to be flexible, reprogrammable and usable with various kinds of algorithms and applications. In this work, targeted applications are related to *Motor Imagery*, for which the imagination of limb movements is used to control the BMI [48] [11] [53].

To demonstrate that the system is working properly and that it can be used for the development of BMIs, a small spontaneous Motor Imagery application is implemented for this work.

The goal is to detect if a user thinks about moving his right/left arm. In response, the system lights the rightmost/leftmost LED of the board. When the user does not think about moving his arms (at rest for instance), no LED is turned on. Several specifications are related to this choice of application. Corresponding implementation details are described in Chapter 4.

This chapter presents the experimentation system developed for this work. An analysis of needs is first done in Section 3.1. Specifications for the entire system as well as for its three composing blocks (Acquisition, Processing and Transmission) are described and justified. Then, Section 3.2 presents the general architecture that is chosen. Choices about the main components are given and justified in Section 3.3. For instance, the electrodes positioning system is determined. Central electronic components such as ADCs and an FPGA are also selected. The design of the *Printed Circuit Board* (PCB) is then described in Section 3.4. Finally, the physical result is shown in Section 3.5.

3.1 Specifications

This first part gives the specifications for the entire experimentation system as well as for the individual blocks that compose it.

Table 3.1 shows the general specifications for the system. Several choices are made, and they are justified here below.

System	
Function	Development and testing of BMIs
Main features	Reprogrammable, flexible, portable, easy to use, real-time
Cost	Around 300 €
Brain signals	EEG
Invasiveness	None
PCB constraints	4 Layers, SMD or through hole
Power supply	5 V, 0.9 A
Interfaces	Electrodes, USB, microSD card, RF antenna, LEDs, buttons, switches

Table 3.1: System specifications

As mentioned above, the experimentation system is dedicated to the development of BMIs. For this reason, EEG signals are used, in order to avoid any surgical intervention. In addition, the system needs to be programmable and flexible, in order to easily make updates or changes of the application under development. Also, the system has to be fast enough, in order for the users to be able to control a BMI in real-time.

There are several constraints. A first constraint is related to the cost. BMIs have to be cheap, otherwise tetraplegic patients will not be able to buy and use them. The same rule goes for the system here. Therefore, a budget of 300 € is fixed for the entire system, in agreement with Synergia Medical.

Related to the PCB design, the number of layers is set to 4, which is a good trade-off between cost and routing's ease. The higher the number of layers, the higher the possibilities for the routing step, yet at a higher price. With four layers, some layers can also be used as power supplies or ground planes, which are useful to shield sensible analog parts against noise for instance. In addition, to enable an easy debugging during the PCB assembly, electronic components have

to be solderable manually. Therefore, only *Surface-Mounted Devices* (SMDs) with *Quad-Flat Package* (QFP) or through-hole components are allowed for this design. *Quad-Flat No-leads* packages (QFN) must be avoided as much as possible. *Ball-Grid Array* packages (BGA) are forbidden, as they are not solderable manually and because they increase the assembly and the costs of the components.

It is decided to power the system using a USB port. This enables the use of the board in a lot of different environments. As a consequence, the input voltage is limited to 5 V and the maximum input current is 0.9 A. USB can also be used for monitoring, debugging and communicating with the system. Other interfaces are added to the experimentation system to give it more flexibility and enable the development of a wide range of applications: a RF antenna enables wireless communications with effectors; a slot for a microSD card can be useful to store data or parameters; LEDs, buttons and switches are used to interact with the users.

The next parts describe the specifications for each of the three blocks that compose the development system. Required needs for the Acquisition, Processing and Transmission parts are successively determined.

3.1.1 Acquisition part

Here are described the required needs for the Acquisition part, which is responsible for the recording of the EEG signals and for their digitisation for subsequent processing. Table 3.2 presents the corresponding specifications. The Acquisition block is one of the most critical part of the system.

Acquisition		
Signal type	**EEG**	
	Amplitude	10-100 μV
	Targeted applications	Spontaneous Motor Imagery
	Bandwidth	8-12 Hz for μ rhythm
	Target cortex area	Motor cortex
Probing type	**Electrodes and positioning system**	
	Type	AgCl, dry and reusable
	Support	Helm or cap
	Cost	< 100 €
Data Conversion	**ADC**	
	Sampling Frequency	> 100 Hz
	Resolution	< 10 μV
	Other characteristics	Very low noise, high CMRR
	Cost	< 100 €

Table 3.2: Specifications for the Acquisition block

The signals to acquire are EEGs, whose bandwidth goes from 0.1 to 35 Hz. They also have amplitude values at the microvolt level, and are thus noisy. For a reminder about EEG characteristics, see Chapter 2.

As previously mentioned, the targeted applications for this work are linked to Motor Imagery. It consists of thinking about doing a particular movement. Corresponding brain signals have proved to be suitable to control a BMI and can be recorded in the μ frequency band. Therefore,

the smallest signal bandwidth that has to be kept for the targeted applications goes from 8 to 12 Hz. With Motor Imagery, motor areas of the cortex activate themselves. There are 17 electrode positions located close to these motor zones and they can be recorded for Motor Imagery applications [30]: C5, C3, C1, Cz, C2, C4, C6, CP3, CP1, CPz, CP2, CP4, FC3, FC1, FCz, FC2, FC4.

Dry and reusable electrodes are chosen, to avoid the use of inconvenient conductive gels. To maintain the SNR of EEG signals as high as possible, a good electrode conductivity is required. Gold or silver chloride (AgCl) electrodes are very common. For this work, silver chloride electrodes are used. To position correctly these electrodes, a positioning system is required, such as a helmet or a cap for instance. This choice is discussed more in details in Section 3.3. In total, the cost for needed electrodes and for the positioning system is set to a maximum of 100 €.

Once brain signals are acquired, they need to be digitised, in order to be processed in the PROCESSING block on a FPGA (see next part). To achieve this, ADCs are required. Their sampling frequency must be at least 2 times higher than the EEGs maximal frequency. Since EEG signals are weak and noisy, amplifiers are needed to reach a resolution of maximum 10 μV. In addition, these ADCs have to be very low noise and must have a high *Common-Mode Rejection Ratio* (CMRR) to avoid the amplifiers saturation. Any additional properties that ease the interface between ADCs and the PROCESSING block can be useful. The maximum price for needed ADC(s) is also fixed to a maximum of 100 €. Section 3.3 describes the ADCs chosen for this work and justifies this choice.

3.1.2 Processing part

The second most critical part of the system is the PROCESSING block. Its role is to decode the brain signals, retrieve the hidden user intention and generate an adequate control signal.

These steps are performed on a FPGA, which needs to be chosen with care. The choice of the FPGA is described with more details in Section 3.3. The decoding of user's intentions relies on a classification algorithm. SVM is chosen for this work because it has several excellent properties. For a reminder about SVMs, refer to Chapter 2.

Table 3.3 summarises the needs for the PROCESSING block and the FPGA.

PROCESSING		
Signal processing	**FPGA**	
	Clock frequency	$>$ 2 MHz
	Logic	$>$ 5 kLEs per binary classifier
	Memory	$>$ 11 KB per binary classifier
	Package	QFP
	Cost	$<$ 100 €
Classification	**SVM**	
	Learning	Offline with MATLAB
	Classification	Online on FPGA

Table 3.3: Specifications for the PROCESSING block

As a reminder, the application implemented in this work aims to decode whether the user thinks about moving his right/left arm or whether it does not want to move. Therefore, three

classes need to be separated: the *Left* class, related to (real or imagined) movements of the left arm; the *Right* class, related to (real or imagined) movements of the right arm; the *Rest* class, when no movement is imagined (for instance when at rest). To do so, the One-versus-all strategy is used (refer to Chapter 2 for more details about Multiclass SVMs), which requires in this case at least two binary SVM classifiers: one to separate *Left* against the two other classes; a second classifier to distinguish *Right* against the two other classes.

Therefore, two classifiers are implemented on the FPGA for online classification for the chosen application. Is is decided to perform the learning phase on MATLAB, offline, to save resources on the FPGA.

From there, several estimations can be made about the minimal requirements needed for the FPGA. In [49], Ruyz-Llata's group implements a binary SVM classifier with a hardware-friendly kernel, 10 SVs, 1024 features per SV and 8 bits per feature. If this design is taken as a reference, it can be estimated that the implementation of a single binary SVM needs about 5000 *Logic Elements* (LEs), 11 KB memory (if all learning parameters are put in memory) and a clock frequency higher than 2 MHz. For this work, as the system is designed for development, the FPGA has to be oversized. More complex algorithms and/or more classifiers have to fit in the FPGA. Also, the classification is not the only processing step: filtering, artefacts removal, display of information, feature generation are also possible signal processing steps that can be useful. That is why an FPGA with far more logic and memory is chosen. See Section 3.3 for more explanations about the choice of the FPGA. As EEG signals have a small bandwidth, the constraint on the clock frequency is not really an issue. The maximum budget allocated for the FPGA is set to 100 €. To ease the soldering, the package has to be of QFP type.

3.1.3 Transmission part

The third part of the system is related to the transmission of the control signal to an effector. For this block, it is decided to use a *Microcontroller Unit* (MCU) from Texas Instruments (TI). The main reason is that Synergia Medical knows and often uses these MCUs. Hence, a strong expertise and all the development materials (software, programmer, ...) are already available.

The selected component is the low-power CC430 from TI. It contains the well-known MSP430 microcontroller together with a RF core. Wireless communications can thus be implemented, which makes this chip ideal for the experimentation system of this work. In addition, it can offer additional computational power and memory to the system with its 32 KB of Flash and its 4 KB of SRAM. A SPI interface is also available to easily communicate with the FPGA as well as an USB interface to perform software simulations.

The CC430 is thus responsible for the management of the RF communications, for the interface with the FPGA and a microSD card and for communications via USB.

3.2 Global architecture

Based on the specifications presented in the previous section, a global architecture of the system is imagined. Figure 3.2 shows the architecture chosen for this work. It contains several blocks discussed above.

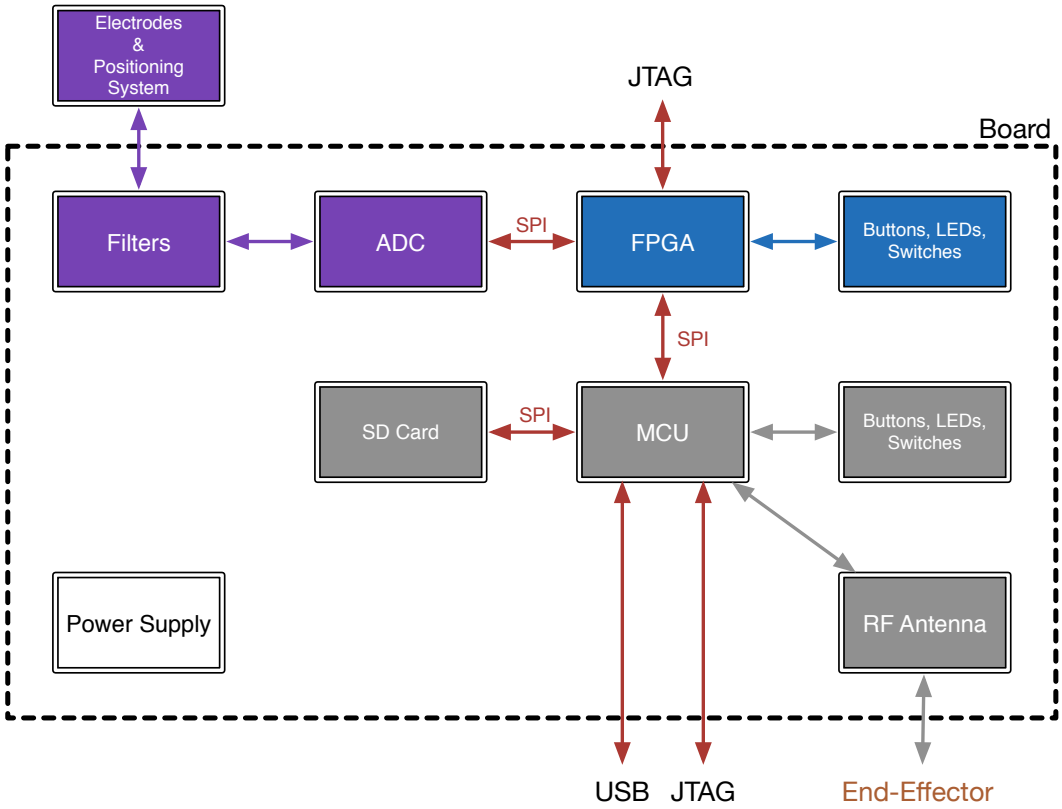


Figure 3.2: Global system overview

The electrodes are the first part of the system. They are attached to a positioning system to improve the comfort and their position accuracy. They are responsible for probing the EEG signals directly on the scalp of the patient. To avoid aliasing, these analog signals are filtered before being digitised by the ADC.

Once converted to digital, the data can be sent to the FPGA using SPI. The FPGA is used to implement algorithms that interpret the brain signals. These algorithms are implemented in hardware on a computer and then flashed on the FPGA using JTAG. The design interacts with the outside world through buttons, LEDs and switches. To lower the complexity and the size of the system, the FPGA has its own on-chip memory. It also interacts with a MCU using SPI.

The MCU is mainly used for interfacing. It is programmed with a computer using JTAG. It allows to easily export the brain data thanks to a microSD card connected through SPI. It can also interact with a computer through USB and it is programmed through JTAG. Some buttons, LEDs and switches are also available for the development. A on-chip RF core is available in the MCU which is linked to an antenna. It allows to send commands to a prospective actuator board wirelessly.

Choices related to the electrodes positioning system, the ADC(s) and the FPGA are not yet made. They are described more in details in next section.

3.3 Choice of components

Several important choices are not discussed in previous sections. This section is related to these choices. They concern the positioning system of the electrodes, the ADC(s) and the FPGA. Other possibilities are also explored and the optimal choice is always justified and described in details.

3.3.1 Electrodes and positioning system

To position the electrodes properly, there are several possibilities.

Indeed, a lot of EEG acquisition products have emerged these last years. The *Neurosky Mindwave* [41], the *Emotiv Insight* and the *Emotiv EPOC+* [17] are good examples. They can be called active EEG headsets since they embed all the electronics needed to acquire EEG signals. However, they are expensive and/or composed of too few electrodes. These electrodes are sometimes located at fixed positions, which does not necessarily match with desired electrodes positions for Motor Imagery.

Another possibility is to choose a passive EEG acquisition system (only the positioning system and the electrodes, without additional electronics). The most spread passive system with many electrodes is certainly a simple EEG cap, such as the *Comby EEG Caps* [18] or the *Electro-Caps* [12]. But once again, they are expensive (> 350 €) and are more frequently used in a clinical environment.

A last possibility is to build a hand-made system. As already mentioned, there exist a lot of open-source websites dedicated to BMIs. OpenBCI [42] is a good example. On this website, hardware and software tools can be found. They allow to measure, analyse and use physiological signals. For instance, plans and files for a 3D-printable helmet called *Ultracortex Mark III* are available. A tutorial for the assembly is also given. The *Ultracortex Mark III* is shown in Figure 3.3. This solution is chosen as positioning system for this work. This gives flexibility for future designs, because the helmet can be modified and adapted to new needs. In addition, since Synergia Medical owns a 3D-printer, it is easily printable.

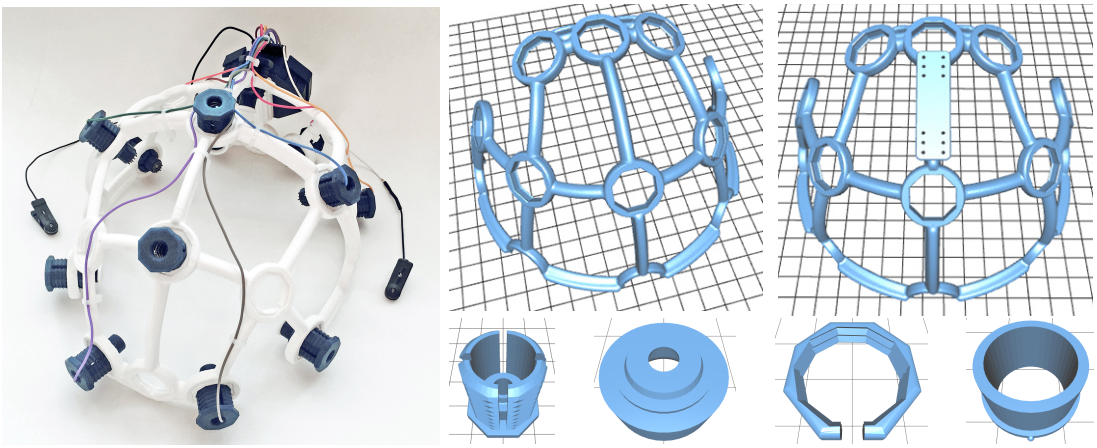


Figure 3.3: *Ultracortex Mark III* helmet from OpenBCI [42]

The *Ultracortex Mark III* helmet can have up to 21 electrodes and corresponds to the 10-20 positioning system. Unfortunately, electrodes positions related to Motor Imagery are mainly

available in the 5-10 positioning system. Only three positions of the *Ultracortex Mark III* are available for Motor Imagery: C3, Cz and C4. However, this is sufficient to develop basic algorithms [48] [11] [53].

It is also possible to print extra parts to get 8 new electrodes positions. As a consequence, in addition to positions C3, Cz and C4, the following electrodes positions can also be used for Motor Imagery: FC1, FC2, CP1 and CP2. These extra positions are not used in this work but can be useful for further improvements.

In addition to the standard silver chloride dry electrodes, two ear-clips are also added: one as a reference, and the other as a bias. The total cost for the printing material (about 10 €), 15 silver chloride electrodes plus the two ear-clips [26], and additional needed components (screws, springs, nuts and wires) is estimated to about 80 €. This does not include shipping costs.

As a remark, a new version of the helmet, called *Ultracortex Mark IV* with more electrodes positions will be released in July-August 2016. It could be used in future works to build more complex algorithms and systems with more electrodes.

3.3.2 ADCs

As previously discussed, acquired brain signals need to be digitised. This part discusses the choice of the required ADC. Since EEG signals are very weak, their digitisation is not easy and requires also extra analog circuits (e.g. instrument amplifiers, bias circuits, etc.) in addition to an ADC to get high quality conversions.

A first possibility could be to design these analog parts. This is quite tricky and requires a long time of development, as there are severe constraints (e.g. low noise or good resolution). In addition, as every component has to be soldered manually on the PCB, this would require more space and more time to assemble. The cost could also be high.

Fortunately, there also exist analog front-end chips available on the market, specially dedicated to the measurement of physiological signals. This second option is preferred and is studied further. In this category, two different chips may be adequate for the experimentation system developed here: the RHD2216 from Intan and the ADS1299 from TI.

The RHD2216 chip from Intan is a 16-bit ADC with 16 input channels. It has a lot of interesting features (e.g. electrophysiology amplifiers, low-pass and high-pass filters, a SPI interface, etc.) [29]. However, its very expensive price of about 260 $ can not be afforded for this work, despite its very good properties.

For the system of this work, the ADS1299 chip from TI is selected. For convenience, the ADS1299 is named ADS for the rest of this document. The ADS is a 24-bit *Sigma-Delta* ($\Sigma\Delta$) ADC devoted to EEG acquisition and has 8 input channels. It is also low-power (5 mW/channel), has a very low input-referred noise (1 μV_{pp} for 70 Hz bandwidth), a customisable data rate (from 250 SPS to 16 kSPS), a high CMRR (-110 dB), a built-in oscillator and a slave SPI interface [27]. Figure 3.4 shows a simplified block diagram of the ADS1299 chip. As can be seen, it has many other options such as programmable multiplexers and gain amplifiers, programmable control registers to enable various kinds of options, etc. The ADS is thus very flexible, which is a very useful feature for a development board. It costs about 50 €. In fact, two ADS's are used in the system, to have a total number of 16 input channels (and thus 16 electrodes).

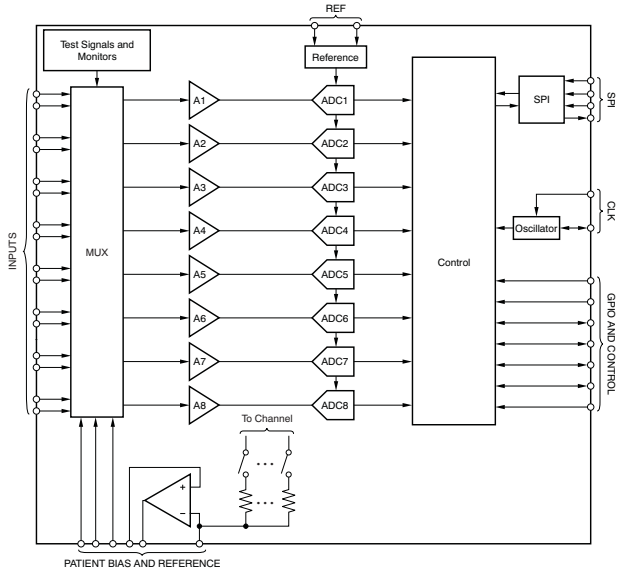


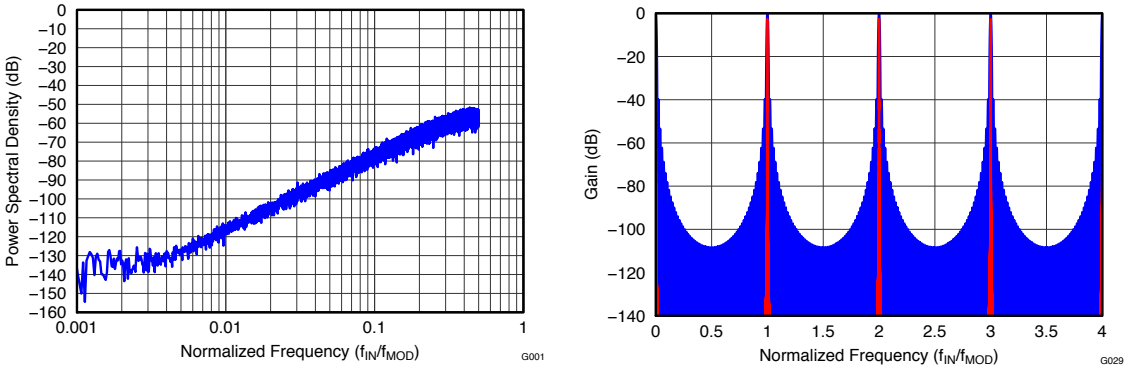
Figure 3.4: Simplified scheme for the ADS1299 [27]

Here below are described some additional details about the ADS. The ADS has 8 differential input channels, each channel having two input terminals (a positive terminal and a negative one). There is one multiplexer per input channel, and each of them can be independently configured. Thanks to these multiplexers, a common reference for all inputs can be used. This feature is really useful for EEG acquisition. Input channels can also be connected to internally-generated signals for tests, temperature and lead-off detection.

Lead-off detection allows to know if a good connection between user's head and the electrodes is available. The ADS is able to inject an excitation signal in the electrodes and it can measure the response to determine if an electrode is correctly connected or not. This is needed, in order to assure a good connectivity and be sure to acquire valid signals. As the impedance between an electrode and user's scalp can change over time (electrodes can move a bit and disconnect), lead-off detection is even more important.

A built-in bias drive (also called "Driven Right Leg") circuit is also available. It can be connected to chosen electrodes to sense their common-mode voltage and produces a negative feedback loop by driving the patient's body with an inverted common-mode voltage. This bias provides two interesting features: reducing the DC voltage between the electrodes and the reference, and then avoiding amplifier saturation; reducing the 50 Hz noise present on the patient's body.

The input data are sampled at a frequency $f_{mod} = f_{clk}/2$ (with f_{clk}, the operating digital clock frequency of the ADS). As mentioned above, the ADS is a $\Sigma\Delta$ ADC. Therefore, the noise is shaped to high frequency until $f_{mod}/2$ and a low-pass filter is required. This is illustrated with Figure 3.5(a). An on-chip digital decimation filter is available on each channel and can be used to filter out the noise at higher frequencies. It consists in a third-order *sinus cardinal* (sinc) filter. The decimation ratio can be configured thanks to a control register. It gives the possibility to find a compromise between resolution and data rate: filter more for higher resolution, filter less for higher data rates. Figure 3.5(b) shows the transfer function of the on-chip decimation filter for a fixed decimation ratio. Since the spectrum repeats itself every f_{mod}, an external R-C antialiasing *Low-Pass Filter* (LPF) with a cut-off frequency of at most $f_{mod}/2$ must be used at the input of every channel, before reaching the ADS.



(a) ADC noise shaping (b) Transfer function of the on-chip decimation filter

Figure 3.5: Illustrations of some properties of the ADS1299 [27]

The only missing feature of the ADS is the ability to band-pass filter the data with specific cut-off frequencies, to only keep the EEG signals bandwidth for example. Hopefully, as brain signals are sent to the FPGA to be processed, digital filters can be implemented on the FPGA.

3.3.3 FPGA

The last choice of component to describe concerns the FPGA. It is discussed in this part. Three FPGAs brands are explored: *Altera*, *Xilinx* and *Microsemi*.

A first selection can be made based on the retailed cost of the FPGAs. The remaining FPGA families are shown in Table 3.4. A comparison is given based on three criteria: the number of *Logic Elements* (LEs), the package type and the approximative retailed cost. Cost estimations come from a comparison of components prices available on `mouser.be/`, `www.digikey.be/` and `www.farnell.com/`

Brands	**Families**	**Logic** (LEs)	**Package**	**Cost** (€)
Altera	Cyclone	4k-301k	BGA, QFP, QFN	10-4150
	Max 10	2k-50k	BGA, WLCSP, QFP	3-115
Xilinx	Artix 7	17k-215k	WB, LFC, FC	22-290
	Spartan 6	4k-147k	WB, QFP	10-280
Microsemi	IGLOO	100-35k	BGA, CS, QFP, QFN	5-70
	IGLOO2	6k-146k	BGA, QFP	17-100
	ProASIC3	100-35k	BGA, CS, QFP	4-275

Packages
BGA: Ball Grid Array
QFP: Quad Flat Package
QFP: Quad Flat No-leads
CS: Chip Scale
WLCS: Wafer Level CS
WB: Wire-Bond
FC: Flip Chip
LFC: Lidless FC

Table 3.4: Comparison of FPGA families based on the amount of logic, the package type and the retailed cost [3] [55] [37]

A second selection can be made, based on the package type this time. As a reminder, only QFP packages are allowed for the FPGA, in order to ease its soldering. FPGA models available in other package types are thus automatically discarded. Models with less than 10 kLEs are also discarded because they do not contain enough logic to be able to develop a complete BMI. Remaining FPGA models are compared in more details with Table 3.5. Again, indicative costs are obtained from a comparison of components prices available on `mouser.be/`, `www.digikey.be/` and `www.farnell.com/`.

Families	**Devices**	**Dev. tools**	**Logic** (LEs)	**I/Os**	**Mult.**	**Memory**	**Package**	**Ind. cost** (€)
Cyclone II	EP2C20	Quartus	19k	142	26	234Kb	PQFP	45
Cyclone III	EP3C10	Quartus	10k	94	23	414Kb	EQFP	22
	EP3C16		15k	84, 160	56	504Kb	EQFP, PQFP	33
	EP3C25		25k	82, 148	66	594Kb	EQFP, PQFP	50
	EP3C40		40k	128	126	1134Kb	PQFP	90
Cyclone IV E	EP4CE10	Quartus	10k	91	23	414Kb	EQFP	21
	EP4CE15		15k	81	56	504Kb	EQFP	22
	EP4CE22		22k	79	66	594Kb	EQFP	49
Max 10	10M16	Quartus	16k	101	45	549Kb, 32K-296KB Flash	EQFP	42
	10M25		25k	101	55	675Kb, 32K-400KB Flash	EQFP	45
	10M40		40k	101	125	1260Kb, 64K-736KB Flash	EQFP	48
	10M50		50k	101	144	1638Kb, 64K-736KB Flash	EQFP	62
IGLOO2	M2GL010	Libero SoC	12k	84	22	912 Kb, 256KB eNVM	TQFP	23
ProASIC3/e	A3P1000	Libero SoC	11k	154	?	144Kb, 1Kb FlashROM	PQFP	75

Table 3.5: Comparison of FPGA models based on the available development tools, the package type, the amount of logic, the number of I/Os, the number of embedded multipliers, the memory size and an indicative retailed cost [3] [55] [37]

Among all these possible models, the 10M50 from Altera's MAX10 family is selected, essentially because it has the biggest amount of logic (50 kLEs). In addition, it embeds also more memory (block memory as well as Flash) and multipliers (144 9x9 bits embedded multipliers), has enough I/Os (101 user's I/Os in total) and an affordable price (around 60 €). Last but not least, it is programmed with QUARTUS, which has a big active community and is a well-known software for the authors. Therefore, it eases and speeds up the development. Its clock frequency is of about 52 MHz, far more than really needed, which is also good for future development. More details about the feature of the 10M50 can be found in [2].

As a remark, 10M40 and 10M50 models can be used interchangeably thanks to *pin mitigation*. They have exactly the same number of pins at exactly the same locations. In practice, a 10M40 FPGA is used because of stock shortage problems with the 10M50.

3.4 Hardware design

To connect all the chosen components together (ADC, FPGA, MCU, LEDs, microSD, etc.), a PCB is designed. This section introduces the schematics and the layouts, which can be found in Appendix A. Guidelines and advice have been found in development kits and in the components datasheets.

The first part of the board design is the power supply design, shown in Figure A.1. The board can be powered thanks to a USB cable or a basic connector linked to a 5 V voltage source. Six linear regulators are used to generate the six needed voltages: +5 V, -5 V, 2.5 V, -2.5 V and 2x 3.3 V (one analog and one digital). The -5 V is only used to generate the -2.5 V. The linear regulators are chosen to be very low noise (approximately 30 μV_{RMS} from 10 Hz to 100kHz for each).

The design of the input R-C filters can be found on Figure A.3. The cut-off frequency of the R-C LPFs is arbitrarily fixed to 6.78 kHz, which is far smaller than $f_{mod}/2 = 512$ kHz as required by [27]. Since passive filters are chosen, almost no noise is added to the EEG signals in this part of the circuit.

The ADSs are powered by three different voltage sources, as shown in Figure A.4 : 3.3 V for the digital part, -2.5 V and 2.5 V for the analog part. Bipolar supply is preferred to unipolar supply (0-5 V) for the analog part to get more flexibility on the input voltage range, negative voltages can then be probed without saturation. The bias circuits of the ADSs are connected together as mentioned in [27]. All the digital signals, including the SPI wires, are directly connected to the I/O's of the FPGA.

The 10M50 FPGA is shown in Figure A.5, together with the LEDs, switches and buttons that are connected to it. It is powered by the same supply than the one used for the ADS digital part. As a remark, future works can use a second PCB connected to this system and use the FPGA connections that are in excess, via the connectors of Figure A.2. This allows to use more ADSs if more channels are needed for instance, and it gives flexibility to the experimentation system. Connectors for the electrodes, JTAG plugs and the slot for microSD card are also visible in Figure A.2.

The connections related to the CC430 are visible on Figure A.6. Two different voltage supplies are used: one for the digital part (3.3 V) and one for the analog RF part (3.3 V, ultra low noise). The CC430 RF core is connected to a specific circuit which ends with a coaxial

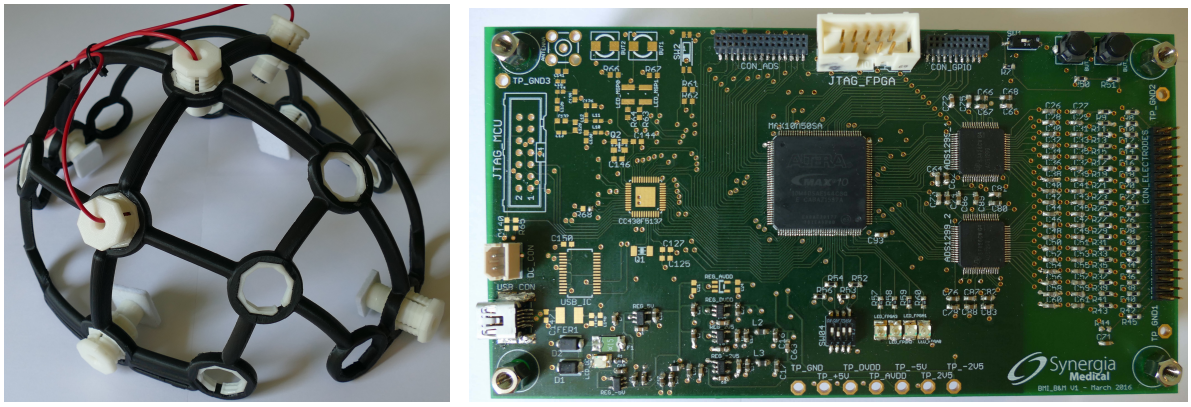
antenna, as mentioned in [28]. The SPI wires are connected to the microSD and the FPGA. The USB interface is connected to the FT232RL which is a USB to serial UART interface, as shown in Figure A.7. Finally, LEDs, switches and buttons are also available to interact with the MCU and then to ease the development.

As mentioned earlier, 4 layers are used to design the PCB layout. They can be seen in Figures A.8, A.9, A.10 and A.11. The top and the bottom layers are used to place the components and to route the connections between them. The second and the third layers are composed respectively by a ground plane and a digital 3.3 V plane.

3.5 Conclusion

In this chapter, the experimentation system imagined for this work is described in details. The starting point is the explanations related to global specifications. Specifications of composing blocks follow and the choices of critical components are justified properly. Finally, a hardware design is done to connect everything together and obtain a functional system.

The Ultracortex Mark III helmet described above has been printed and assembled, together with its electrodes. It is shown in Figure 3.6(a). The electrodes are linked to simple wires thanks to nuts, machines screws and glue. These wires can easily be plugged on the board thanks to simple connectors.



(a) Helmet with electrodes (b) PCB

Figure 3.6: Illustrations of the experimentation system

The board has been assembled manually. The result is shown in Figure 3.6(b). Simple tests have been carried out during the assembly and afterwards, to verify that everything was working as expected. As a remark, parts related to the MCU, the RF antenna and the microSD card have not been mounted for this work. The basic application implemented here (see Chapter 4) does not need the MCU. However, for future works, the MCU should be very useful.

To demonstrate that BMIs can be implemented and that the designed experimentation system is usable and works properly, an application is implemented. Chapter 4 is related to this subject and gives all the corresponding details.

Chapter 4

Implementation

A good BMI design does not only rely on a good hardware design, but also (and maybe mainly) on a performing implementation which must be robust, reliable and fast. To validate the hardware design presented in Chapter 3, a basic Motor Imagery application is implemented and described in this chapter. The application goal is to light a LED if the patient thinks about moving his right arm, to light an other LED if the patient thinks about moving his left arm and not to light anything otherwise.

The implementation of the application can be divided into three parts, as shown in Figure 4.1: the interface between the ADSs and the FPGA followed by the data filtering; the offline learning of SVM classification algorithms done with MATLAB; the classification algorithm made online on the FPGA.

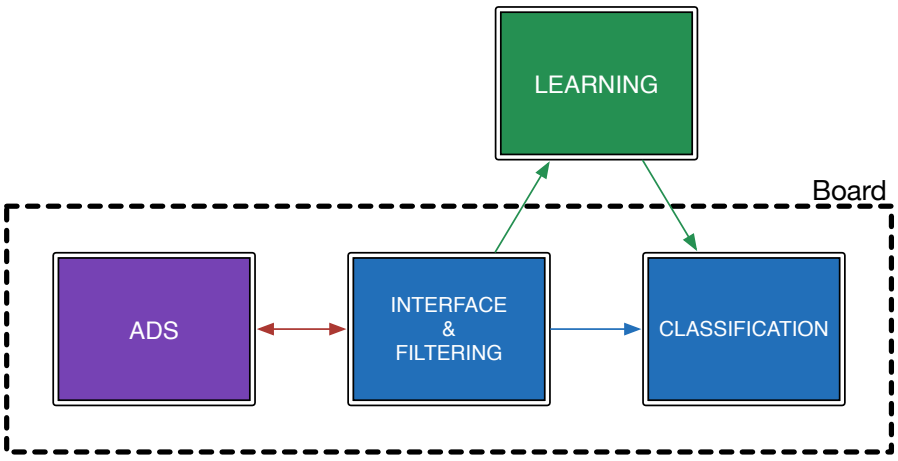


Figure 4.1: Implementation overview

Each section of this chapter explains one of these parts. First, the implementation of the communication between the FPGA and the ADSs through SPI is explained and tested in Section 4.1. Thanks to that implementation, digital EEG data can be extracted from the ADSs and used in the FPGA for the processing. The digital filtering of these data is also presented afterwards. Then, the software responsible for the learning algorithm is explained in Section 4.2. It requires many EEG training samples acquired thanks to the first part of the implementation. Finally, the hardware implementation of the classification algorithm can be found in Section 4.3. The parameters computed during the learning phase are used in the hardware classification to determine which LED has to be turned on.

4.1 From ADS to FPGA

The interface between the ADSs and the FPGA is detailed in this section. A brief description of the SPI protocol is first done. ADSs key features used in this work are then presented. Afterwards, the hardware implementation of the communication between the ADSs and the FPGA is explained in details. An implementation validation is proposed after that. Finally, the digital filtering of the data is explained.

4.1.1 Communication between FPGA and ADS through SPI

The *Serial Peripheral Interface* (SPI) bus is a synchronous serial communication working in full duplex and based on a master-slave architecture. Four wires are present in a basic communication: `SCLK` (the clock generated by the master and used to synchronise the communication), `SDO` (Serial Data Output), `SDI` (Serial Data Input) and `nCS` ("not Chip Select" controlled by the master and used to warn a slave that a communication has started).

As mentioned in Chapter 3, the ADS has a slave SPI interface. A master SPI interface is then implemented in the FPGA to allow a communication with the two ADSs. The connections between them are shown in Figure 4.2, where the red wires represent the SPI connections. The `START` signal is used to inform the ADS that it can begin to convert data, and the `nDRDY` signal is used to know if the ADS is ready to send new digital data.

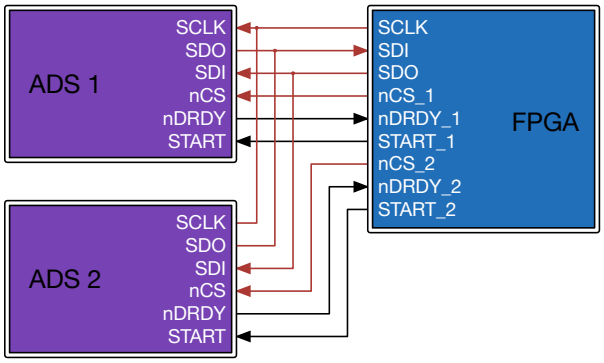


Figure 4.2: Main connections between FPGA and ADSs

Since the SPI is a serial bus, it is not possible to communicate with the two ADSs at the same time. In the FPGA, a signal `ADS_SELECT` is used to choose which of the `nCS`, `nDRDY` and `START` wires are used for the communication. In this manner, the implementation of all the communications does not need to be doubled. To be more concise, explanations and examples are given for one ADS only in the following parts.

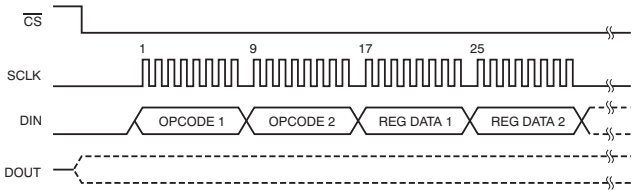
4.1.1.1 ADS features

A communication with the ADS always begins with the sending of an opcode by the master. The opcode corresponds to a specific command (`STANDBY`, `RESET`, `RDATAC`, `RREG`, `WREG`,...) and is composed of one or two bytes, depending on the command type. Some of them are stand-alone. For example, if the `RESET` opcode is sent, the ADS executes the command without answering. Other commands such as `WREG`, `RREG`, `RDATAC` need a specific second part in the communication, as shown in Figure 4.3. Only these three last commands are explained here in details since they are the ones used in this work. The other ones can be found in [27].

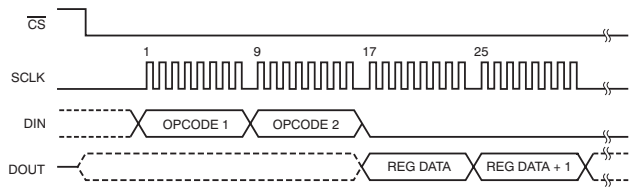
The `WREG` command allows to write in the ADS registers (see Figure 4.3(a)). The first opcode byte is divided into two parts. The 3 MSBs correspond to the `WREG` command and the rest of the byte corresponds to the first register address. The second opcode byte indicates the number of registers minus 1 in which the user wants to write. The following data bytes, sent by the master, are written in the corresponding registers by the ADS.

The `RREG` command allows to read ADS registers (see Figure 4.3(b)). The first opcode byte is divided into two parts. The 3 MSBs correspond to the `RREG` command and the rest of the byte corresponds to the first register address. The second opcode byte indicates the number of registers minus 1 that the user wants to read. The following data bytes, sent by the ADS, contain the value of the corresponding registers.

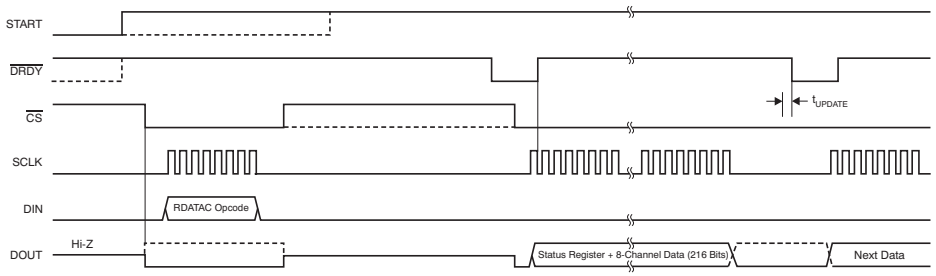
The `RDATAC` command allows to read continuously new converted data from the ADS without sending new opcode each time. The protocol is a little bit different from the other commands (see Figure 4.3(c)). Before sending the `RDATAC` opcode, the `START` signal must be pulled high to trigger the ADS conversion. Once this is done, the ADS begins to convert new data continuously at a given sampling rate f_s (specified in a register). As soon as the ADS has finished a conversion, the signal `nDRDY` is pulled low. At this moment, the master has to enable `SCLK` for 216 clock cycles (corresponding to 216 bits, which include 24 bits of data for each of the 8 channels plus 24 bits for the status register) to receive a complete data packet. The `nDRDY` is pulled high after the first clock cycle and pulled low again after $1/f_s$ seconds. The `RDATAC` opcode has not to be sent again for the next conversion. The master has simply to wait for the `nDRDY` to be pulled low by the ADS and enable again `SCLK` to receive a complete data packet.



(a) Write in a register (`WREG`)



(b) Read a register (`RREG`)



(c) Read data continuously (`RDATAC`)

Figure 4.3: Illustrations of several communications with the ADS [27]

4.1.1.2 Hardware implementation

The main connections between the ADS and the module responsible for the SPI interface, called `ADS_SPI_COM`, are represented on Figure 4.4. The top level module `ADS_SPI_COM` contains two submodules, called `SCLK_Generator` and `ADS_WRD`.

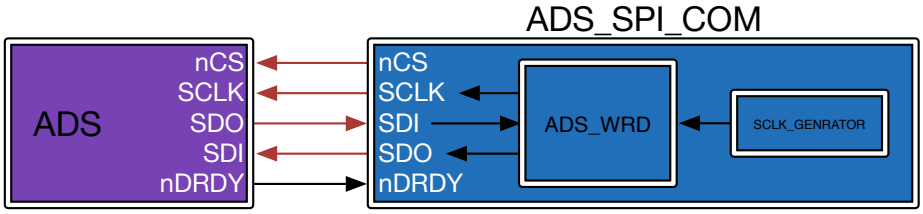


Figure 4.4: Main connections between the ADS and the module `ADS_SPI_COM`

The first submodule, called `SCLK_Generator`, generates the `SCLK` used in the SPI communication. A parameter can be adapted to choose the desired `SCLK` frequency. In practice, the SCLK frequency is arbitrarily set to about 6.5 MHz, which corresponds to the FPGA clock frequency divided by 8. As a remark, the maximum `SCLK` frequency supported by the ADS is 20 MHz.

The top level module, called `ADS_SPI_COM`, is responsible for the interface between the FPGA and the ADS. It is a huge FSM represented on Figure 4.5 (where green signals are active high and red ones are active low). The representation is slightly simplified for readability but the main features can be seen on it.

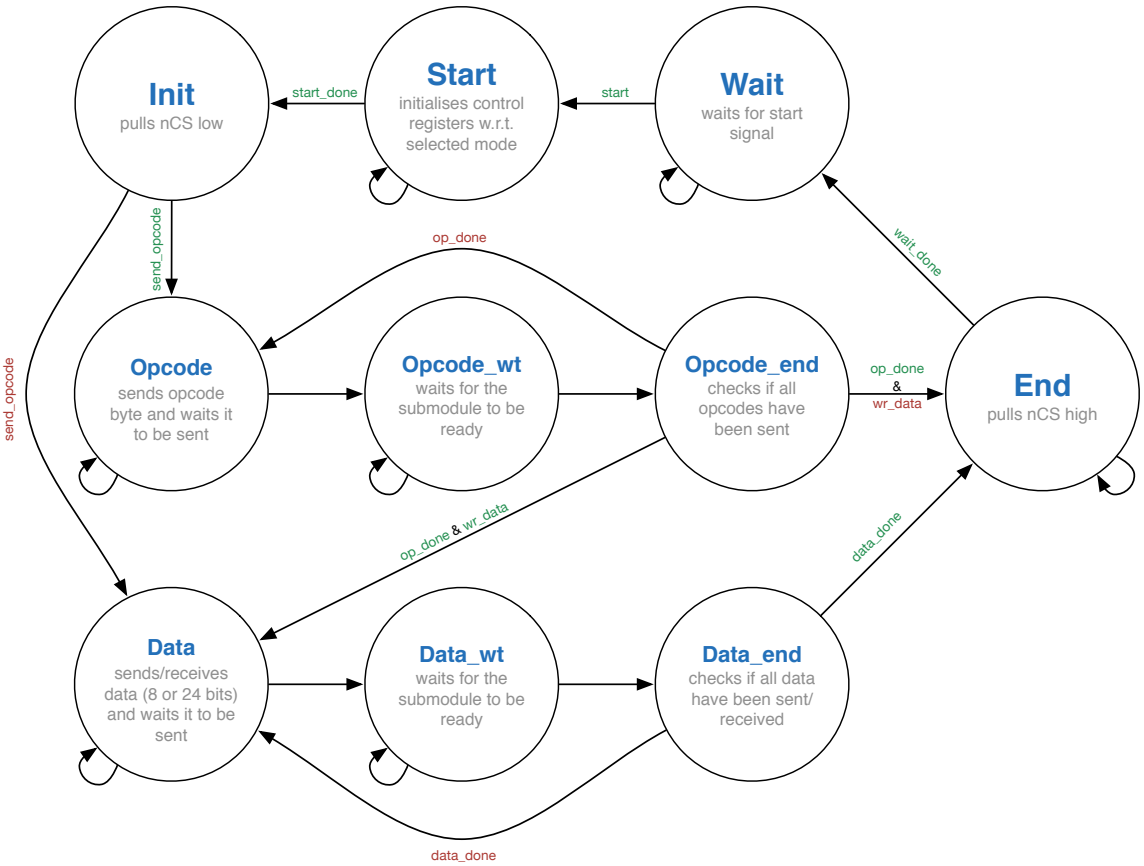


Figure 4.5: FSM of the module `ADS_SPI_COM`

An important input signal is the signal `mode` because it allows to choose which command the user wants to execute during the communication. There are 4 possible modes: the `OPCODE` mode that sends only one chosen opcode byte (as the `RESET` opcode seen earlier, for example); the `RREG` mode that reads the value of a desired ADS register; the `WREG` mode that writes a chosen value in a chosen ADS register; the `RDATAC` mode that reads the 216 bits corresponding to the converted data, when they are available.

The FSM initially waits for a `start` signal in the `Wait` state. In the `Start` state, control registers (as `send_opcode`, `wr_data`, ...) are initialised with different values depending on the selected mode. The `nCS` signal is pulled low in the `Init` state. After that, depending on the `send_opcode` value, the FSM goes to the `Opcode` state or to the `Data` state.

In these two states, the submodule called `ADS_WRD` is enabled. It is a small FSM that controls three of the four SPI wires (`SCLK`, `SDI`, `SDO`). Its 3 states are described in Figure 4.6. It also has control signals that allow to choose if the user wants to write or read data through the SPI and to choose the size of the data (a byte or 24 bits). These control signals are chosen in the main module `ADS_SPI_COM` depending on its current state and on the selected mode.

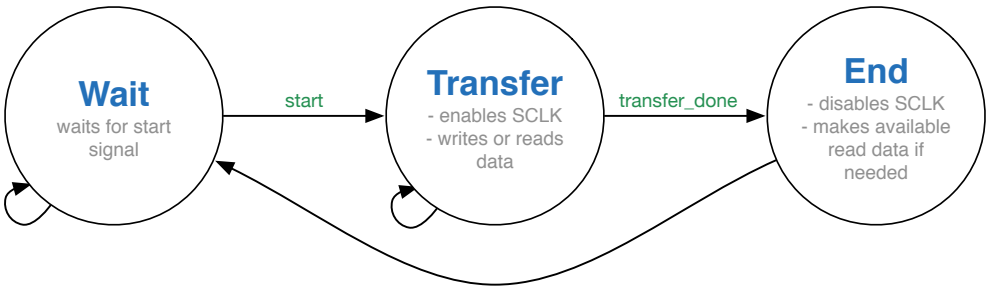


Figure 4.6: FSM of the module `ADS_WRD`

In summary, states `Opcode` and `Data` enable the submodule `ADS_WRD` with different control signals, states `Opcode_wt` and `Data_wt` wait for the submodule to be ready, and states `Opcode_end` and `Data_end` check if all the opcodes/data have been sent or received. If not, the FSM comes back respectively to the `Opcode` state or the `Data` state. If all the opcodes have been sent and if some data have to be sent or received (`wr_data` high, for the `RREG` or the `WREG` commands for example), the FSM goes to the state `Data`. Otherwise, it goes to the state `End`. If all the data have been sent, the FSM goes from the `Data_end` state to the `End` state that pulls `nCS` high after a little delay, as required in [27]. The communication is then over and the FSM goes back to the `Wait` state where it waits for a new `start` signal from the user to restart a communication.

An Altera `NIOS II` processor is also implemented on the FPGA to ease the interactions between the user and the ADSs. Thanks to that, it is possible to control the module `ADS_SPI_COM` directly in software (in C) and to export results more easily. All the communication is made in hardware, which provides a great robustness and reliability, but the interface can be done in software, which provides more flexibility. Many basic functions are developed to easily interact with the ADSs.

4.1.1.3 Hardware validation

Many tests are performed to validate the module `ADS_SPI_COM`, but only one is presented in this work. The mode `WREG` is tested first on ModelSim and then on the FPGA thanks to SignalTap.

The goal of the test is to export the internal oscillator signal of the ADS to the FPGA, thanks to the `CLK` pin. The corresponding control bit, called `CLK_EN`, is the 5th bit of the `CONFIG1` ADS register. Its default value is "0" and has then to be pulled high.

As a reminder, the mode `WREG` allows to write a chosen value in a chosen ADS register. The corresponding protocol is explained earlier and is described in Figure 4.3(a).

The first opcode of this test is divided into two parts: the 3 MSBs corresponding to the `WREG` command ("`010`") and the 5 last bits corresponding to the address of the `CONFIG1` register ("`00001`"). The first byte to send is then equal to "`0x41`". The second opcode byte value is always "`0x00`" in this work since the implementation deals only with one register at a time. The default value of the `CONFIG1` register is "`0x96`". Since only the 5th bit has to be pulled high, the data that has to be sent is "`0xB6`", which corresponds to the third byte of the communication.

Figure 4.7 shows the simulation test on ModelSim. The `start` signal enables the FSM. The different states can be seen (or imagined thanks to Figure 4.5) next to the `ADS_SPI_COM_state` signal. The first opcode corresponds well to "`0x41`", the second to "`0x00`" and the data byte to "`0xB6`", as expected. The `nCS` signal is well pulled low at the begin of the communication and pulled high again at the end.

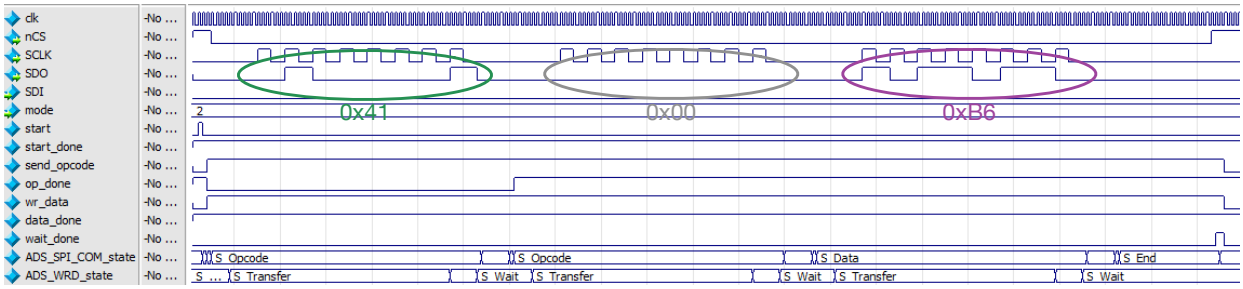


Figure 4.7: Simulation of the module `ADS_SPI_COM` on ModelSim

Figure 4.8 shows the simulation test directly on the FPGA, thanks to SignalTap. Again, the opcode bytes and the data byte correspond to the expected values. It is clearly visible that communication works correctly since the `ADS_CLK` begins to toggle soon after the sending of the data byte, as expected.

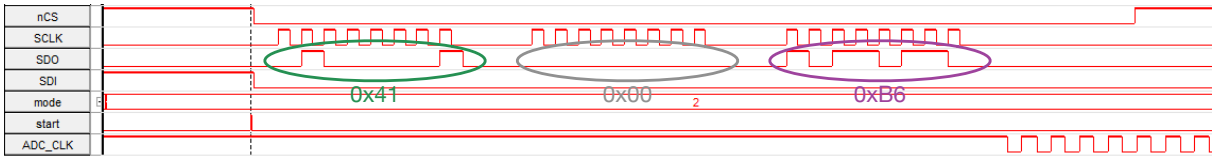


Figure 4.8: Simulation of the module `ADS_SPI_COM` on SignalTap

4.1.2 Digital Band Pass Filter

The data rate of the ADS is fixed to 250 Hz, the rate at which the SNR is the highest. As a consequence, the bandwidth of the converted signal is spread from 0 to 125 Hz. As a reminder, the EEG bandwidth goes from 0.1 to 35 Hz and the μ rhythm bandwidth from 8 to 12 Hz. A *Band Pass Filter* (BPF) is then needed to only keep the band of interest, and then attenuate the prospective offset and the ambient noise (e.g. supply noise at 50 Hz in Europe) present in the acquisitions.

4.1.2.1 Hardware implementation

To limit the bandwidth of acquired EEG data, a *Finite Impulse Response* (FIR) filter is implemented digitally on the FPGA. FIR filters are simple and stable. The output of a FIR filter $y[n]$ is simply a linear combination of the N last values of the input sequence x, with N the filter order:

$$y[n] = \sum_{i=0}^{N} b_i \, x[n-i]. \tag{4.1}$$

A FIR filter has a group delay that depends on the number of coefficients $N+1$ used in the filter. The number of coefficients is directly linked to the filter complexity. Thus, since the delay and the size (in LEs) are critical in the application developed in this work, the filter has to be as simple as possible. The desired filter specifications are shown in Table 4.1. The selected BPF bandwidth is chosen so as to keep at least the μ rhythm band (8 to 12 Hz) which is related to Motor Imagery. A BPF with a bandwidth that goes from 7 to 35 Hz is simple and fast enough for the application.

FIR BPF Specifications		
Frequency specifications		
	Input sample rate	250 Hz
	Stopband frequency 1	0.1 Hz
	Passband frequency 1	7 Hz
	Stopband frequency 2	35 Hz
	Passband frequency 2	45 Hz
Magnitude specifications		
	Stopband attenuation 1	60 dB
	Stopband attenuation 2	60 dB
	Passband ripple	0.01 dB
Filter implementation		
	Direct-form symmetric FIR	

Table 4.1: Specifications for the BPF

The filter is built thanks to a MATLAB tool called `filterbuilder`, which allows to design a FIR and to quantise it with a specific level of precision (specific number of bits). It is then possible to export it in VERILOG, which is directly usable on FPGA. Figure 4.9 shows the magnitude response of the reference filter (full precision) and the quantised filter, built based on the specifications of Table 4.1. There is almost no visible difference between the two filters. To meet the desired specifications of Table 4.1, 126 coefficients are needed. The group delay is then equal to 63 samples since a symmetric implementation is chosen. As the data rate is fixed to 250 Hz, this corresponds to approximately 252 ms.

The hardware design proposed by MATLAB requires to implement one filter per channel since the 63 previous values of the input are stored in registers and used to filter the input data. It is then not possible to use only one module several times for several different channels.

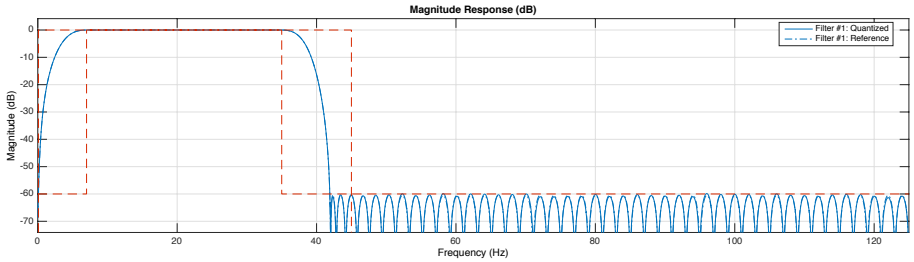
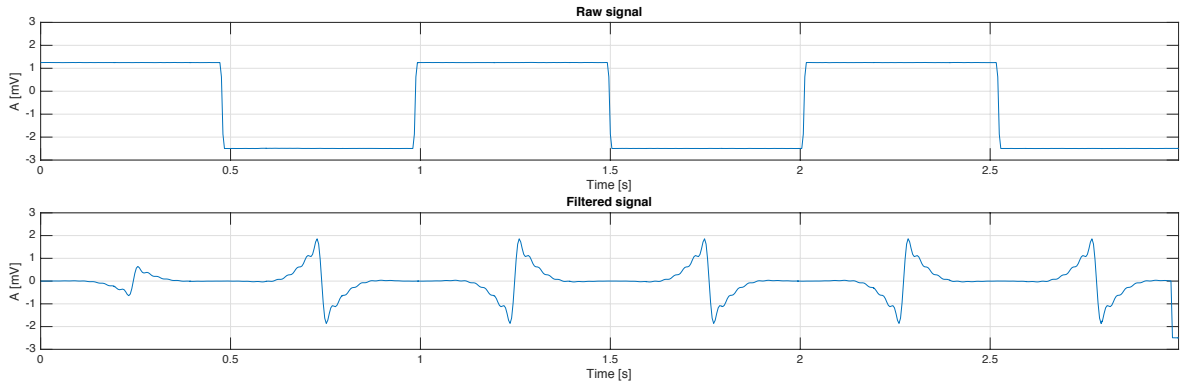


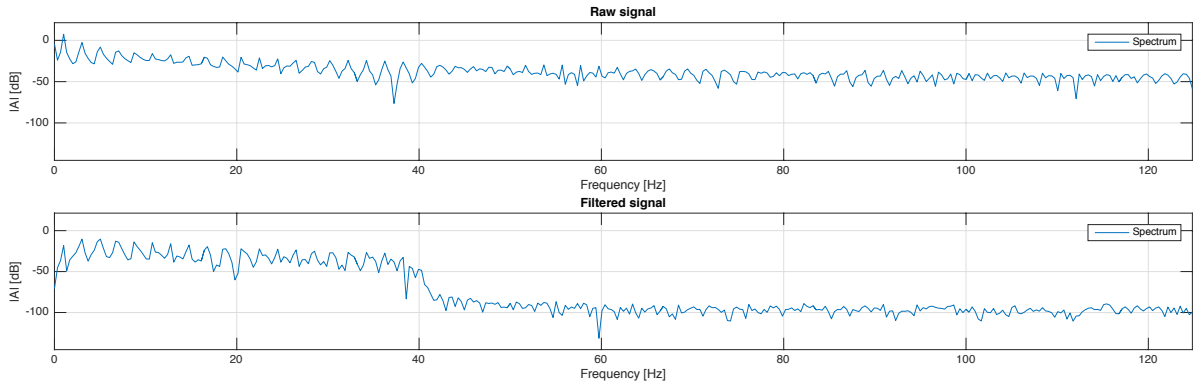
Figure 4.9: Magnitude response of the FIR BPF

4.1.2.2 Hardware validation

The hardware implementation of the filter, slightly modified compared to the filter proposed by MATLAB (a `ready` signal is added), is named `myBPF`. In this part, its implementation is validated.



(a) Temporal representation



(b) Spectral reprentation

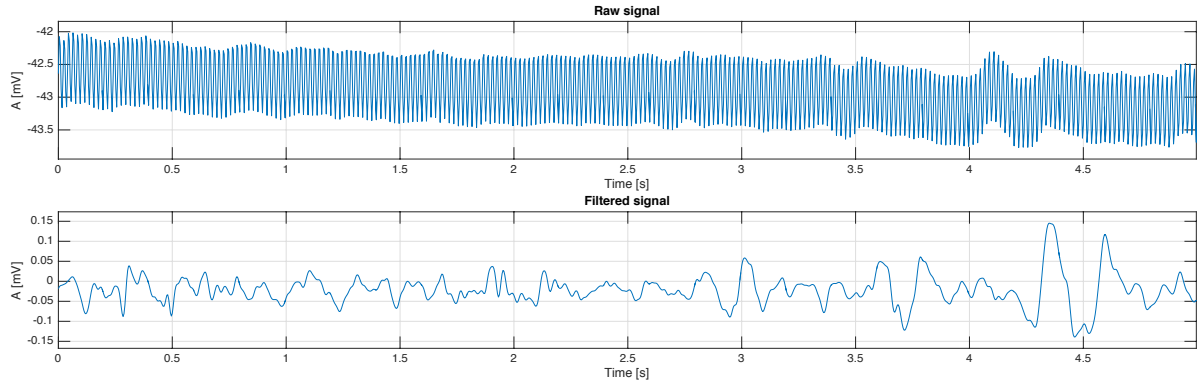
Figure 4.10: Illustrations of filtered data using MODELSIM

It is first tested on MODELSIM with a square waveform. The result is presented in Figure 4.10. The delay of 252 ms is visible in Figure 4.10(a). The offset of the raw signal is well attenuated in the filtered signal. Figure 4.10(b) shows the spectra of the raw signal and the

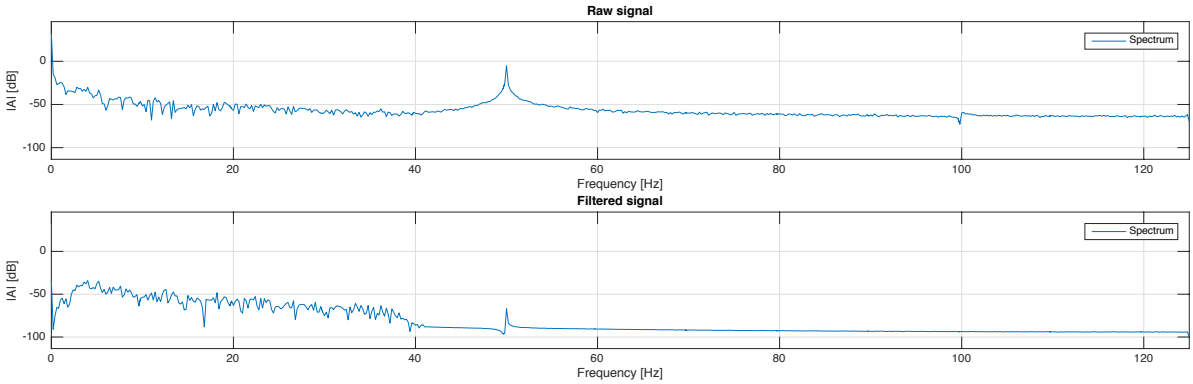
filtered signal. The frequencies located outside the band of interest are well attenuated by at least 60 dB, as expected.

Another test is performed directly in the FPGA, this time with a real EEG signal probed thanks to the experimentation system explained in Chapter 3. The probed signal is filtered thanks to the module `myBPF`. The raw and filtered data are extracted from the FPGA thanks to the `NIOS II` and displayed thanks to MATLAB. The results are shown in Figure 4.11.

The two main benefits offered by the filter are clearly visible here: the big offset (compared to the amplitude) as well as the big 50 Hz noise have to be attenuated, and the filter does it perfectly.



(a) Temporal representation



(b) Spectral reprentation

Figure 4.11: Illustrations of filtered data using the FPGA

4.2 Offline Training

To decode if a user thinks about moving his left or right arm, a classification algorithm is used. As described in Chapter 2, SVM is chosen for the application implemented in this work. The goal is to classify EEG data in real-time on FPGA. The classification has to determine whether the current feature vector belongs to the *Left* class, the *Right* class or to the *Rest* one. As a reminder, these three classes are defined in Chapter 3.

The linear kernel is chosen for this work, because it is the most simple case of SVM algorithm. Other kernels can be used in future works, to obtain non-linear decision boundaries and (maybe) improve the results for instance.

The hardware implementation details for the online classification are given in Section 4.3. This part is about the learning of the parameters needed for the online classification. The features generation is first discussed. Then, some details are given about the learning. Several concepts introduced in this part are reused in next section.

4.2.1 Features generation

As discussed in Chapter 2, several types of features can be used to control a BMI. In this work, *Autoregressive* (AR) coefficients are used. This type of features has already been used in the field of BMIs [48] [21] [19] [30]. AR coefficients seem to be easier to compute on FPGA than other features, that is mainly the reason why they are chosen for this work.

An AR model is used to approximate a signal. This parametric modelling technique can be formulated either as a spectral matching problem in the frequency domain, or as a linear prediction problem in the time domain [44]. The second approach, which is more intuitive, is briefly described here. It assumes that the current sample value $s[n]$ of a data sequence can be approximated by a linearly weighted sum of the p most recent sample values $s[n-1]$, $s[n-2]$, ..., $s[n-p]$ plus an error term $e[n]$. The value p is the order of the model and is chosen much smaller than the sequence length. The current sample value can then be expressed as

$$s[n] = -\sum_{i=1}^{p} a_i \; s[n-i] + e[n], \tag{4.2}$$

where a_i denotes the i^{th} autoregressive coefficient of the model.

To find optimal values for the model coefficients by minimization of the *Mean Square Error* (MSE), the so-called *Yule-Walker equations* have to be solved using the *Levinson-Durbin algorithm* [44] [20]. This requires to compute the autocorrelation functions r_0, r_1, ..., r_p of the sequence, which are estimated by the following expression for a sequence of N samples:

$$r_k = \sum_{i=k}^{N} s[n] \; s[n-k] = \sum_{i=0}^{N-k} s[n] \; s[n+k]. \tag{4.3}$$

It is assumed that the length of the sequence N has to be greater than $10p$ in order for the autocorrelation values r_k's to be enough accurate [20]. In addition, for a real sequence, it can be shown that r_0 is always positive and that

$$r_0 \geq |r_i| \geq 0 \qquad \text{for } i = 1, \; ..., \; p. \tag{4.4}$$

This interesting property is useful for the implementation on FPGA. As an additional remark, sometimes the autocorrelation values r_i's are defined as (4.3) but divided by a division factor. In fact, this division factor does not matter here because the information which is needed is the relative importance of a given autocorrelation value with respect to the others. Divide or multiply all the autocorrelation coefficients given by (4.3) by a factor does not have any effect for the following steps (when working in full precision). This property is used for the implementation on FPGA as well. Refer to Section 4.3 for the implementation details.

The Levinson-Durbin algorithm computes the AR coefficients values based on the autocorrelation functions of (4.3). It is described here below with Listing 4.1 [44] [50] [34] [20]. As a

remark, coefficient a_0 is not really an AR coefficient but is introduced to ease the algorithm. Only a_1, a_2, ..., a_p values are considered as AR coefficients. Therefore, for an order p AR model, there are p AR coefficients, as shown with (4.2).

```
// Initializations
a_0 = 1;
E_0 = r_0;

for i = 1 to p
    // Reflection coefficients
    k_i = 0;
    for j = 0 to i - 1
        k_i = k_i - a_j * r_{i-j};
    end
    k_i = k_i / E_{i-1};

    // AR values
    c_i = k_i;
    for j = 1 to i - 1
        c_j = a_j + k_i * a_{i-j};
    end

    // Error
    E_i = (1 - k_i^2) * E_{i-1};

    // Update AR values
    for j = 1 to p
        a_j = c_j
    end
end
```

Listing 4.1: Levinson-Durbin algorithm

4.2.2 Learning

To separate three classes (e.g. *Left*, *Right* and *Rest* in this work), the One-Versus-All strategy is used and two binary SVM classifiers are trained. These two classifiers are named `SVM_Left` and `SVM_Right` in this document. Classifier `SVM_Left`/`SVM_Right` is trained to distinguish the *Left*/*Right* class from the two others. Their parameters, needed for the online classification, have to be learned. This is done in MATLAB, using functions from the *Statistics and Machine Learning Toolbox*. Here below are described the main steps followed to obtain the SVM parameters for each classifier, with the help of Figure 4.12. Several choices are justified afterwards.

1. *Acquisition*: The first step is to acquire EEG data with the experimentation system, for each class, in order to form good learning sets. To do this, 40 trials are recorded for each class. Each trial lasts 8 s and contains time samples for the 3 chosen EEG channels (C3, Cz, C4). As a reminder, each channel is bandpass filtered with a digital FIR filter (refer to previous section). During the *Left*/*Right* class trials, the left/right arm is continuously moved up and down. During the trials of the *Rest* class, no movement is done. As a sampling frequency of 250 Hz is used by the ADS, each trial contains 2000 samples per channel, that is to say 6000 samples in total for the three channels together! The resulting database of EEG data thus contains 120 trials in total and that corresponds to situation 1 in Figure 4.12(a). To each trial corresponds a known label (*Right*, *Left* or *Rest*) which is related to the type of movements done during the trial.

2. *Features generation*: Then, the data have to be converted into features. This is done in two phases.

   (a) Each acquired trial of 8 s is first cut into 4 successive subintervals of 2 s. This is represented by situation 2 in Figure 4.12(a). The number of trials is thus multiplied by 4, resulting in 160 trials per class.

   (b) Then, on each of these trial subintervals and for each channel, AR coefficients are computed with the earlier described Levinson-Durbin algorithm. The order of the AR model is chosen to be equal to $p = 6$. As a consequence, the dimension of the database is highly reduced, as now a trial is represented by 6 features per channel, or equivalently 18 features per trial (instead of 1500 samples per trial). This is shown with situation 3 in Figure 4.12(a). As a remark, a trial can from now be called feature vector.

3. *Learning, validation and test sets*: The third step is to form three kinds of sets, respectively used for training, validation and testing.

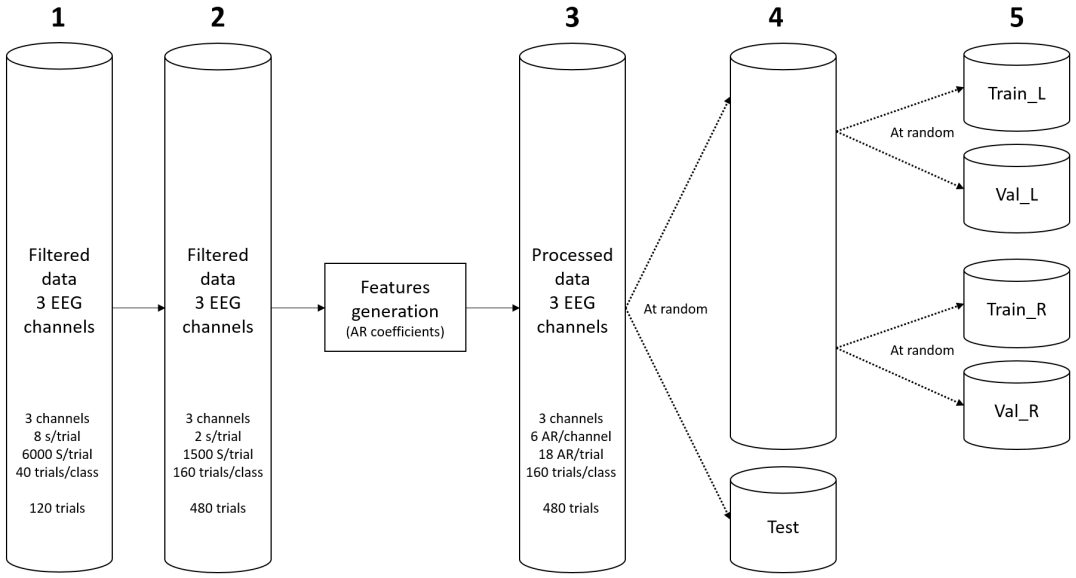
   (a) A `Test` set is needed to evaluate the performances when combining the two classifiers together. This is shown with situation 4 in Figure 4.12(a). As some feature vectors are more relevant than others for the classification (and as this is not easy to know in advance), the cutting of the database into sets is done randomly.

   (b) Feature vectors that are not assigned to the `Test` set are then used to form the training and validation sets for the two classifiers. The separation is again performed randomly. Four sets are thus formed: sets `Train_L` and `Val_L` correspond respectively to learning and validation sets for classifier `SVM_Left`; sets `Train_R` and `Val_R` are used for classifier `SVM_Right`. Notice that the union of `Train_L` and `Val_L` is equal to the union of `Train_R` and `Val_R`. This is illustrated with situation 5 in Figure 4.12(a).

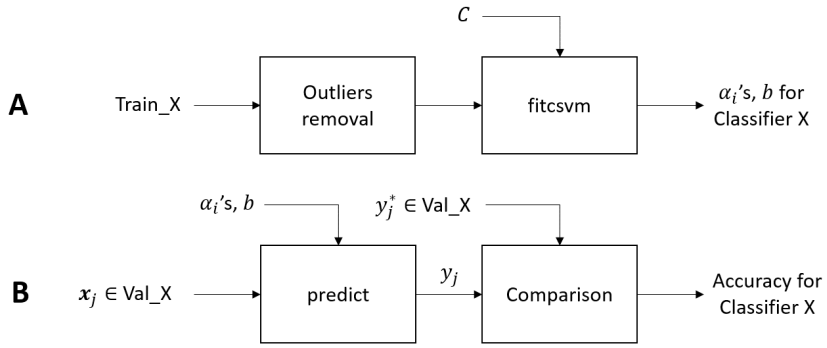
4. *Learning*: The next step is to learn the parameters for classifiers `SVM_Left` and `SVM_Right`. The parameters that need to be found are the Lagrange multipliers α_i's and the offset b, for each classifier. Refer to Chapter 2 for a reminder about SVM. The model parameter C of equation (2.6) has also to be determined. This is done by using the MATLAB function `fitcsvm` on the learning sets `Train_L` and `Train_R` for several values of C. Function `fitcsvm` outputs a structure with the indices of the SVs in the learning set and the needed parameters α_i's and b. As a remark, to improve the results, outliers are removed from the learning sets `Train_L` and `Train_R` before the learning step. Situation A in Figure 4.12(b) corresponds to the explanations given here.

5. *Validation*: The performances (e.g. the number of misclassification errors) can not be evaluated on the training set only, because good performances on this set can be the consequence of overfitting. The real goal is to be able, from the learning set, to generalise and have good performances with new data. That is the reason why validations sets are used. As they are not used for the learning, they can be considered as new data from the classifiers point of view. The choice of the best C value and corresponding SVM parameters are evaluated thanks to validations sets `Val_L` and `Val_R`. The MATLAB function `predict` is used to classify input data using the learned parameters. Its output is a label y, that can be compared to the true known label y^*. This is illustrated with situation B on Figure 4.12(b).

6. *Test*: The final step is to combine the two classifiers (as it is done in real-time on FPGA) to evaluate their performances on the `Test` set, which can be viewed as totally new data for both classifiers.



(a) Formation of learning, validation and testing sets



(b) Learning and validation phases

Figure 4.12: Steps followed for the learning of an SVM algorithm

Several choices must be justified. First of all, only three electrode positions are used for the application of this work [48] [11] [53]: C3, Cz and C4. These are the only channels available with the helmet that are really related to motor areas of the cortex and Motor Imagery applications.

Secondly, it has been chosen to perform real movements during the acquisition phase and the formation of the learning databases, because this eases the acquisition protocols. By performing the desired movement, it is sure that a movement-related brain activity is recorded. For imagined movements, more complex acquisition protocols are needed and extracted data need to be analysed and verified more carefully.

Thirdly, it is decided to cut each acquired trial of 8 s into subintervals, to increase the database dimensions from one hand, and catch more accurately temporal details on the other hand. The length of the subintervals is fixed to 2 s for the offline learning. This value is arbitrary

but seems to be a good tradeoff: a lower subinterval duration increases the capability to catch accurately temporal details and increases the size of the learning databases, but the risk to use only parts of the recorded movement-related brain activities is higher; and conversely. For the online classification, the subinterval duration is fixed to 1 s, to improve the system response and perform the operations in real-time. A lot of different duration values are used in the literature and could be tried in future works [30] [21] [7] [19] [48].

A last choice concerns the AR model order p. It is arbitrarily fixed to 6, as it was also done in other works [21]. A smaller value would result in a poorer model that fits less the data, while a higher value would result in more features and a higher dimensional space for the classification. Therefore, the value of $p = 6$ seems to be a good compromise.

4.3 Online Classification

The hardware implementation of the classification made on the FPGA is presented in this section. The corresponding diagram is shown in Figure 4.13.

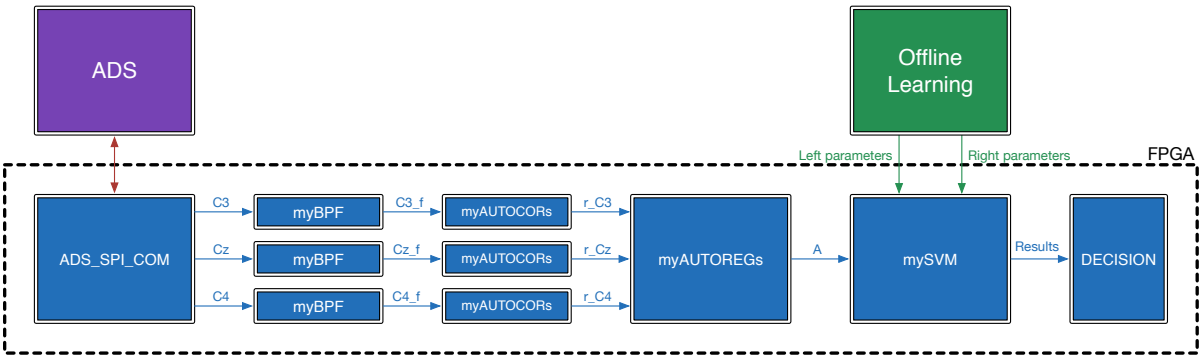


Figure 4.13: Diagram of the online classification

Several steps are followed. The data of the three channels `C3`, `Cz` and `C4` are first sent by the ADS and collected by the module `ADS_SPI_COM` on the FPGA, as explained in Section 4.1. These data are then filtered thanks to the module `myBPF`. The autocorrelation values of the filtered signals `C3_f`, `Cz_f` and `C4_f` are then computed on time intervals of 1 s, thanks to the module `myAUTOCORs`. Based on these autocorrelation values `r_C3`, `r_Cz` and `r_C4`, the AR coefficients of each signal are then computed, thanks to the module `myAUTOREGs`. The AR coefficients of the three channels are concatenated in a feature vector $\mathbf{A}$ (this notation corresponds to the generic notation $\mathbf{x}$ used previously). The module `mySVM` executes the SVM classifications on the input feature vector $\mathbf{A}$, thanks to the parameters computed during the learning phase (refer to Section 4.2). Based on the SVM results, a decision is made and the appropriate LED is lighted.

For the hardware implementation of the classification, a unique guideline is always followed: use as less LEs as possible. For example, serial implementations are preferred to parallel ones since main constraint is related to the maximum amount of logic available (in this work, 40k LEs). In comparison, the speed is not really a concern when working at such a high clock frequency (52 MHz in this work) and such a low data rate (250 Hz). Many clock cycles are available between two data samples. Thus, only one module `myAUTOREGs` is implemented and used 3 times in a row to compute the AR coefficients of the three channels. The same approach is used with the module `mySVM`, used 2 times in a row with different parameters to execute the

two SVM classifications for classifiers `SVM_Left` and `SVM_Right`.

Additions, subtractions, multiplications and divisions are used in the implementation. When complex arithmetic is performed on hardware, two binary representations are conceivable: the *floating point representation* or the *two's complement fixed-point representation.* For the same number of bits, the first representation is more accurate, but at the cost of heavier hardware modules. Basic arithmetic operations are thus slower. The second representation requires simpler hardware modules and is then faster to perform basic operations. However, care must be taken about the precision. To use less LEs and increase the speed of the developed modules, the fixed-point representation is chosen in this work.

The implementation of the online classification made in hardware is explained in this section. The implementation details related to features generation are first introduced. This includes the computation of the autocorrelation values r_i's and the determination of the AR coefficients a_i's. Afterwards, the hardware implementation of the SVM classification and of its kernel is explained.

4.3.1 Features generation

As explained in Chapter 2, the SVM algorithm classifies an input feature vector based on several learned parameters. The determination of the SVM parameters is explained in Section 4.2. The hardware features generation is presented here. It is divided into two parts: first, the autocorrelation values r_i's of equation (4.3) are computed on a given time interval duration (1 s in practice, or equivalently 250 samples); secondly, the AR coefficients a_i's of each signal are computed, with a hardware implementation of the Levinson-Durbin algorithm, which is given in Listing 4.1.

4.3.1.1 Computation of autocorrelation values

Hardware implementation As a reminder, the autocorrelation values of a sequence of length N are estimated as follows:

$$r_k = \sum_{i=k}^{N} s[n]\ s[n-k] = \sum_{i=0}^{N-k} s[n]\ s[n+k]. \tag{4.5}$$

Figure 4.14 illustrates the hardware implementation of the computation of the $p+1$ autocorrelation coefficients r_0, r_1, ..., r_p of (4.5). A FSM is used to synchronise the operations. As a reminder, p is fixed to 6 in this work. This requires to keep constantly in memory the $p = 6$ last values of the data, in order to be able to compute r_6. This is done with registers. In the implementation, there are also 7 accumulators to compute iteratively the 7 needed autocorrelations values.

The FSM begins with the `Rst` state, where all registers are cleared, together with the 7 accumulators and a counter i. Then, the FSM goes to the `Wait` state. A `start` signal is sent by the module `myBPF` as soon as a new data is ready. The FSM can then move from the `Wait` state to the `Shift` state. In this state, the 6 last values of the input data, saved in registers, are updated with the arrival of the new data: the new data $s[n]$ is introduced into the register bank, while previous values $s[n-j]$, for $j = 1, 2, ..., p$, are shifted by one position in the register bank.

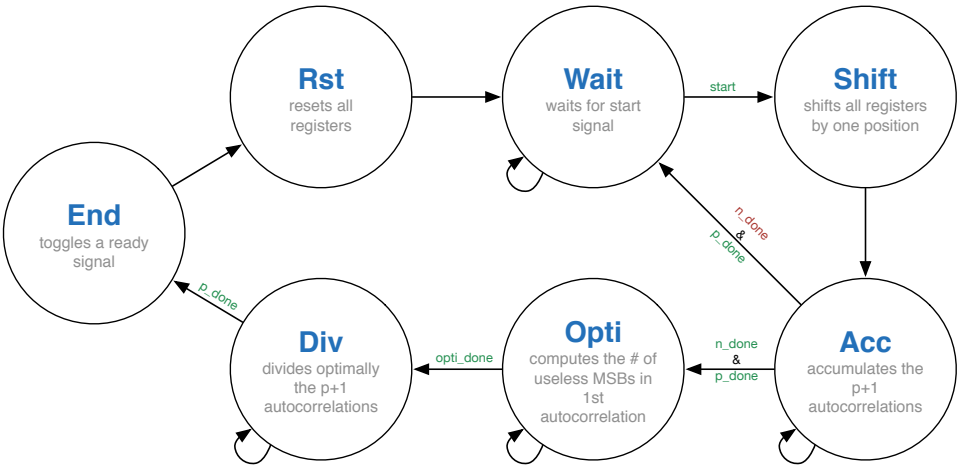


Figure 4.14: FSM of the module `myAUTOCORs`

The next state is named `Acc` and is responsible for the computation of the sum in equation (4.5). It computes the products $s[i]\cdot s[i+k]$, for $k = 0, 1, ..., p$ and adds them to the 7 accumulator values. The i counter value is then incremented.

States `Wait`, `Shift` and `Acc` are repeated N times to obtain the autocorrelation values r_k's. The FSM can then go to the `Opti` state. At this step, the autocorrelation coefficients are available but can be quite big, especially if a high N value is chosen. In practice, the r_k's are computed in two's complement representation on 57 bits to avoid overflow. The purpose of the `Opti` state and the `Div` state is to optimally represent the r_k's with only 32 bits, losing as less precision as possible. It is important to reduce the size of the autocorrelation values since they are used in the next module, `myAUTOREGs`.

To do this, the `Opti` state counts the number n of following 0's located in the MSBs of r_0. For example, if r_0 was equal to 6 (`"00000110"`), represented with 8 bits in two's complement representation, the `Opti` state would count $n = 5$ which means that the 4 first 0's can be removed without loosing any information on the value (not the 5 first 0's since the two's complement representation is used and the sign bit can not be forgotten). In this simple example, the r_0 might then be represented with full precision by only 4 bits (`"0110"`).

Based on this number n computed by the `Opti` state, the `Div` state removes the $n-1$ MSBs of all the r_k's. No precision is lost during this first operation, thanks to the property of the autocorrelation coefficients given by (4.4). The 32 first MSBs of the results are then kept to represent the final autocorrelation values. This last operation corresponds to a division of all the r's by the same factor. Only LSBs are eventually lost during this last operation.

Once this is done, the FSM finally goes to the `End` state where it toggles a `ready` signal before clearing all its registers in the `Rst` state. The FSM waits then again in the `Wait` state to receive a new `start` signal to restart all the operation.

As a remark, it is not very interesting to serialize this module to process the three chosen channels. Indeed, if only one module is used, the N last values of each channel (instead of p) have to be kept in memory (in registers for example) before being processed by the module. The saving of hardware space is then not big compared to a parallel implementation. For this reason, one module per channel is used, to save logic space as well as computation time.

Hardware validation One of the many tests performed to validate the module `myAUTOCORs` is presented here.

As a reminder, the ADS data rate is fixed to 250 Hz, which means that 250 new data are available each second. In practice, the N value is fixed to 250 to allow to compute the autocorrelation values on time interval of 1 s, as explained in Section 4.2. However, for readability, the N is fixed to 7 for this test.

An input signal is first created on MATLAB and then exported to be usable on MODELSIM and on the FPGA. The corresponding autocorrelation values are computed on MATLAB and taken as references. These are shown in Figure 4.15.

```
Input signal:
[-593  -690 -730 -676 -570 -495 -497]
Autocorrelation coefficients:
[2634559  2319835 1933340 1491490 1042370 636465 294721]
```

Figure 4.15: Computation of autocorrelation coefficients using MATLAB

The same signal is used with the module `myAUTOCORs` on MODELSIM. The results correspond perfectly with the references computed on MATLAB, as shown in Figure 4.21.

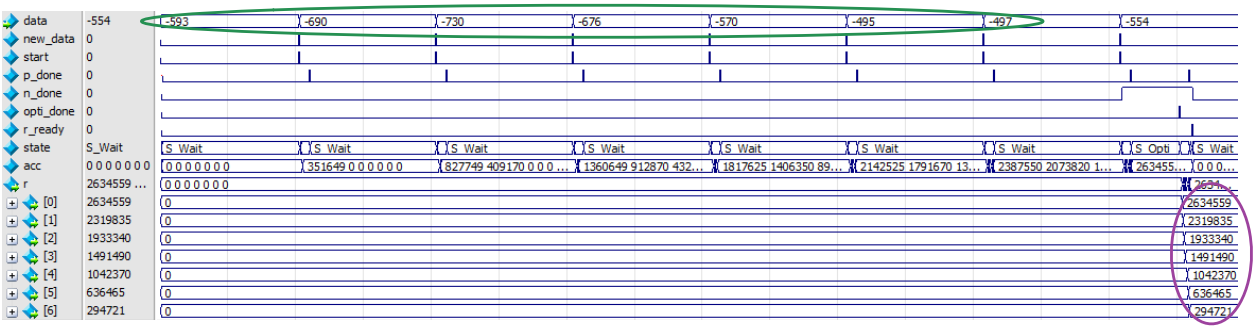


Figure 4.16: Simulation of the module `myAUTOCORs` on MODELSIM

The same test is performed directly on the FPGA. Again, the results fit perfectly with the expectations, as shows Figure 4.22.

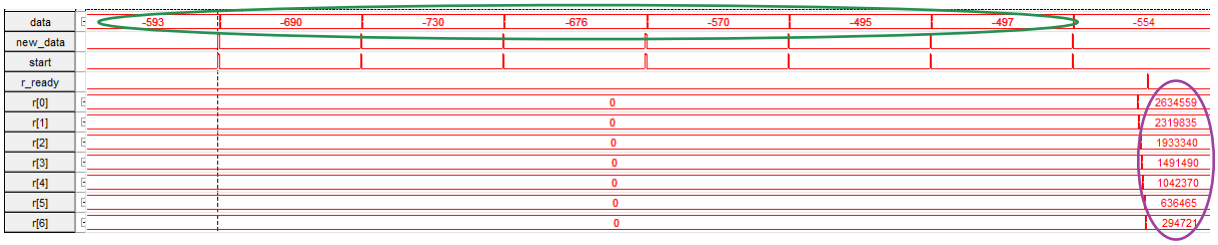
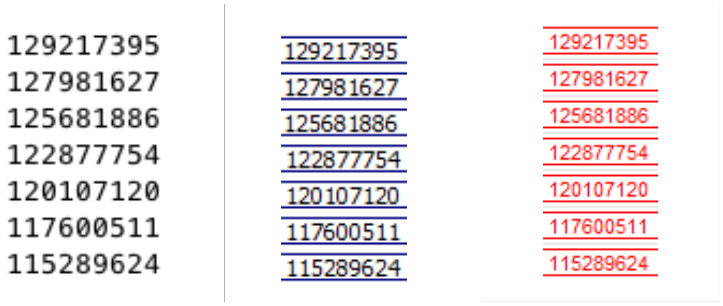


Figure 4.17: Simulation of the module `myAUTOCORs` on SIGNALTAP

Of course, tests with N equal to 250 are also performed. The results are shown in Figure 4.18 and are used later on. Again, the results on MODELSIM and on the FPGA correspond perfectly with the MATLAB references.



(a) MATLAB (b) MODELSIM (c) SIGNALTAP

Figure 4.18: Simulations results of the module `myAUTOCORs` with $N = 250$

4.3.1.2 Computation of AR coefficients

Hardware implementation As a reminder, the Levinson-Durbin algorithm is shown in Listing 4.1 and is recalled here.

```
// Initializations
a_0 = 1;
E_0 = r_0;

for i = 1 to p
    // Reflection coefficients
    k_i = 0;
    for j = 0 to i - 1                    // A.
        k_i = k_i - a_j * r_{i-j};
    end
    k_i = k_i / E_{i-1};                  // B.

    // AR values
    c_i = k_i;
    for j = 1 to i - 1                    // C.
        c_j = a_j + k_i * a_{i-j};
    end

    // Error
    E_i = (1 - k_i^2) * E_{i-1};          // D.

    // Update AR values
    for j = 1 to p                        // E.
        a_j = c_j
    end
end
```

Listing 4.2: Levinson-Durbin algorithm

The Levinson-Durbin algorithm is implemented in the module `myAUTOREGs`. A FSM is used to synchronise the operations, once again. It is represented in Figure 4.19.

The module `myAUTOCORs` sends a `start` signal to the module `myAUTOREGs` when the auto-correlation values r_k's are ready. The FSM goes then from the `Wait` state to the `K_acc` state.

In this state, the k_i value is accumulated, as done in the first inner loop of Listing 4.2 (step A.). Then, the FSM moves to the `K_div` state where, the accumulation result is divided by E_{i-1}. This state is particularly sensible because a division of 64 bits is performed. Since complex hardware division can not be computed in only one clock cycle, a delay of 8 cycles is

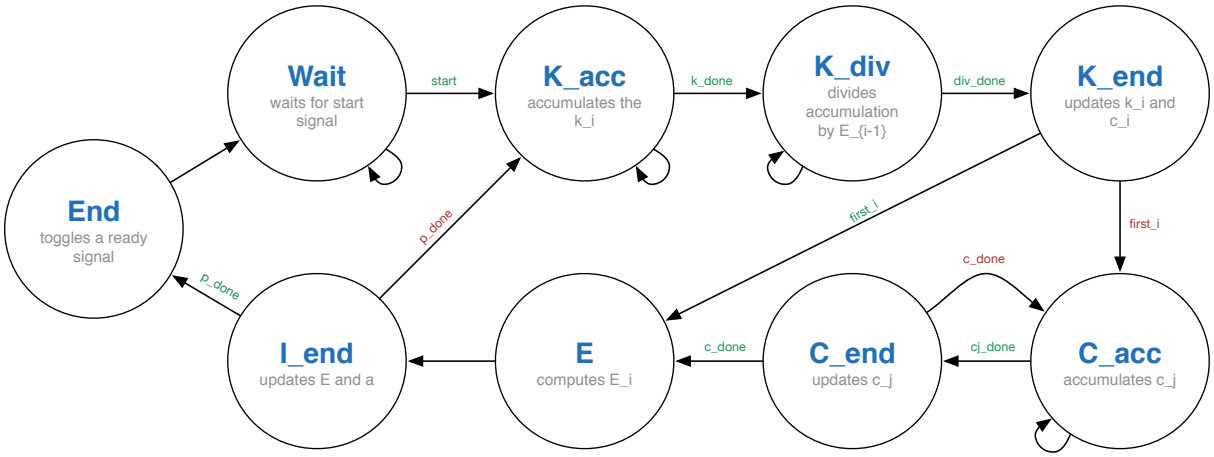


Figure 4.19: FSM of the module `myAUTOREGs`

introduced in this state. The result of the division is finally saved in k_i and c_i in the `K_end` state (step B.).

If i is strictly bigger than 1, the second inner loop of Listing 4.2 is executed in the `C_acc` and `C_end` states before the FSM goes to the `E` state (step C.). Otherwise, the `E` state is directly reached. This state is responsible for the computation of the value E_i (step D.). The `I_end` state is the last step of the main loop. The E_i value is saved, the a_i's are updated, and the i counter is incremented (step E.).

These 7 states are repeated until i equals $p+1$. At this moment, the computation of the AR coefficients a_i's is done. The FSM goes then to the `End` state where it toggles a `ready` signal, before reaching the `Wait` state where it waits again for a `start` signal to restart the operation.

As explained earlier, only one module is used 3 times in a row with 3 sets of autocorrelation values r_k's to compute the AR coefficients of the 3 channels.

Hardware validation Again, a test using MATLAB, MODELSIM and SIGNALTAP is proposed here. The autocorrelation coefficients computed with $N = 250$ in the previous subsection are used in this test to compute the corresponding AR coefficients. As a reminder, p is fixed to 6.

A MATLAB function is first used to compute the AR coefficients. These values, shown in Figure 4.20, are taken as references.

```
Autocorrelation Coefficients:
[129217395  127981627 125681886 122877754 120107120 117600511 115289624]

Corresponding Autoregressive Coefficients:
[1.000000  -1.406582 0.324585 0.192776 -0.020191 -0.111521 0.032987]
```

Figure 4.20: Computation of AR coefficients using MATLAB

The module `myAUTOREGs` is then tested with MODELSIM, using the same autocorrelation coefficients r_k's. The results correspond perfectly with the references computed on MATLAB, as shown in Figure 4.21.

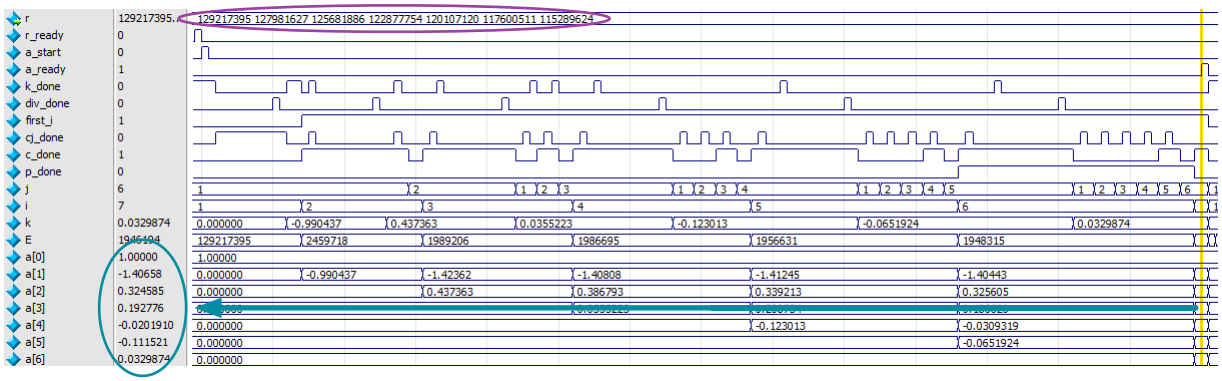


Figure 4.21: Simulation of the module `myAUTOREGs` on ModelSim

Finally, the module is tested on the FPGA and the results are checked thanks to SignalTap. The AR coefficients a_i's are displayed in hexadecimal because it is not possible to convert them directly in SignalTap. They are represented in two's complement fixed point on 32 bits, with 28 fraction bits. A Matlab function is used to convert them in decimal representation. The results are: $a_0 = 1$; $a_1 = -1.40658$; $a_2 = 0.324585$; $a_3 = 0.192776$; $a_4 = -0.0201910$; $a_5 = -0.111521$; $a_6 = 0.0329874$. They correspond perfectly with the results obtained on ModelSim, and thus with the ones obtained on Matlab.

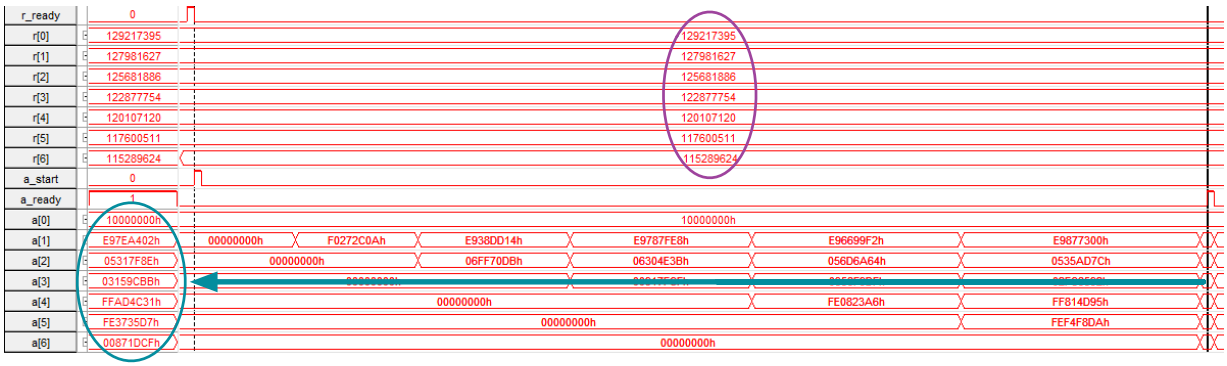


Figure 4.22: Simulation of the module `myAUTOREGs` on SignalTap

4.3.2 SVM classification

The SVM classification step is the last part of the data processing flow implemented in hardware. The corresponding classification function f of equation (2.7), introduced in Chapter 2, can be rewritten as follows, to be implemented on FPGA:

$$y = f(\mathbf{x}) = \text{sign}\left(\underbrace{\sum_{i=1}^{N_{SV}} y_i^* \alpha_i \; K\left(\mathbf{x}_i, \mathbf{A}\right) + b}_{y_c}\right). \tag{4.6}$$

In this expression, only the Support Vectors (SVs) are taken into account. SVs are vectors $\mathbf{x}_i$'s for which corresponding Lagrange multipliers α_i's are non zero. The number of SVs is denoted by N_{SV}. The parameters α_i's, the SVs $\mathbf{x}_i$'s, the corresponding true labels y_i^*'s and the offset b are all known thanks to the learning phase, explained in Section 4.2. The input feature vector, previously denoted by $\mathbf{x}$, is written $\mathbf{A}$. The kernel is written $K\left(\mathbf{x}_i, \mathbf{A}\right)$, and is linear for this application.

As a remark, the implementation on FPGA computes the argument of the sign function, written as y_c. This value informs about the confidence of the classification: when y_c has a big positive value, the output label corresponds to the positive class and the classifier is very confident about this decision; when y_c has a very negative value, the output label corresponds to the negative class and the classifier is very confident about this decision, once again; for small absolute values of y_c, the classifier is less confident about its decision.

In practice, as mentioned before, two classifications are performed: the first one distinguishes the *Left* class against the two other classes (that corresponds to classifier `SVM_Left`); the other one distinguishes the *Right* against the two other classes (that corresponds to classifier `SVM_Right`). The results of these two classifications are then combined to take the final decision. Needed parameters and SVs for each classifier are determined thanks to the learning phase. Both classifiers classify the same input feature vector $\mathbf{A}$. As a reminder, this input feature vector $\mathbf{A}$ is the concatenation of 6 AR coefficients computed for each channel, on time intervals of 1 s. The length of $\mathbf{A}$ is therefore 18.

Hardware implementation The value y_c in (4.6) is computed thanks to the module `mySVM`. A FSM is used to control the operations, as represented in Figure 4.23.

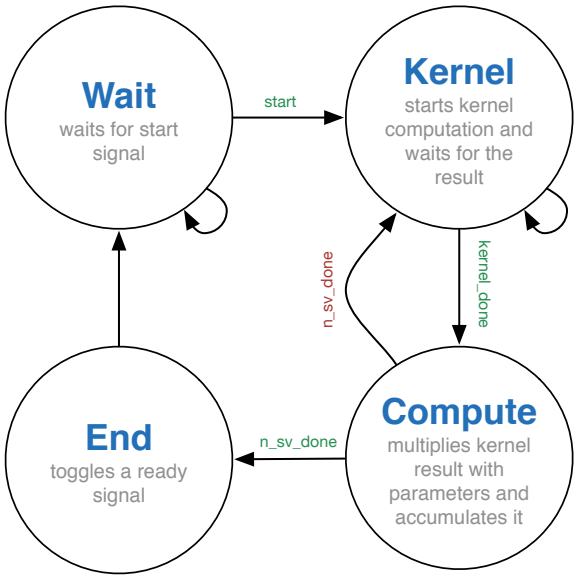


Figure 4.23: FSM of the module `mySVM`

The FSM first waits in the `Wait` state, in which an accumulator `acc` is initialized to the offset value b and a counter value i is initialized to 1. The accumulator is used to compute iteratively the sum of equation (4.6). When the FSM receives a `start` signal, it moves to the `Kernel` state. There, the kernel computation of the index i is started.

The kernel used in this work is the linear one, which is a simple scalar product between the i^{th} SV $\mathbf{x}_i$ and the input feature vector $\mathbf{A}$. This is done by a simple submodule called `myKernel`. Using an external submodule for the computation of the kernel gives flexibility to the implementation: when moving to a more complex kernel (exponential, polynomial or hardware friendly), the only needed change to make is to modify this submodule, not the entire FSM.

Once the kernel result is ready, the FSM goes to the `Compute` state. In this state, the FSM multiplies the kernel result by the i^{th} value of parameters α_i's and y_i's, accumulates the multiplication result in `acc` and increments the i counter.

States `Kernel` and `Compute` are repeated N_{SV} times to compute the entire sum of equation (4.6). When the final result is available, the FSM goes to the `End` state. The FSM then toggles a ready signal before going back to the `Wait` state, where it waits again for a `start` signal to restart the operations.

In practice, to compute the two classification results for classifiers `SVM_Left` and `SVM_Right`, the module `mySVM` is used two times in a row with the adequate parameters. The same input feature vector $\mathbf{A}$ is used. Two results are thus computed, y_{cL} and y_{cR}, which correspond to the output of classifiers `SVM_Left` and `SVM_Right` respectively. To take the final decision about the class label that corresponds to the input vector $\mathbf{A}$, these two results are compared to each other: if they are both negative, the *Rest* class is assigned to $\mathbf{A}$; otherwise, the class which has the biggest output result wins.

Hardware validation A classification test is realised to validate the module `mySVM`. Three signals and random classification parameters are generated and exported thanks to MATLAB. The AR coefficients of each signals are then computed to get the feature vector `A`. The SVM classification is finally performed.

The test is first realised on MATLAB. The expected SVM result is: $y_c = -4.287762$.

Figure 4.24 shows the corresponding results obtained on MODELSIM. The obtained SVM result is $y_c = -4.28271$. The little difference compared to the expected value of `Matlab` has two explanations. First, the parameters used in `Matlab` are represented in full precision, while the parameters used in ModelSim are represented on 32 signed bits (28 fractions bits for the support vectors and 11 fraction bits for the other parameters). In addition, the final SVM result computed in the module `mySVM` is compressed from 99 bits with 67 fractions bits to 32 bits with 11 fractions bits, to easily use it outside the module. However, the difference between the MATLAB result and the MODELSIM is very small and is then not an issue.

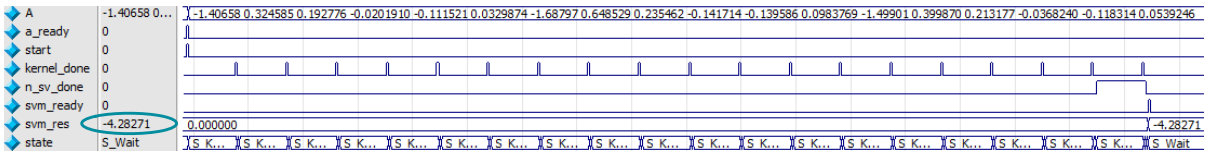


Figure 4.24: Simulation of the module `mySVM` on MODELSIM

The same test is also performed on the FPGA, as shown in Figure 4.25. The obtained SVM result is converted from hexadecimal to decimal and is equal to $y_c = -4.28271$, the exact same value obtained with MODELSIM.

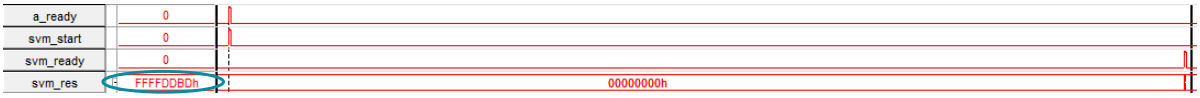


Figure 4.25: Simulation of the module `mySVM` on SIGNALTAP

4.4 Conclusion

The purpose of this chapter is to demonstrate that the experimentation system, designed in Chapter 3 thanks to the background given in Chapter 2, can be used to develop and test BMIs. A basic Motor Imagery application is imagined and its implementation is described in details in this chapter. The aim of the application is to turn on the leftmost/rightmost LED of the experimentation system if the user thinks about moving his left/right arm, and turn all the LEDs off otherwise. Two SVM classifiers, which need an offline training, are used in the application.

The first block of the application is responsible for the extraction of EEG signals produced by the user and their transmission via SPI from the ADSs to the FPGA. This is done by the module `ADS_SPI_COM`. A BPF is also used, in order to attenuate the annoying offset and the 50 Hz noise present in acquired EEG data.

The second step is related to the learning of the needed SVM parameters. This is done on MATLAB. The training of the classification algorithm is done thanks to many EEG training samples. The features used for the classification algorithm are AR coefficients, which require the computation of the autocorrelation values of the EEG signals on time intervals of fixed duration. MATLAB tools are then used for the learning of two SVM classifiers. The procedure to find optimal parameters and corresponding SVs is explained.

The last part describes the online classification, and the corresponding hardware modules implemented on FPGA. AR coefficients, used as features, are computed in real-time thanks to modules `myAUTOCORs` and `myAUTOREGs`. The SVM classifications are then performed by the module `mySVM`, using the parameters computed during the learning step. Based on the classification results, a control signal is generated to turn on or off the adequate LEDs.

Next chapter is about the results obtained with the experimentation system and the application described here.

Chapter 5

Results

While Chapter 3 presents the experimentation system design, Chapter 4 explains and validates all the modules implemented on the FPGA. This chapter is devoted to the analysis of the final system results.

First, characterisations of the ADS1299 and of the implementations made on the FPGA are done. The classification results are then described.

5.1 Characterisation

The ADS1299 is characterised in this section. Its input-referred noise is determined, its supply voltages as well as its internal clock frequency are checked and the bias feature is analysed. A typical EEG acquisition made by the experimentation system is also proposed. Afterwards, the whole implementation characteristics are studied in details. Implemented modules are compared based on the amount of logic they need and based on their response time.

5.1.1 ADS

5.1.1.1 Input-referred noise

The ADS is supposed to be very low noise to allow an accurate measurement of EEG signals, which have an amplitude between 10 and 100 μV. As explained in Chapter 3, a multiplexer is available on each input channel of the ADS. By configuring these multiplexers, it is possible to shortcut the amplifier inputs of each channel and then to measure the input-referred noise of the ADS.

An acquisition is thus executed with this multiplexer configuration. During this test, the ADS is configured with the same characteristics than during the SVM classification process, with a data rate of 250 Hz and a amplifier gain of 24. This configuration is supposed to give the best ADS performances in term of noise. The acquisition is performed during 10 s and Figure 5.3 presents the results for one channel. The raw signal represents the data sent by the ADS and the filtered signal is generated thanks to the filter presented in Chapter 4 (which has a bandwidth spreading from 7 Hz to 35 Hz).

Two values are generally used to characterise the noise in a system: its *root mean square* (RMS) amplitude and its *peak-to-peak* amplitude. For the raw signal, the RMS amplitude is equal to 28.805 μV_{RMS} and the *peak-to-peak* amplitude value is 1.632 μV_{pp}. The filter is useful to remove the quite significant RMS amplitude. Indeed, for the filtered signal, the RMS

amplitude is equal to 0.142 μV_{RMS} and the *peak-to-peak* amplitude to 1.006 μV_{pp}, which is exactly what is specified in [27]. These values are really good results and EEG signals are thus not really polluted by the experimentation system.

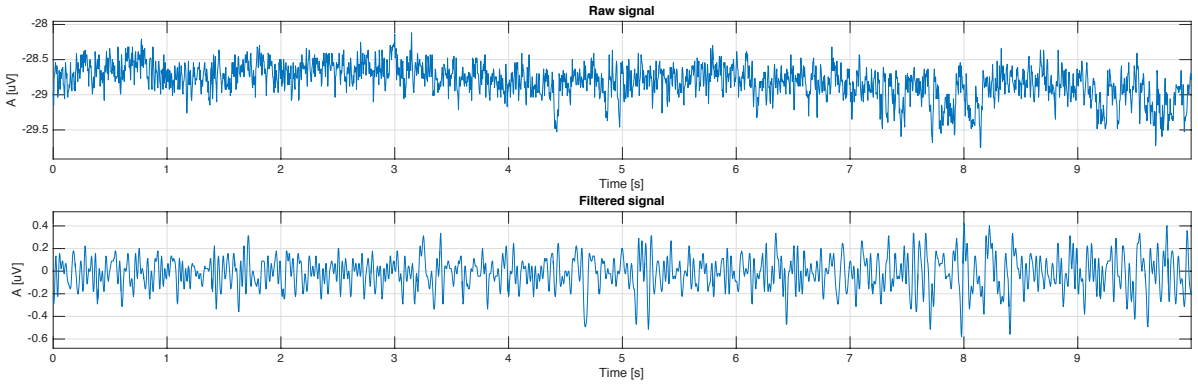


Figure 5.1: Input referred noise on the ADS

5.1.1.2 Supply voltages

The ADS needs 3 different supply voltages to work properly: *AVDD*, *AVSS* and *DVDD*. As explained in Chapter 3, these voltages are fixed respectively to 2.5 V, -2.5 V and 3.3 V. The input multiplexers can be configured in such a way that the supply voltages can be checked directly thanks to the ADS. It is thus possible to measure two different voltages: $v_{test1} = (AVDD - AVSS)/2$ (on channel 1, 2, 5, 6, 7 and 8) and $v_{test2} = DVDD/4$ (on channel 3 and 4). An acquisition of 3 s is performed with this configuration and the results of channel 1 and channel 3 are shown in Figure 5.2. For the first graph, the expected result of v_{test1} is equal to 2.5 V while the experimental result is equal to 2.498 V_{RMS} with 118.554 μV_{pp}. For the second graph, the expected result of v_{test2} is equal to 825 mV while the experimental result is equal to 822.746 with 111.58 μV_{pp}. The small differences between the expected values and the experimental ones show that the supply voltages of the ADS are as good as expected.

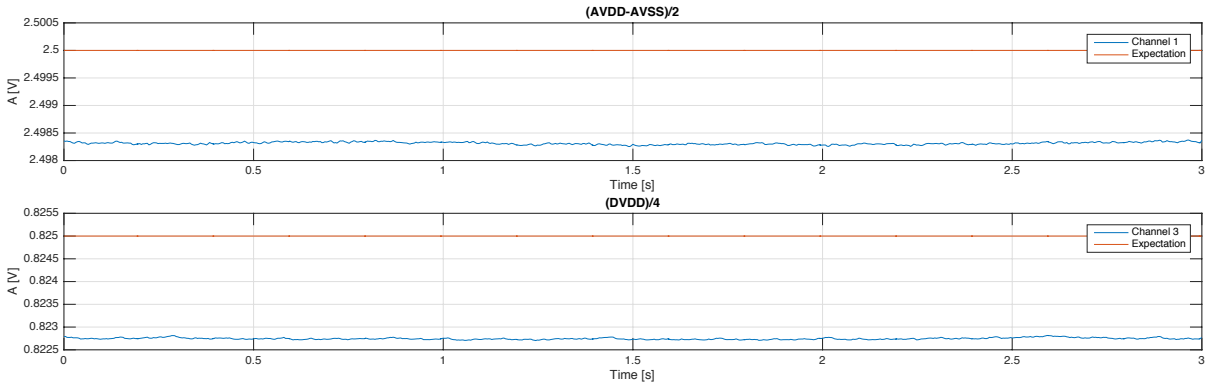


Figure 5.2: Supply test on the ADS

5.1.1.3 Internal oscillator frequency

A last kind of test signal can be generated by the ADS: a square wave with two possible frequencies. The signal frequency can be fixed to either $f_{test1} = f_{clk}/2^{21}$ Hz or $f_{test2} = f_{clk}/2^{20}$ Hz, which allows to check the internal clock frequency supposed to be equal to 2.048 MHz. Figure 5.3 shows the results. For the two signals, the computed experimental frequency corresponds

perfectly with the expected one, equal to respectively 0.9765 Hz and 1.9531 Hz. As the internal clock frequency is truly equal to 2.048 MHz, the selected data rate can be expected to be accurate too.

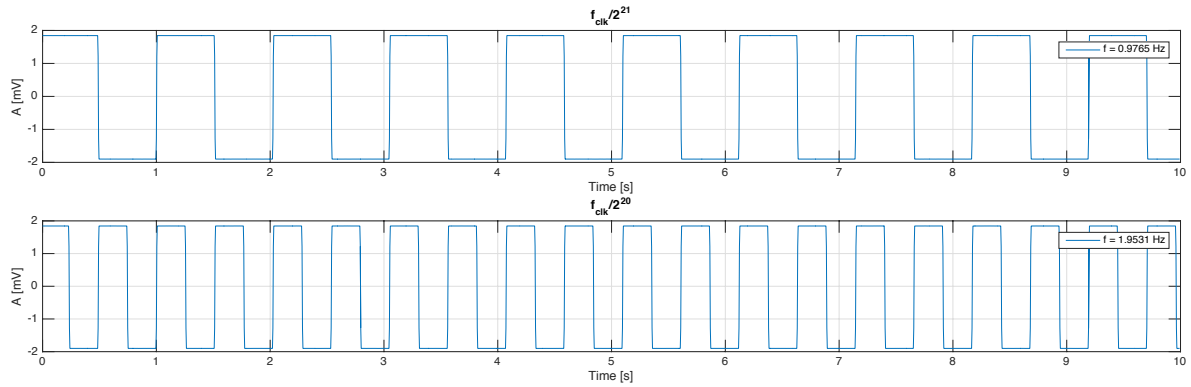


Figure 5.3: Clock frequency test on the ADS

5.1.1.4 Bias feature

The ADS has a really useful feature for EEG acquisitions. Indeed, as explained in Chapter 3, the bias feature allows to theoretically enhance the CMRR and lower the channels correlated noise (as the noise at 50 Hz for example). To check the performance of the bias feature, several tests are performed without the use of the BPF.

First, an EEG acquisition of the three channels C3, Cz and C4 is made during 5 s without bias generation. The result, presented in Figure 5.4 for one channel, shows that the 50 Hz noise amplitude is really significant (9.96 dB) compared to the one of the signal of interest (below -50 dB).

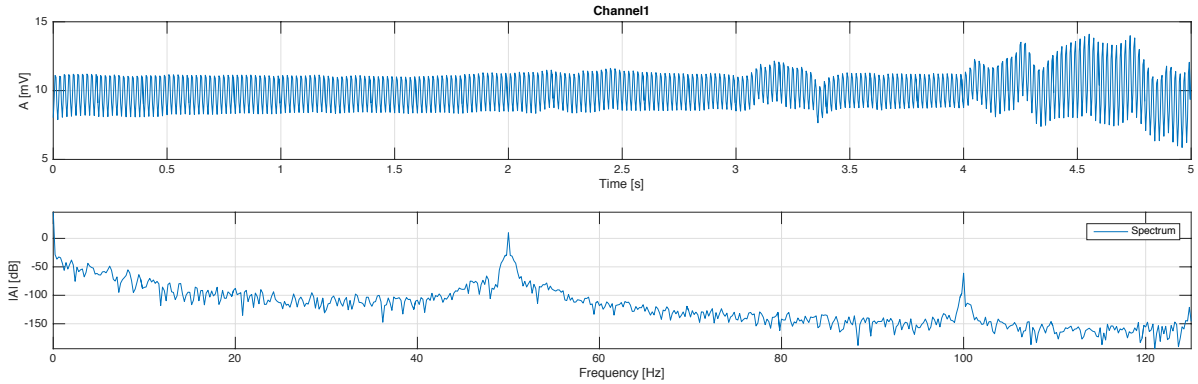
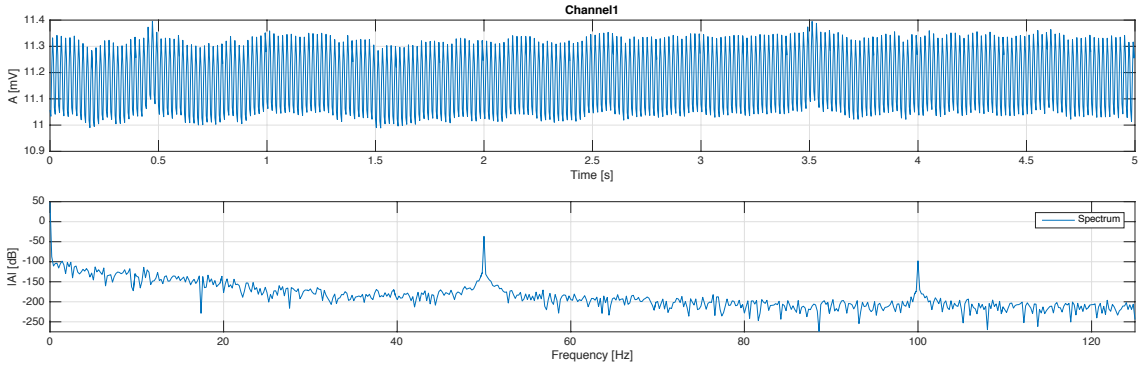
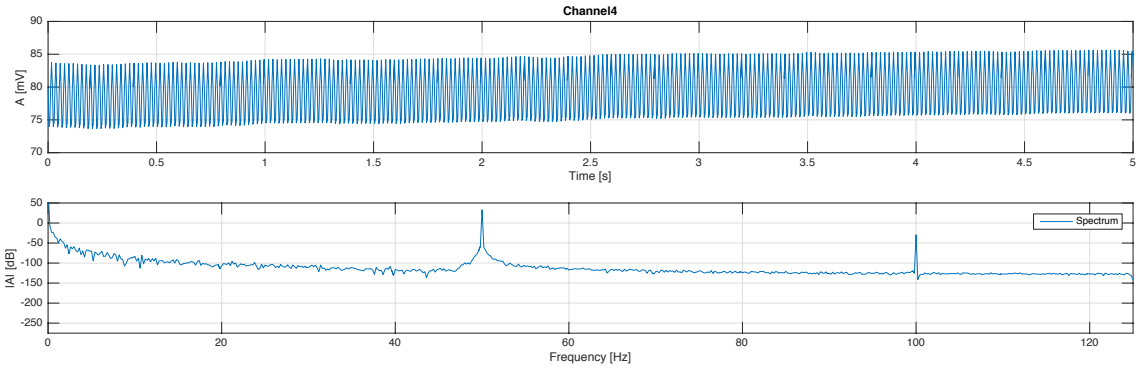


Figure 5.4: EEG acquisition without bias

After that, the bias circuit is enabled for the three channels. The generated bias signal, shown in Figure 5.5(b), leads well to a noise diminution on the three channels, as shown for C3 in Figure 5.5(a). The 50 Hz noise is well reduced to -36.58 dB but is still too significant compared to the amplitude of the signal of interest. Furthermore, an offset of more than 10 mV is still present.



(a) EEG signal with bias



(b) Corresponding bias signal

Figure 5.5: EEG acquisition with bias

5.1.1.5 Typical acquisition

As seen before, the bias feature enhances the acquisition results but not enough to properly acquire EEG signals. The 50 Hz noise and the offset are still too significant. That is why a BPF is used. A typical EEG acquisition is performed in the exact same conditions than for Figure 5.5, but the BPF is enabled this time. The result can be seen in Figure 5.6. After filtering, the 50 Hz noise amplitude (-180 dB) is well smaller than the amplitude of the signal in the band of interest.

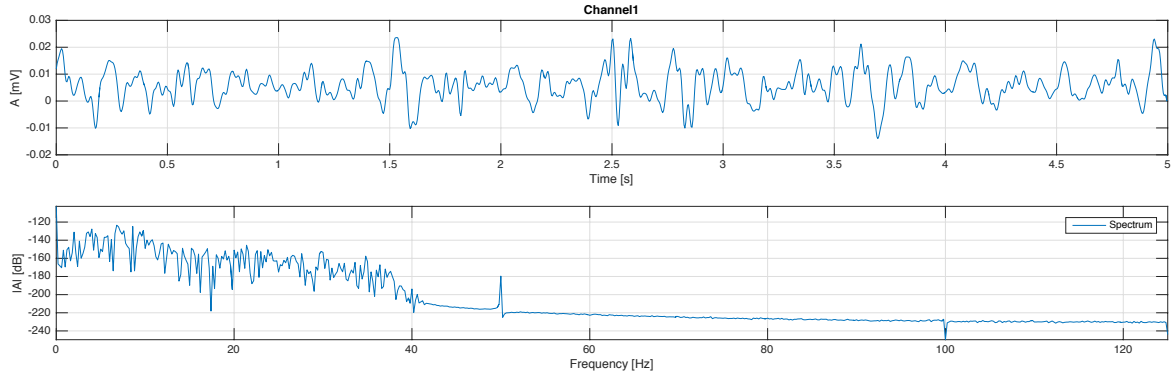


Figure 5.6: Typical acquisition with the experimentation system

At this stage, a conclusion can be made: the EEG acquisition is reliable and can be used for the classification step.

5.1.2 Hardware implementation

To characterise the modules implemented in hardware on FPGA, two criteria are used: the amount of logic they need and their response time.

5.1.2.1 Size repartition

Figure 5.7 represents the repartition of logic, in LEs, between the several hardware modules. These values are computed thanks to QUARTUS and can vary a little bit since the compiler optimises the size of the modules at each compilation and this optimisation is configurable. The total size of the hardware implementation is equal to 29.356 kLEs.

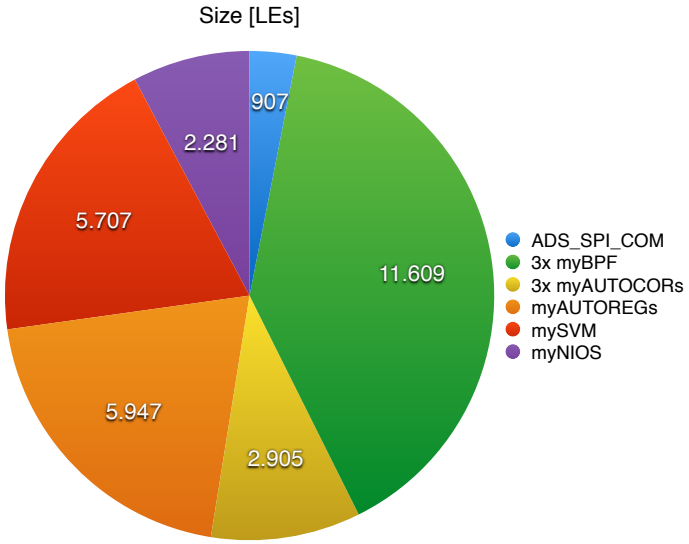


Figure 5.7: Size repartition of the FPGA implementation

The module `ADS_SPI_COM` is very small (907 LEs), which is a good result since it is the only module that has to always be implemented in the development board, to make the interface between the ADS and the FPGA. All the other modules can change in the future if another classification algorithm is chosen or if the application is improved.

As explained in Chapter 4, the only objective of the module `myNIOS` is to ease the interface between the user and the ADS, but the whole implementation is able to work perfectly without it. The amount of logic it takes (2.281 kLEs) can then be removed if needed.

The 3 filters `myBPF`, implemented thanks to MATLAB, take a lot of space in the implementation (11.609 kLEs) but are essential to allow a clean EEG acquisition. Less stringent specifications can be used to lower their size, if needed. These filters could also be implemented in software (on the `NIOS II` or on a MCU).

The module `myAUTOCORs`, which is responsible for the computation of the autocorrelation coefficients r_k's, is quite small (2.905 kLEs for the three channels), thanks to the serialisation of the multiplications. Only one multiplier is then needed per module. The module `myAUTOCORs` would have been probably far bigger if parallel multiplication had been chosen, since it would have required more multipliers.

As a reminder, a 64 bits division is performed in the module `myAUTOREGs`, thanks to a QUARTUS divider module. This single divider module already takes about 1.5 kLEs. The

rest of the implementation of `myAUTOREGs` is also quite big (4.447 kLEs), because many heavy arithmetic operations are performed. As explained in Chapter 4, the module `myAUTOREGs` computes the AR coefficients. It is used 3 times in a row, once per channel. This is done to lower the total size of the project. If needed, less precision can be used to lower the module size.

The size of the SVM classification process highly depends on the number N_{SV} of needed SVs, which are determined during the learning phase, and on the number of features N_x (here equal to 18, as 6 AR coefficients are computed for each of the three used channels). The module `mySVM`, which implements the SVM classification, requires an amount of logic equal to 5.707 kLEs. This size corresponds to about $N_{SV} = 40$ and to $N_x = 18$. As a reminder, two classifications are performed. To do that, the module `mySVM` is used twice in a row.

5.1.2.2 Response time repartition

As explained in Chapter 4, several delays exist in the implementation. First, the BPFs intrinsically have a time lag of 252 ms. Secondly, the AR coefficients are computed on an interval of time of 1 s. Therefore, the classification step is executed only once every second.

The response time of the whole hardware implementation is computed from the time at which the last data needed to compute the autocorrelation coefficients is available on the ADS, until the result of the second SVM classification is available. The total response time is equal to 83.46 μs and its repartition can be seen in Figure 5.8.

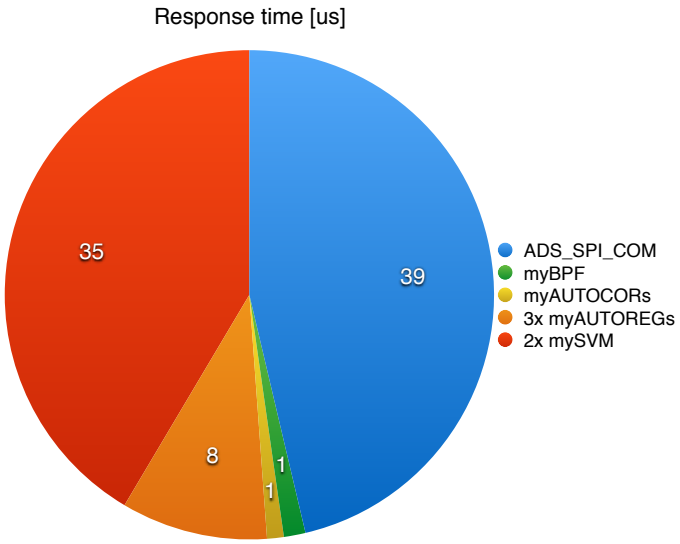


Figure 5.8: Response time repartition of the FPGA implementation

The two modules that take the most time are `ADS_SPI_COM` and `mySVM`. The reason for the first one (`ADS_SPI_COM`) is that it has to read in series the 216 bits sent by the ADS at the `SCLK` frequency (equal to 6.5 MHz). For the second module (`mySVM`), the time taken to compute the SVM results depends again on the number N_{SV} of needed SVs and on the number of features N_x. Two classifications are done in a row with only one module `mySVM` to lower the size of the design. But it is totally possible to perform the classifications in parallel, to lower the response time, if needed.

5.2 Classification Results

This section gives the performances achieved by the two classifiers `SVM_Left` and `SVM_Right` that are trained for this work. The performances are evaluated by computing the *accuracy* of the classifier, which corresponds to the number of correct classifications divided by the total number of classified samples. This metric can not be used alone: the type of misclassification errors matters as well. As an example, if the accuracy of a binary classifier on a given test set is of 90% but everything is classified in one class only, this is not good.

For this reason, the *confusion matrix* is also usually computed. For a binary classifier, it is defined as follows:

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, \tag{5.1}$$

where C_{ij} is the number of trials classified in class i but that belong in reality to class j. Ideally, for a perfect classifier, the confusion matrix is diagonal. As a consequence, the accuracy can also be defined like this:

$$\text{accuracy} = \frac{\#\text{ correct classifications}}{\#\text{ classifications}} = \frac{C_{11} + C_{22}}{C_{11} + C_{12} + C_{21} + C_{22}}. \tag{5.2}$$

This section is structured in the following way. First, the individual theoretical performances of classifiers `SVM_Left` and `SVM_Right` are evaluated. Then, the theoretical performances of the classifiers taken together are given. Finally, results obtained with the online classification are described.

5.2.1 Performances of each classifier taken alone

In this part, results are given for each classifier taken individually. As a reminder, the binary `SVM_Right` classifier aims to distinguish the *Right* class (class 1) from the two other classes (*Left* and *Rest*, which form together the class 2). In a similar manner, the `SVM_Left` classifier aims to distinguish the *Left* class (class 1) from the two other ones (*Right* and *Rest*, which form together the class 2).

The performances of classifier `SVM_Right` are given in Table 5.1 and are evaluated on three sets: the training set `Train_R` used to learn the classifier parameters; the validation set `Val_R` used to choose the C value; and the `Test` set, which is common to both classifiers. These sets are defined in Chapter 4. The parameter C has a value of 60000 and the classifier uses 39 SVs.

Classifier `SVM_Right`				
Sets	**Size**	**Errors**	**Accuracy**	**Confusion matrix**
`Train_R`	53	14	73.58%	$\begin{bmatrix} 13 & 5 \\ 9 & 26 \end{bmatrix}$
`Val_R`	60	19	68.33%	$\begin{bmatrix} 9 & 11 \\ 8 & 32 \end{bmatrix}$
`Test`	360	119	66.94%	$\begin{bmatrix} 58 & 58 \\ 61 & 183 \end{bmatrix}$
Total	473	152	67.87%	$\begin{bmatrix} 80 & 74 \\ 78 & 241 \end{bmatrix}$

Table 5.1: Performances of classifier `SVM_Right`

Table 5.2 shows the corresponding performances for the classifier `SVM_Left`. They are also evaluated on three sets: the training set `Train_L` used to learn the classifier parameters; the validation set `Val_L` used to choose the C value; and the `Test` set, which is common to both

classifiers. These sets are defined in Chapter 4. For this classifier, the parameter C is equal to 8000 and 19 SVs are used.

Classifier `SVM_Left`				
Sets	**Size**	**Errors**	**Accuracy**	**Confusion matrix**
`Train_L`	55	2	96.36%	$\begin{bmatrix} 18 & 2 \\ 0 & 35 \end{bmatrix}$
`Val_L`	60	14	76.67%	$\begin{bmatrix} 12 & 9 \\ 5 & 34 \end{bmatrix}$
`Test`	360	108	70.00%	$\begin{bmatrix} 67 & 52 \\ 56 & 185 \end{bmatrix}$
Total	475	124	73.89%	$\begin{bmatrix} 97 & 63 \\ 61 & 254 \end{bmatrix}$

Table 5.2: Performances of classifier `SVM_Left`

As can be observed, the performances on the training sets are always better than on other sets. This is normal, because the training sets are used to learn the parameters. The classifier tries to fit to these data. Validation and test sets can be viewed as new data from the classifiers point of view, and they are used to evaluate the generalization capabilities of the classifiers. As the performances do not drop too much when using the validation or test sets, overfitting is normally avoided and generalization capabilities can be assumed to be good.

5.2.2 Performances of the classifiers taken together

What matters in this work, is the combination of the two classifiers and the performances they obtain together. Table 5.3 shows these performances, evaluated on two sets: the `Test` set introduced in Chapter 4 is first used, which is obtained by cutting acquired EEG data into subintervals of 2 s before computing AR coefficients on each of these intervals; an additional set, called `Final`, is also used and is obtained by cutting acquired EEG data into subintervals of 1 s before computing the AR coefficients. The `Final` set is built to emulate what is really done on FPGA and estimate the performances that could be obtained in real-time. The *Right* class corresponds to class 1 in the confusion matrices, the *Left* class corresponds to class 2 and the *Rest* class is class 3.

Combination of classifiers				
Sets	**Size**	**Errors**	**Accuracy**	**Confusion matrix**
`Test`	360	158	56.11%	$\begin{bmatrix} 46 & 39 & 34 \\ 31 & 63 & 29 \\ 18 & 7 & 93 \end{bmatrix}$
`Final`	1080	476	55.93%	$\begin{bmatrix} 144 & 93 & 123 \\ 107 & 162 & 91 \\ 48 & 14 & 298 \end{bmatrix}$

Table 5.3: Performances of both classifiers taken together

All errors committed are not of the same nature. Trials that belong to the *Rest* class (class 3) and that are assigned to either *Right* (class 1) or *Left* (class 2) are the worst errors for this application. They correspond to entries C_{31} and C_{32} in the confusion matrices. They imply that even when performing no movement at all, LEDs can turn on. The results shown in Table 5.3 are quite good with this respect, as the accuracy for class 3 on the `Final` set is equal to

$$\text{accuracy for Class 3} = \frac{C_{33}}{C_{31} + C_{32} + C_{33}} = 82.78\%. \tag{5.3}$$

Trials that come from the *Right* class (class 1) and are assigned to *Left* (class 2), or conversely,

are also annoying errors. Instead of lighting the leftmost LED, the system lights the rightmost, or conversely. This kind of errors corresponds to entries C_{12} and C_{21} in the confusion matrices.

The last type of errors is the least worst. It is related to trials that belong to either the *Right* class (class 1) or the *Left* class (class 2) and which are classified into the *Rest* class (class 3). This situation corresponds respectively to the entries C_{13} and C_{23} in the confusion matrices. When a movement of the right or left arm is performed, no LED is turned on in response. This is less annoying, the user has just to retry. With these errors, the accuracies for classes 1 and 2 on the `Final`, computed as follows, are poor:

$$\text{accuracy for Class 1} = \frac{C_{11}}{C_{11}+C_{12}+C_{13}} = 40.00\%; \tag{5.4}$$

$$\text{accuracy for Class 2} = \frac{C_{22}}{C_{21}+C_{22}+C_{23}} = 45.00\%; \tag{5.5}$$

Without taking into account these less annoying errors in the computation of the accuracies of class 1 and class 2, this gives much better results on the `Final` set, which correspond more to what is perceived in practice when using the application in real-time:

$$\text{modified accuracy for Class 1} = \frac{C_{11}}{C_{11}+C_{12}} = 60.76\%; \tag{5.6}$$

$$\text{modified accuracy for Class 2} = \frac{C_{22}}{C_{21}+C_{22}} = 60.22\%; \tag{5.7}$$

Clearly, the results obtained are non extraordinary, without being too bad. The main objective of the implemented application is not to have the highest performances ever. It is rather linked to the fact that the experimentation system is functional and can be used to develop BMIs, which is the primary goal.

Moreover, the low accuracy values can have a lot of explanations and can depend on a high number of choices that are made. As an example, EEG signals can be corrupted by artefacts due to muscle tension or eye movements. These artefacts can confuse the classifier, either at the learning or the classification phases. Another explanation can also be related to the choice of the kernel. A linear kernel is probably not the best choice to classify EEG-based features.

5.2.3 Examples of online classification

This last part gives examples of online classification. Good classifiers parameters and corresponding SVs, determined during the learning phase (refer to Chapter 4), are introduced in the FPGA in order to classify in real time the user's thoughts. As a remark, the application is tested on the authors themselves.

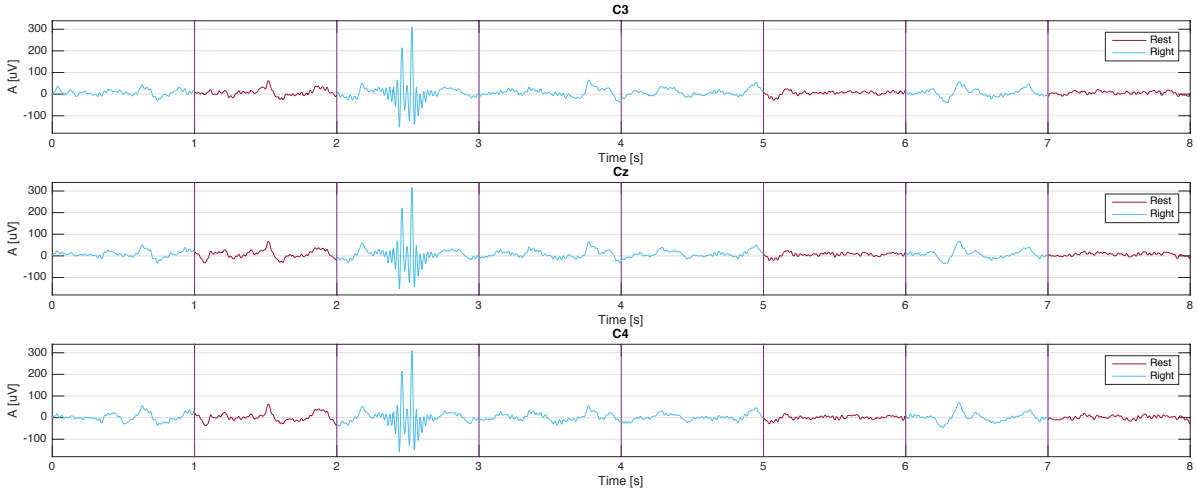
For the online classification, 5 electrodes are used: 3 are used to probe EEG signals on positions C3, Cz and C4; one electrode is used as a reference, located on a ear of the user; the last electrode is used as a bias, and is located on the other ear of the user. The sample rate of the ADS is fixed to 250 Hz and the autocorrelation values are computed every second. As a consequence, a classification result is also available every second. An online example for each class is presented here below in Figure 5.9 .

First, the user is asked to move continuously his right arm, that is to say, produce brain signals that belong to the *Right* class. The corresponding EEG signals as well as the corresponding classification decision made by the FPGA are shown in Figure 5.9(a). Secondly,

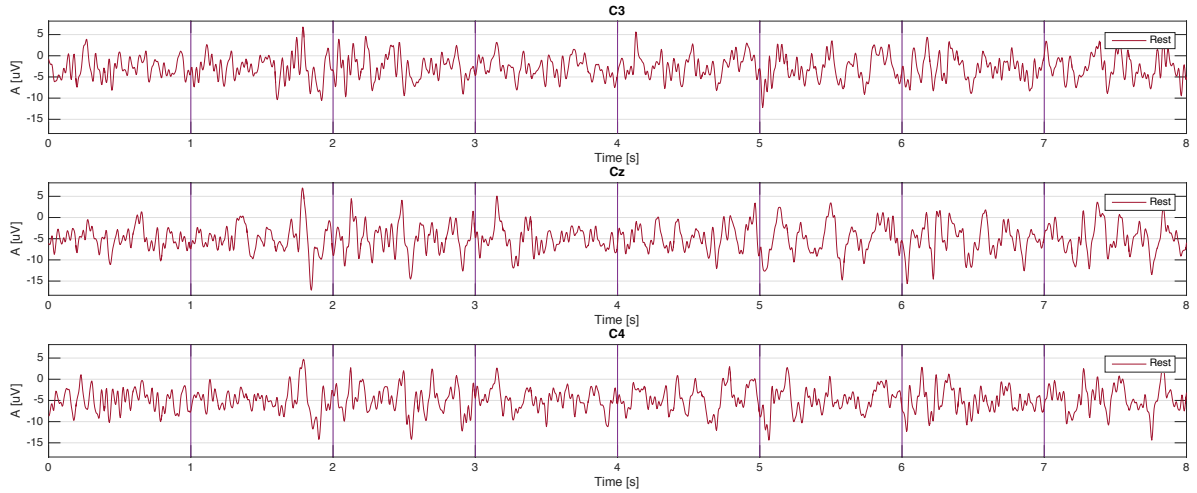
Figure 5.9(b) shows the results when the user is asked to remain quiet, without performing any movement (that corresponds to the *Rest* class). Finally, the user is asked to move his left arm (corresponding to the *Left* class) and the results are shown in Figure 5.9(c).

As expected, the classification of the *Rest* class works very well since, no mistake is made during the 8 s of acquisition. This is unfortunately less the case for the other class decisions. However, the application still works globally well.

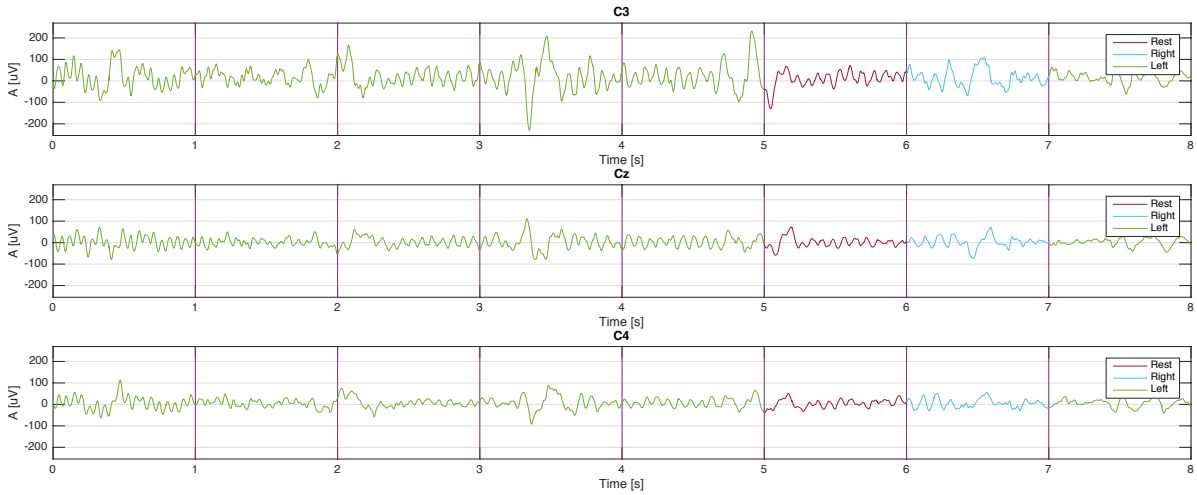
As a final remark, the application works only with real movements, and not with imagined ones. This can be explained by the fact that classifiers are trained with acquisitions of real movements, which are easier to do. Normally, the same brain areas activate when real or imagined movements are done. Real movements are maybe more affected by artefacts, as muscles are in action. To improve the application and make it work truly with Motor Imagery, it would be necessary to train the classifiers with acquisition of imagined movements.



(a) *Right* class



(b) *Rest* class



(c) *Left* class

Figure 5.9: Examples of online classification

Chapter 6

Conclusion

To help tetraplegic patients retrieve some kind of autonomy in their movements or communications, the field of *brain-machine interfaces* (BMIs) has been explored in this document. Indeed, BMIs seem to be the ideal solution as they are controlled by the thoughts of their users. As long as tetraplegic patients keep a functional brain, there is a possibility to use a BMI.

In this work, an experimentation system devoted to the development of BMIs has been imagined and built in collaboration with *Synergia Medical*. The objective of this system is to give the possibility to implement applications related to BMIs to help tetraplegic patients. To demonstrate that the system is appropriate for such a mission, a small and simple application is also implemented. To be able to develop BMIs in a flexible manner, a *Field-Programmable Gate Array* (FPGA) has been chosen as central component for the board.

6.1 Summary

Obviously, the design of such a system is not easy, as there are many degrees of freedom. Indeed, BMIs are at the crossroad of many disciplines: neurophysiology, medicine, computer science, electronics, signal processing, mathematics, machine learning, mechanics, robotics, and even psychology are examples of disciplines that participate to the development of this technology. Chapter 2 has tried to define the main trends followed by researchers in the field of BMIs. In addition, two key aspects have been explored: brain signals on one hand, classification algorithms on the other hand. The type of neural signal which is used to control a BMI is paramount. In a context of development, the emphasis has been put on *Electroencephalogram* (EEG) signals, as they are non invasive and thus easier to acquire. Concerning classification algorithms, they are also very important and have a tremendous influence on the overall performances. The *Support Vector Machine* (SVM) algorithm has been chosen in this work, as it is efficient, flexible and simple.

From there, the experimentation system has been imagined and built. Chapter 3 is related to this process. It first describes the specifications of the system and explains some constraints that need to be taken into account. A lot of choices were made, related to the interfaces of the system, to the supply voltages, to the requirements in terms of speed or amount of logic or to the electrodes type for instance. Then, a global architecture has been defined and three central components have been chosen: a 3D-printable helmet coming from `www.openbci.com/` is used to position correctly the electrodes on the user's head; ADS1299 ADCs from TI have been found to be the best option to acquire, sample and digitise the data; and the 10M50 FPGA from Altera has been chosen because of its great quality-price ratio. Chapter 3 explains

also the design of the *Printed Circuit Board* (PCB), which has been imagined to give to the entire system as much flexibility, debug capability, connectivity and interactivity as possible. Some photographs of the system after assembly are shown. Some parts of the initial design were not assembled (they are let for future works), but the resulting experimentation system is impressive, and moreover, it is totally functional!

The development board is thus ready to be used to build, try and test applications for BMIs. To demonstrate that, a simplistic Motor Imagery application has been imagined. It consists in lighting in real-time the leftmost/rightmost LED of the board when the user thinks about moving his left/right arm. To achieve this, several blocks needed to be implemented, either on FPGA or in MATLAB. Chapter 4 explained and validated the three main implemented parts: first, hardware modules responsible for the SPI communication between the ADCs and the FPGA on one hand, and for the band-pass filtering of the data on the other hand, are detailed; then, the offline learning of the parameters for the classification algorithms is described; finally, additional hardware modules were implemented on FPGA to generate the features from acquired EEG data and to classify them in real time.

The last chapter of this document, Chapter 5, was dedicated to the results obtained in this work. A characterisation of the ADCs and of the hardware design made on the FPGA was proposed. Also, theoretical performances of the implemented classifiers were analysed and compared with tests performed in real-time. Due to several choices that were made, there is room to improve the results, which are not bad nor extraordinary.

The SVM classification algorithms were trained with real movements (and not imagined movements, as initially planned). This choice has been made to ease the acquisition step. As a consequence, the resulting application works, but only when real movements are performed. Although this achievement does not fully meet the initial plans, the developed application still proves that the experimentation system can be used to develop BMIs. In addition, a lot of choices that were made can be changed to improve the results.

6.2 Possible improvements

A lot of improvements could be imagined to obtain better results. Several examples are given below. They can be listed in five categories. Some ideas that fall in different categories may be linked.

1. *Improvements linked to features*:

   In this work, the implemented SVM classifiers used *Autoregressive* (AR) coefficients as features. They were computed on time intervals of 2 s for the learning step and 1 s for the online classification.

   An analysis to find the best values for the lengths of these intervals could be run. Smaller durations could enable the classifiers to detect more accurately relevant time variations. The risk is to catch only parts of the movement which is performed or imagined. In this context, concatenation of features could help too (by grouping the current feature vector with the previous one to catch time variations). Also, other types of features exist and could be generated. A study could be done to determine which are the best features to be used for a given application. This seems to be completely nonexistent in the literature. In addition, all generated features are not necessary relevant. Some features are more

application-relevant than others. Feature selection algorithms could be used to find most relevant features.

2. *Improvements linked to classification algorithms*:

   The SVM algorithm has a lot of qualities and was chosen for this work. For simplicity, the linear kernel has been used.

   For future works, in order to improve the results, other kernels could be tried and might give better results. Other good classification algorithms exist too. Although they are maybe less easy to implement on FPGA, they could be tried if this is possible. Combination of several classifiers could be interesting as well.

3. *Improvements of the learning of classification algorithms*:

   The classifiers were trained with recordings of brain signal that correspond to real movements of the arms. As a consequence, the application does not work with imagined movement.

   To be able to develop Motor Imagery applications, it is then essential to form learning sets based on acquisitions of imagined movements. The acquisition protocol should be modified in this sense. In addition, the recording of more trials should help to improve the results.

4. *Improvements of other processing parts*:

   For the application developed here, the bandwidth going from 7 to 35 Hz is kept for the acquired EEG signals, thanks to a *Band-Pass Filter* (BPF). The goal was to keep at least the μ rhythm, which is confined between 8 to 12 Hz and is related to Motor Imagery.

   Modifying the bandwidth of the BPF could have an impact on the performances of the system. Another aspect that can be taken into account, is the fact that brain signals are not the same for everybody. As an example, the μ rhythm is assumed to be between 8 and 12 Hz, but from one individual to the other, this can change [30]. The subject health status is also critical: disabled people do not have the same capabilities to control a BMI. In this work, the system has simply been used on the authors. Future works should run tests on targeted users.

   Furthermore, EEG signals are linked to brain activity but can be polluted by artefacts, due to muscle tension or eye movements for instance. In this work, no particular care has been taken against that. This is clearly something that has to be studied in the future. Finally, other types of brain signals exist (EEG or non EEG) and could possibly be used.

5. *Extension of the system*:

   The last category of improvements that is presented here concerns possible extensions of the current system. The experimentation system designed for this work is flexible and can be extended.

   The first thing that could be done is to assemble the remaining parts of the PCB. This concerns the *Microcontroller Unit* (MCU) and RF parts. The MCU gives not only additional connectivity to the board, it has also memory and computational power that could

be used to improve the results and/or develop more complex applications. Also, the current board has several not yet used connectors, which were added to the design for future extension. An additional PCB could be imagined and connected to the current system, to increase the computational power, the memory size and/or the amount of logic. This extension could contain additional ADCs to be able to use more than 16 electrodes, if needed.

Clearly, all the examples that were given here show that the number of possibilities to improve the system is infinite. They prove also that the world of BMIs is vast: this work has only covered an infinitesimal part of it! A lot of other aspects, sometimes mentioned in this work but not really covered, need to be taken into account in future developments.

The experimentation system developed for this work should be helpful to further explore the field of BMIs and to build reliable, fast, efficient, safe, easy-to-use, flexible and usable applications. It will help change tetraplegic patients' life, and turn their dreams into reality.

Bibliography

[1] C.F. Aliferis and I. Tsamardinos. Machine Learning Methods for Decision Support and Discovery, Support Vector Machines, 2004. slideplayer.com/slide/4986658/, consulted 29/05/2016.

[2] Altera. MAX 10 FPGA Device Overview, 2015. www.altera.com/products/fpga/max-series/max-10/support.html, consulted 04/06/2016.

[3] Altera. Altera Product Catalog, Version 16, April 2016. www.altera.com/products/configuration-devices/overview.html, consulted 04/06/2016.

[4] Apparelyzed. Paraplegia and Paraplegic. www.apparelyzed.com/paraplegia-paraplegic.html, consulted 02/05/2016.

[5] Apparelyzed. Quadriplegia and Quadriplegic. www.apparelyzed.com/quadriplegia-quadriplegic.html, consulted 02/05/2016.

[6] E. Astrand, C. Wardak, and S.B. Hamed. Selective visual attention to drive cognitive brain–machine interfaces: from concepts to neurofeedback and rehabilitation applications, August 2014. journal.frontiersin.org/article/10.3389/fnsys.2014.00144/full, consulted 26/05/2016.

[7] F. Babiloni, F. Cincotti, L. Lazzarini, J. Millan, J. Mourino, M. Varsta, J. Heikkonen, L. Bianchi, and M.G. Marciani. Linear Classification of Low-Resolution EEG Patterns Produced by Imagined Hand Movements. *Rehabilitation Engineering, IEEE*, 2000.

[8] J.D. Bayliss. *A Flexible Brain-Computer Interface.* PhD thesis, Rochester University, 2001.

[9] T.W. Berger, A. Ahuja, S.H. Courellis, S.A. Deadwyler, G. Erinjippurath, G.A. Gerhardt, G. Gholmieh, J.J. Granacki, R. Hampson, M. Chi Hsaio, J. Lacoss, V.Z. Marmarellis, P. Nasiatka, V. Srinivasan, D. Song, A.R. Tanguay, and J. Wills. Restoring Lost Cognitive Function. *IEEE*, October 2005.

[10] T.W. Berger, M. Baudry, R. Diaz Brinton, J-S Liaw, V. Z. Marmarelis, A. Yoondong Park, B.J. Sheu, and A.R. Tanguay. Brain-Implantable Biomimetic Electronics as the Next Era in Neural Prosthetics. *IEEE*, July 2001.

[11] S. Bhattacharyya, A. Sengupta, T. Chakraborti, D. Banerjee, A. Khasnobish, A. Konar, D.N. Tibarewala, and R.J. Janarthanan. EEG controlled remote robotic system from motor imagery classification. *IEEE*, 2012.

[12] Bio-Medical. Electro-Caps. bio-medical.com/products/electro-caps.html, consulted 31/05/2016.

[13] Mayo Clinic, January 2015. www.mayoclinic.org/tests-procedures/functional-electrical-stimulation-for-spinal-cord-injury/basics/definition/prc-20013147, consulted 07/05/2016.

[14] David. Ça peut vous arriver. www.capeutvousarriver.org/, consulted 02/05/2016.

[15] K. Doherty. Lésions de la moelle épinière : comment les cellules souches peuvent-elles aider?, August 2012. m.eurostemcell.org/fr/factsheet/l%C3%A9sions-de-la-moelle-%C3%A9pini%C3%A8re-comment-les-cellules-souches-peuvent-elles-aider, consulted 02/05/2016.

[16] J.F. Désert. Les lésions médullaires traumatiques et médicales (paraplégies et tétraplégies). cirrie.buffalo.edu/encyclopedia/fr/article/359/, consulted 02/05/2016.

[17] Emotiv. emotiv.com/, consulted 31/05/2016.

[18] EEG Info Europe. Comby EEG Caps. www.eeginfo-neurofeedback.fr/naviprincipale/produits/electrodes/comby-eeg-cap.html, consulted 31/05/2016.

[19] G.E. Fabiani, D.J. McFarland, J.R. Wolpaw, and G. Pfurtscheller. Conversion of EEG activity into cursor movement by a brain-computer interface (BCI). *Neural Systems and Rehabilitation Engineering, IEEE*, 2004.

[20] R. Galli. *Design and evaluation of on-line arithmetic modules and networks for signal processing applications on FPGAs.* PhD thesis, Oregon State University, 2001.

[21] D. Garrett, D.A. Peterson, C.W. Anderson, and M.H. Thaut. Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. *Rehabilitation Engineering, IEEE*, 2003.

[22] B. Graimann, B.Z. Allison, and G. Pfurtscheller. *Brain-computer interfaces: Revolutionizing human-computer interaction.* Springer, 2010.

[23] C. Guger and B.Z. Allison. *Brain-Computer Interface Research, A State-of-the-Art, Summary.* Springer, 2013.

[24] S. Hawking. Stephen Hawking. www.hawking.org.uk/index.html, consulted 05/05/2016.

[25] P.A. House, J.D. Macdonald, P.A. Tresco, and R.A. Normann. Acute Microelectrode Array Implantation into Human Neocortex: Preliminary Technique and Histological Considerations, 2006. www.medscape.com/viewarticle/542359, consulted 27/05/2016.

[26] Florida Research Instruments. fri-fl-shop.com/, consulted 31/05/2016.

[27] Texas Instruments. *Datasheet ADS1299: Low-Noise, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements.* Texas Instruments, 2012.

[28] Texas Instruments. *Datasheet CC430Fxxxx: MSP430 SoC With RF Core.* Texas Instruments, 2013.

[29] Intan. *Datasheet RHD2xxx: Digital Electrophysiology Interface Chips.* Intan, 2012.

[30] T.N. Lal, M. Schröder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, and B. Schölkopf. Support vector channel selection in BCI. *Biomedical Engineering, IEEE Transactions*, 2004.

[31] B. Lee, C.Y. Liu, and M.L.J. Apuzzo. A Primer on Brain-Machine Interfaces, Concepts and Technology: A Key Elemet in the Future of Functional Neurorestoration. *World Neurosurgery*, April 2013.

[32] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi. A Review of Classification Algorithms for EEG-based Brain Computer Interfaces. *Journal of neural engineering*, 2007.

[33] Ludovic. La Roue Tourne. `larouetourne.net/#`, consulted 02/05/2016.

[34] O. Maslennikow, N. Maslennikowa, P. Ratushnyak, M. Wozniak, and A. Sergiyenko. Application Specific Processors for the Autoregressive Signal Analysis.

[35] Synergia Medical. `synergiam.com/`, consulted 07/05/2016.

[36] C. Mestais, G. Charvet, F. Sauter-Starace, M. Foester, D. Ratel, and A.L. Benabid. WIMAGINE: Wireless 64-channel ECoG recording implant for long term clinical applications. *IEEE*, June 2014.

[37] Microsemi. FPGA and SoC Product Catalog, 2016. `www.microsemi.com/products/fpga-soc/fpgas`, consulted 04/06/2016.

[38] J.D.R Millan, P.W. Ferrez, and A. Buttfield. *Non Invasive Brain-Machine Interfaces*. European Spatial Agency, the Advanced Concepts Team, 2006.

[39] Science Museum. `www.sciencemuseum.org.uk/broughttolife/objects/display?id=93225`, consulted 25/05/2016.

[40] Sidney Neurology. `www.sydneyneurology.com.au/electroencephalography-eeg/`, consulted 26/05/2016.

[41] Neurosky. EEG Hardware Platforms, 2016. `neurosky.com/biosensors/eeg-sensor/biosensors/`, consulted 31/05/2016.

[42] OpenBCI. `www.openbci.com/`, consulted 31/05/2016.

[43] OpenVibe. `openvibe.inria.fr/`, consulted 31/05/2016.

[44] J. Pardey, S. Roberts, and L. Tarassenko. A review of parametric modelling techniques for EEG analysis. *Medical engineering and physics*, 1996.

[45] L. Parent. Tétraplégique, je me bats pour mon autonomie, June 2014. `www.maveritesur.com/loic-parent/tetraplegique-je-me-bats-pour-mon-autonomie/791`, consulted 05/05/2016.

[46] B. Perrouin-Verbe, September 2012. `sante.lefigaro.fr/actualite/2012/09/17/19060-nouvelles-technologies-espoir-pour-tetraplegiques`, consulted 05/05/2016.

[47] B. Perrouin-Verbe, T. Albert, I. Laffont, and A. Morin. *Guide affection de longue durée, paraplégie*. Haute autorité de santé, July 2007.

[48] G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *IEEE*, 2001.

[49] M. Ruiz-llata, G. Guarnizo, and M. Yébenes-Calvino. FPGA implementation of a support vector machine for classification and regression. *Neural Networks (IJCNN), IEEE*, 2010.

[50] A. Sergiyenko, D. Ivanov, Juriy Vinogradov, and T. Lesyk. High Speed AR Analysis Based on FPGA.

[51] F. Seuret. Portrait chiffré des blessés médullaires. *Faire Face Paratétra*, September 2011.

[52] A. Shahanaz, September 2015. `fr.slideshare.net/shahanazahamed/artifacts-normal-variants-in-eeg`, consulted 25/05/2016.

[53] Y. Wang, Y. Li, Z. Zhang, X. Gao, S. Gao, and F. Yang. BCI competition 2003-data set IV: an algorithm based on CSSD and FDA for classifying single-trial EEG. *IEEE*, 2004.

[54] K. Warwick, M. Gasson, B. Hutt, I. Goodhew, P. Kyberd, H. Schulzrinne, and X. Wu. Thought communication and control: a first step using radiotelegraphy. *IEEE*, June 2004.

[55] Xilinx. All Programmable Low-End Portfolio, Product Tables and Product Selection Guide, 2015. `www.xilinx.com/products/silicon-devices/fpga.html`, consulted 04/06/2016.

Appendix A

Schematics of the development board

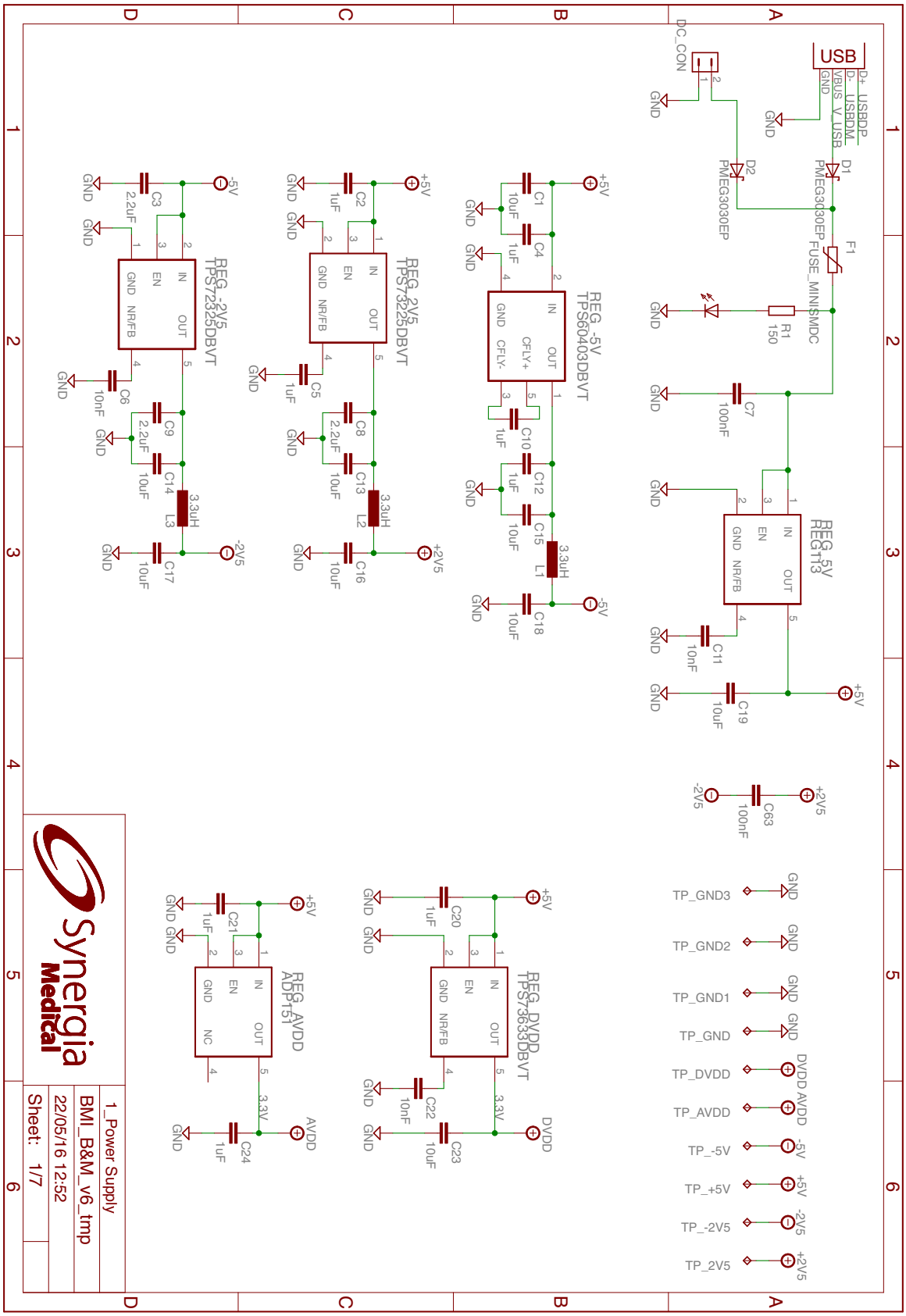


Figure A.1: Schematics - Power Supply

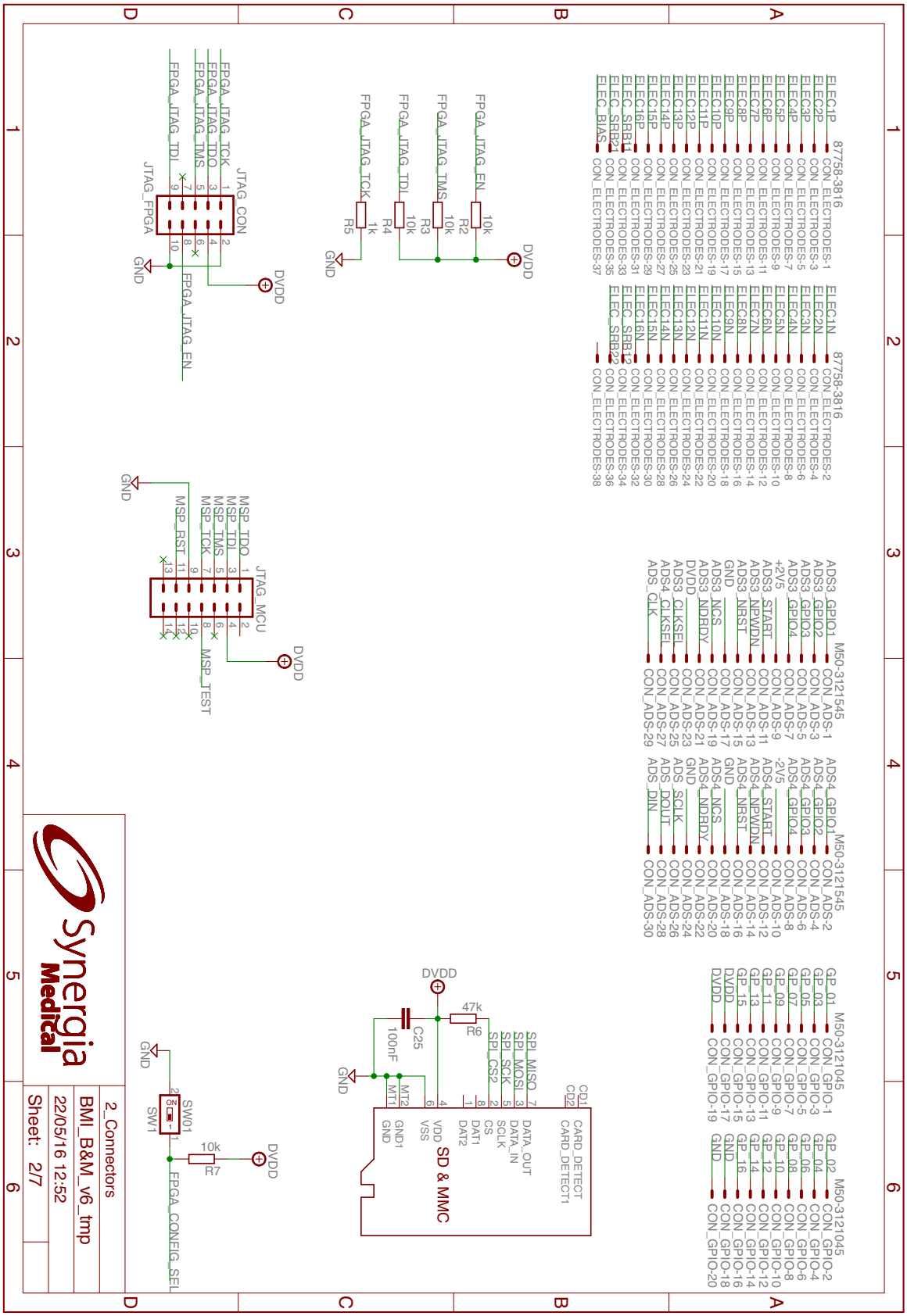


Figure A.2: Schematics - Connectors

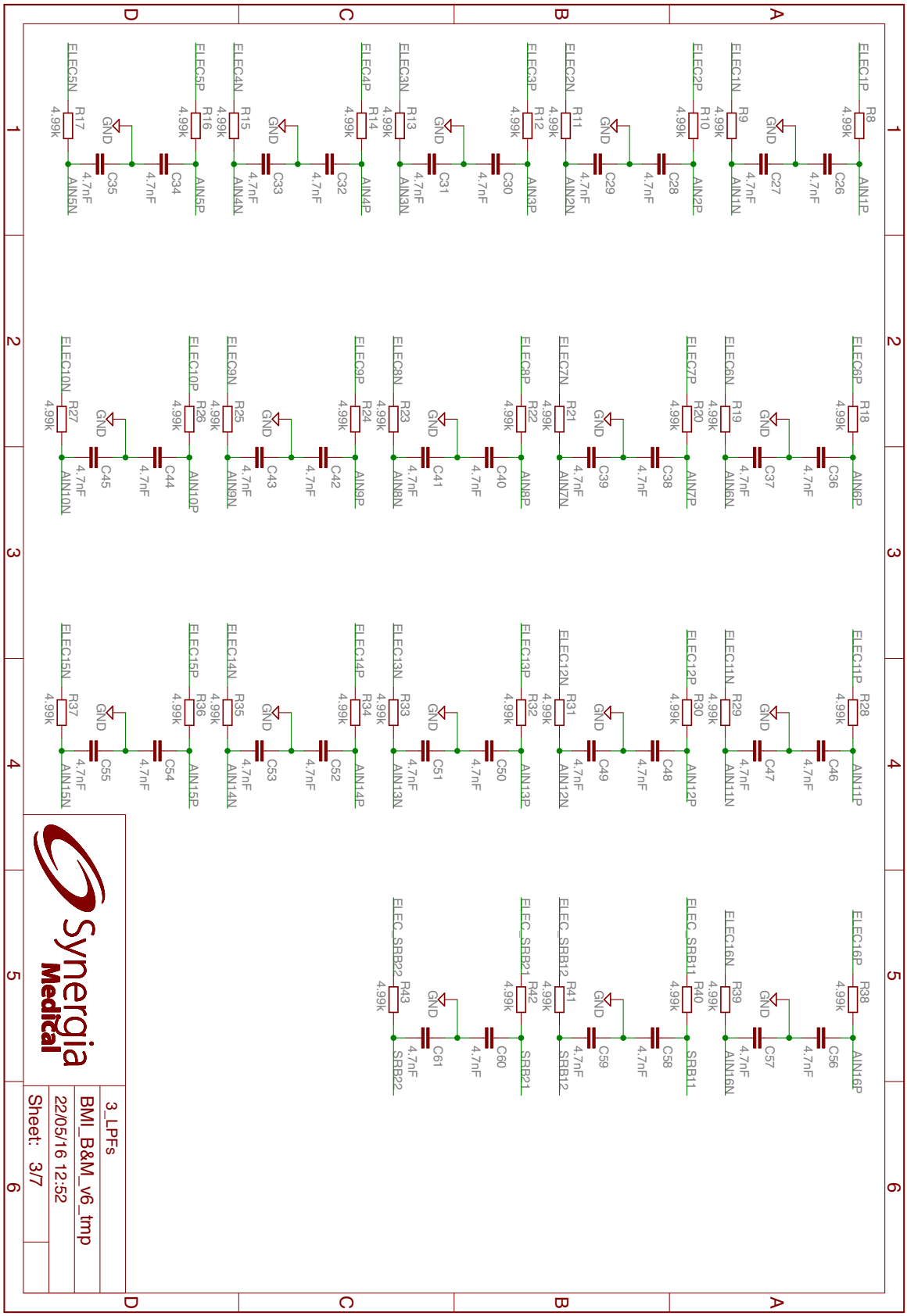


Figure A.3: Schematics - LPFs

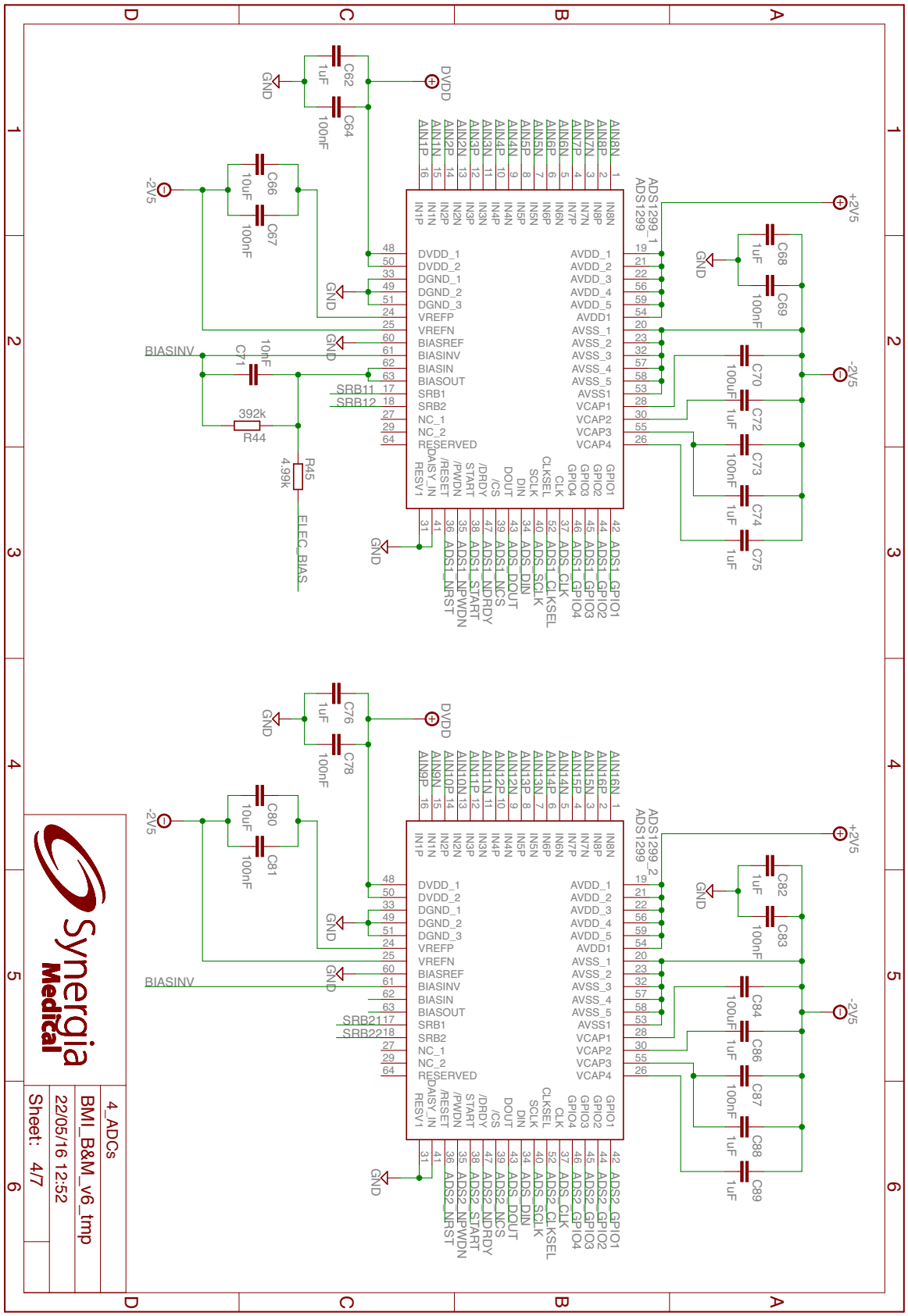


Figure A.4: Schematics - ADCs

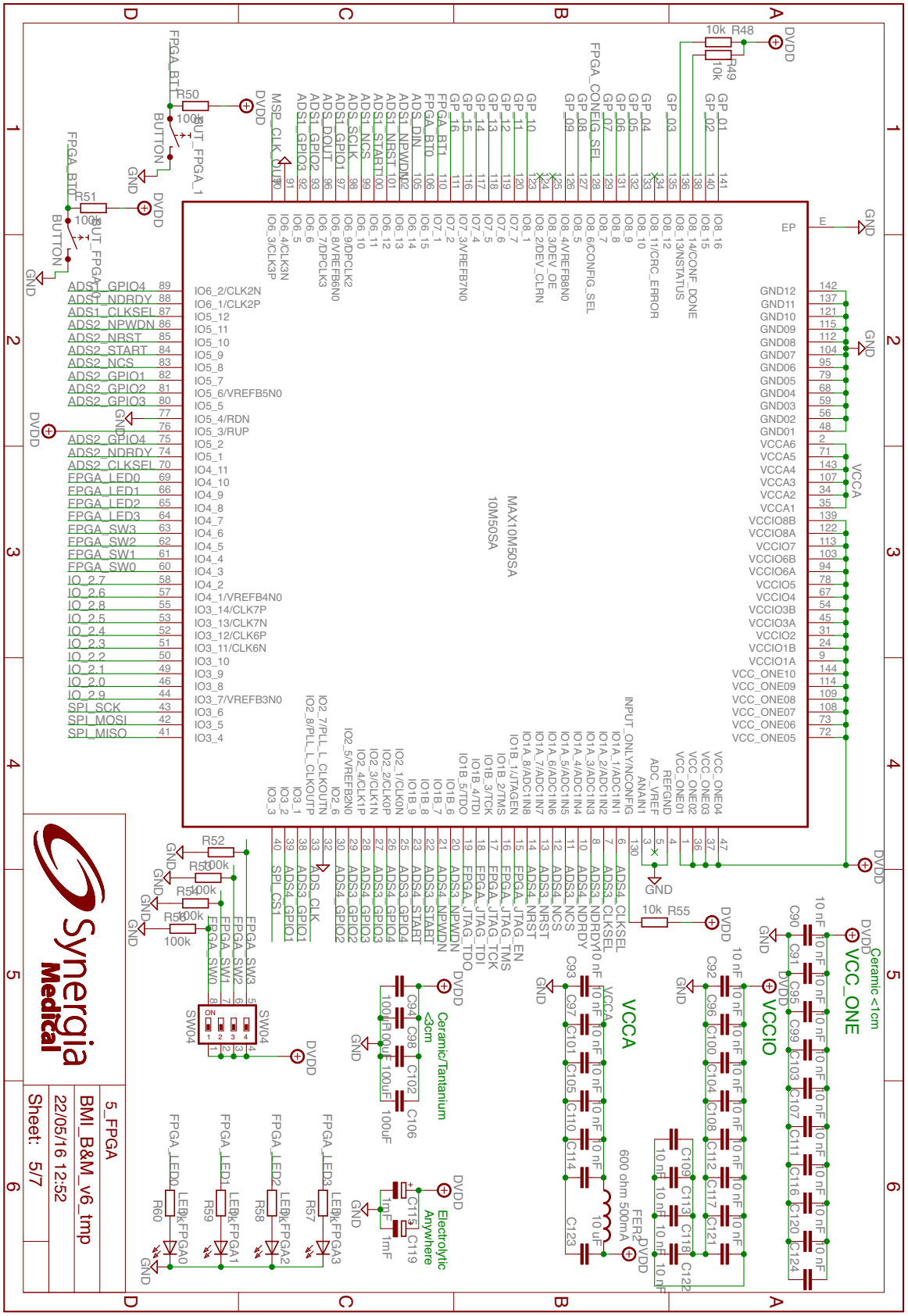


Figure A.5: Schematics - FPGA

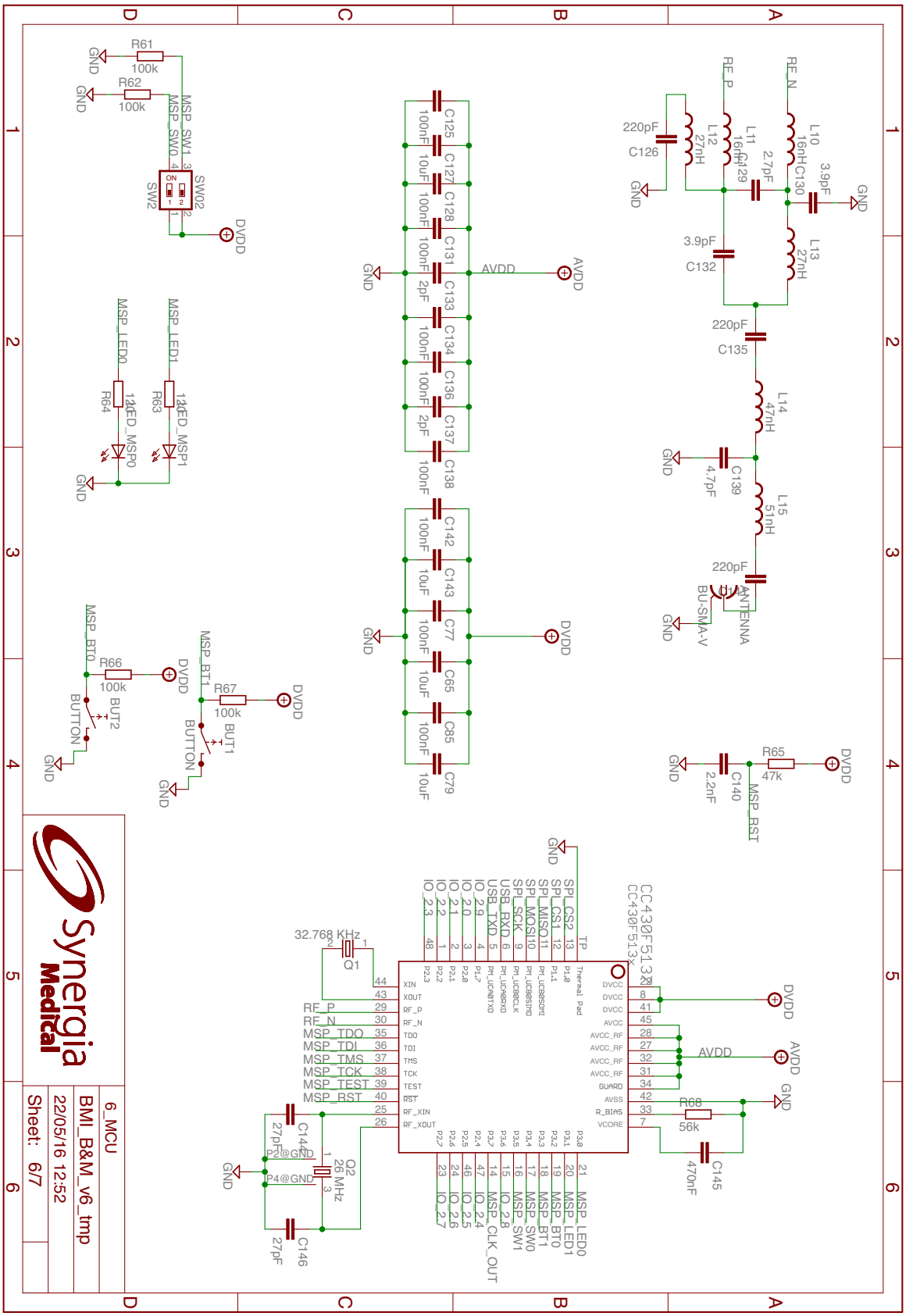


Figure A.6: Schematics - MCU

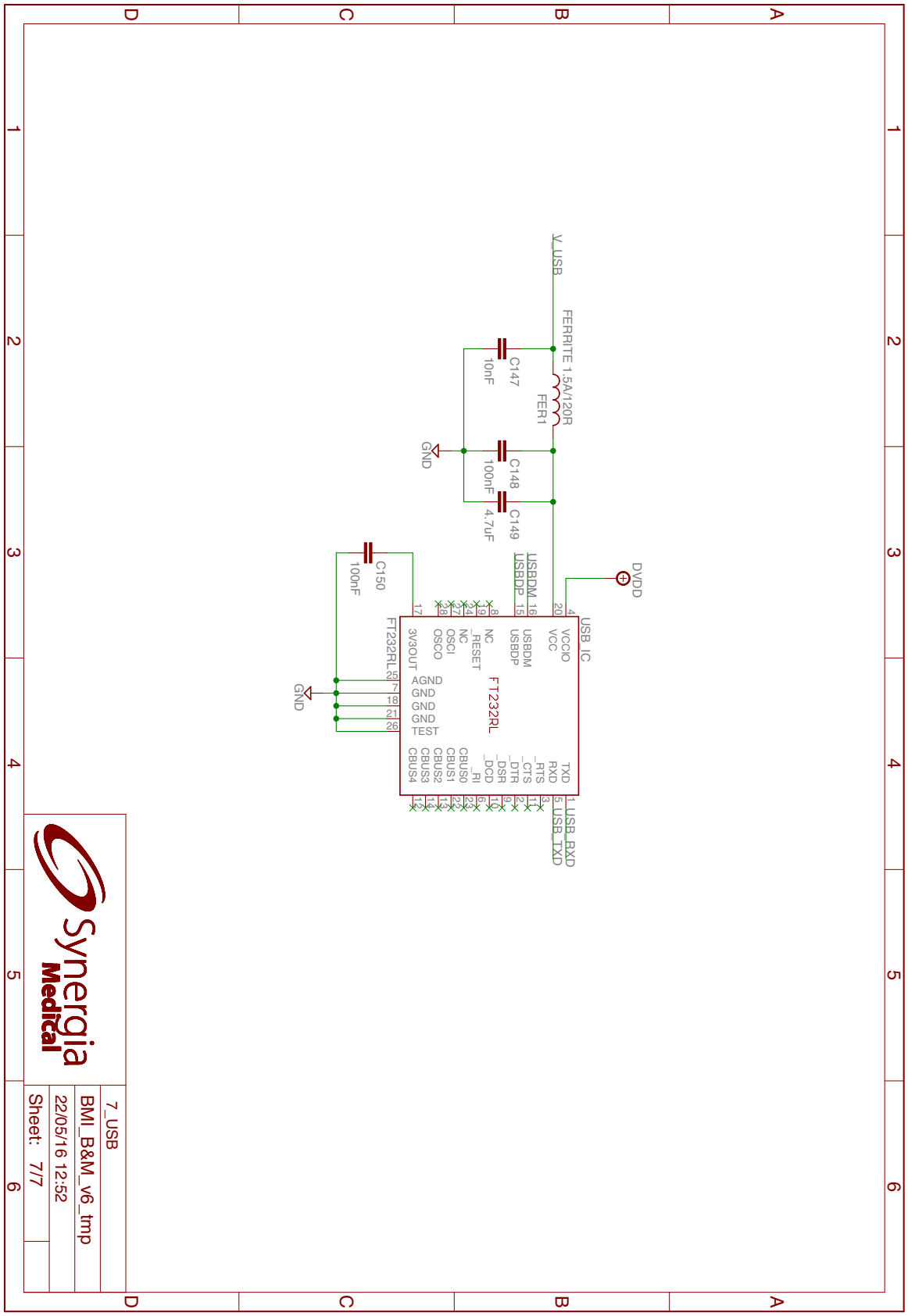


Figure A.7: Schematics - USB

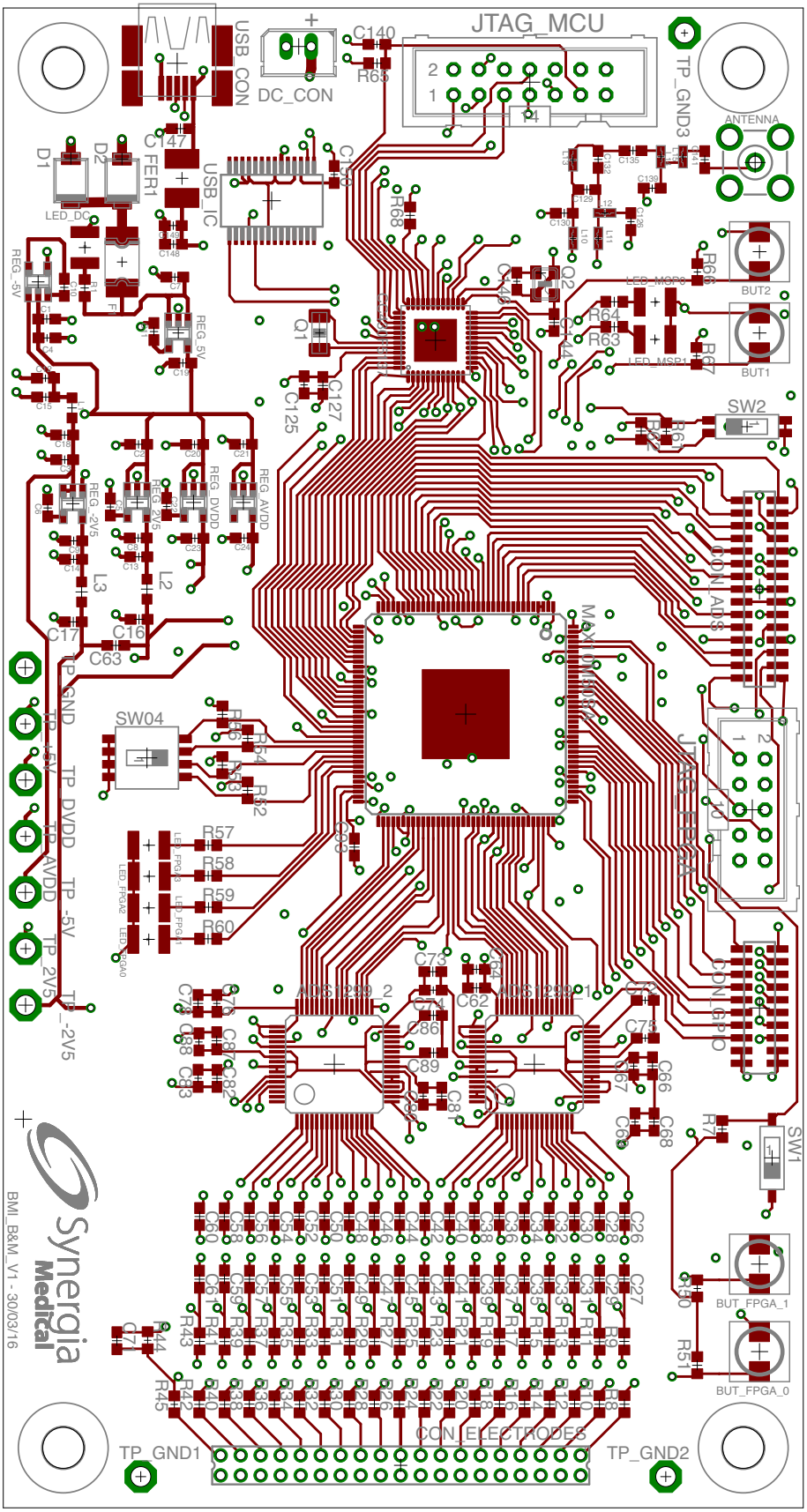


Figure A.8: Board - Layer 1 - Top

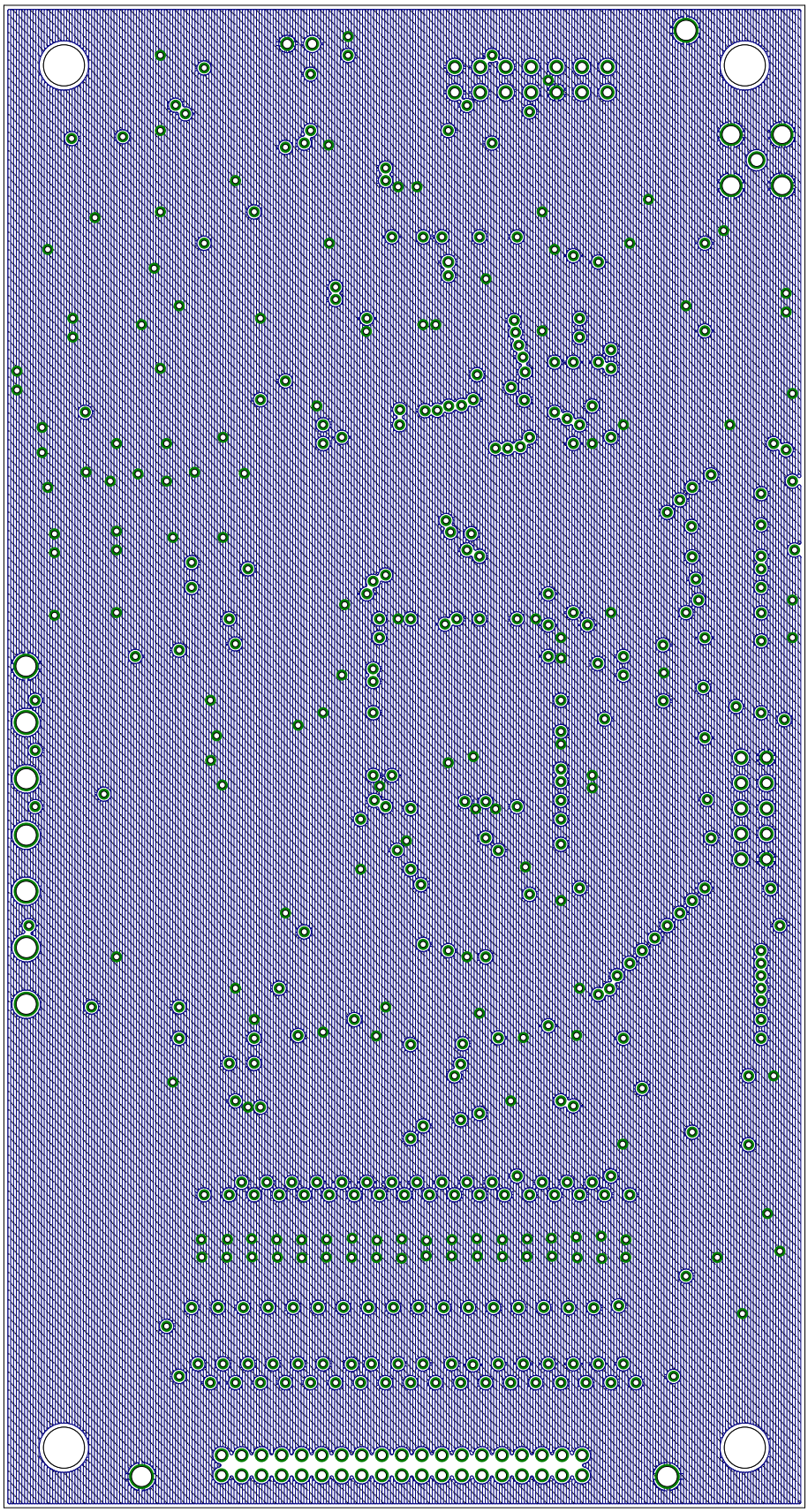


Figure A.9: Board - Layer 2 - Ground

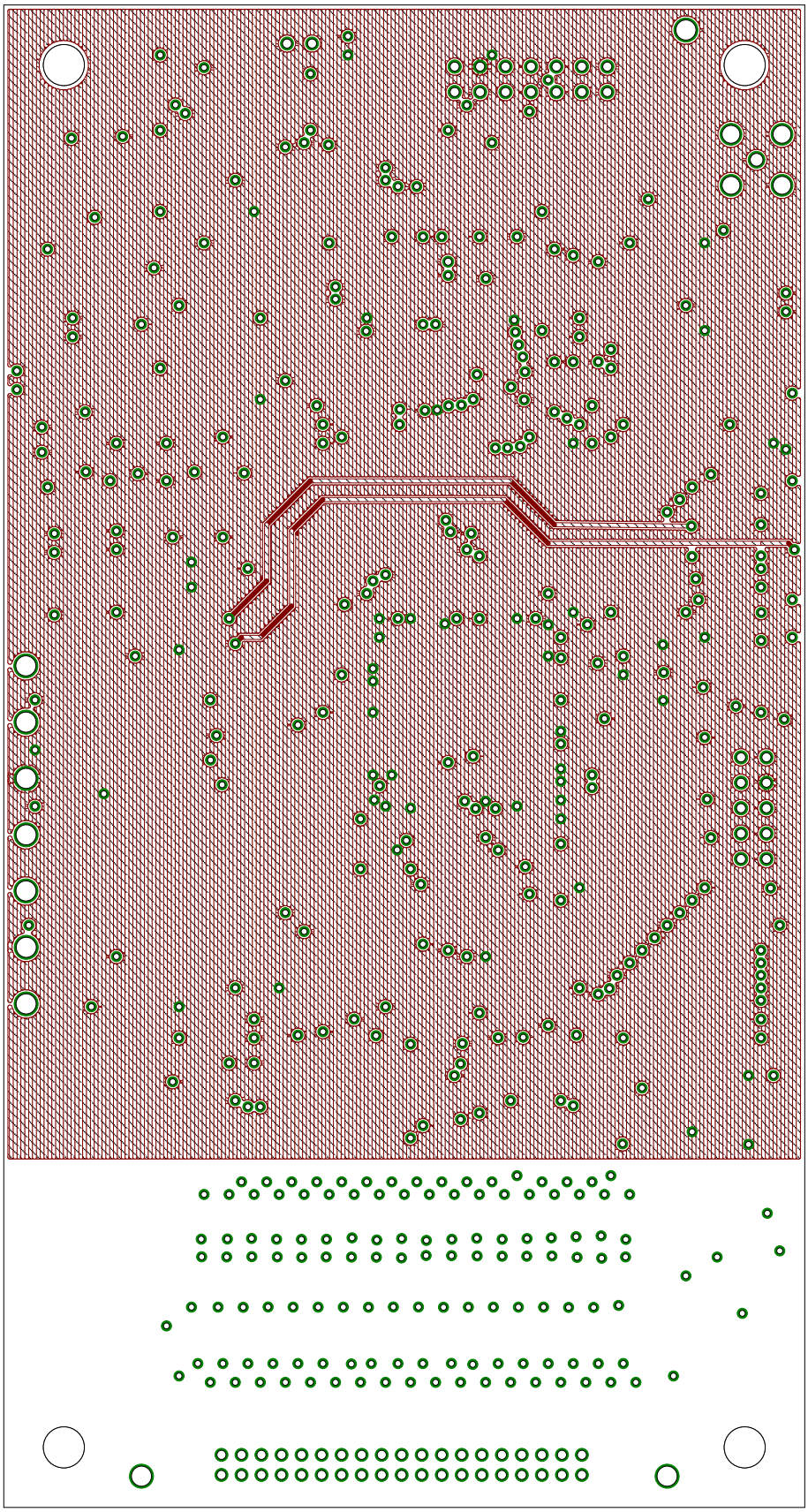


Figure A.10: Board - Layer 3 - DVDD

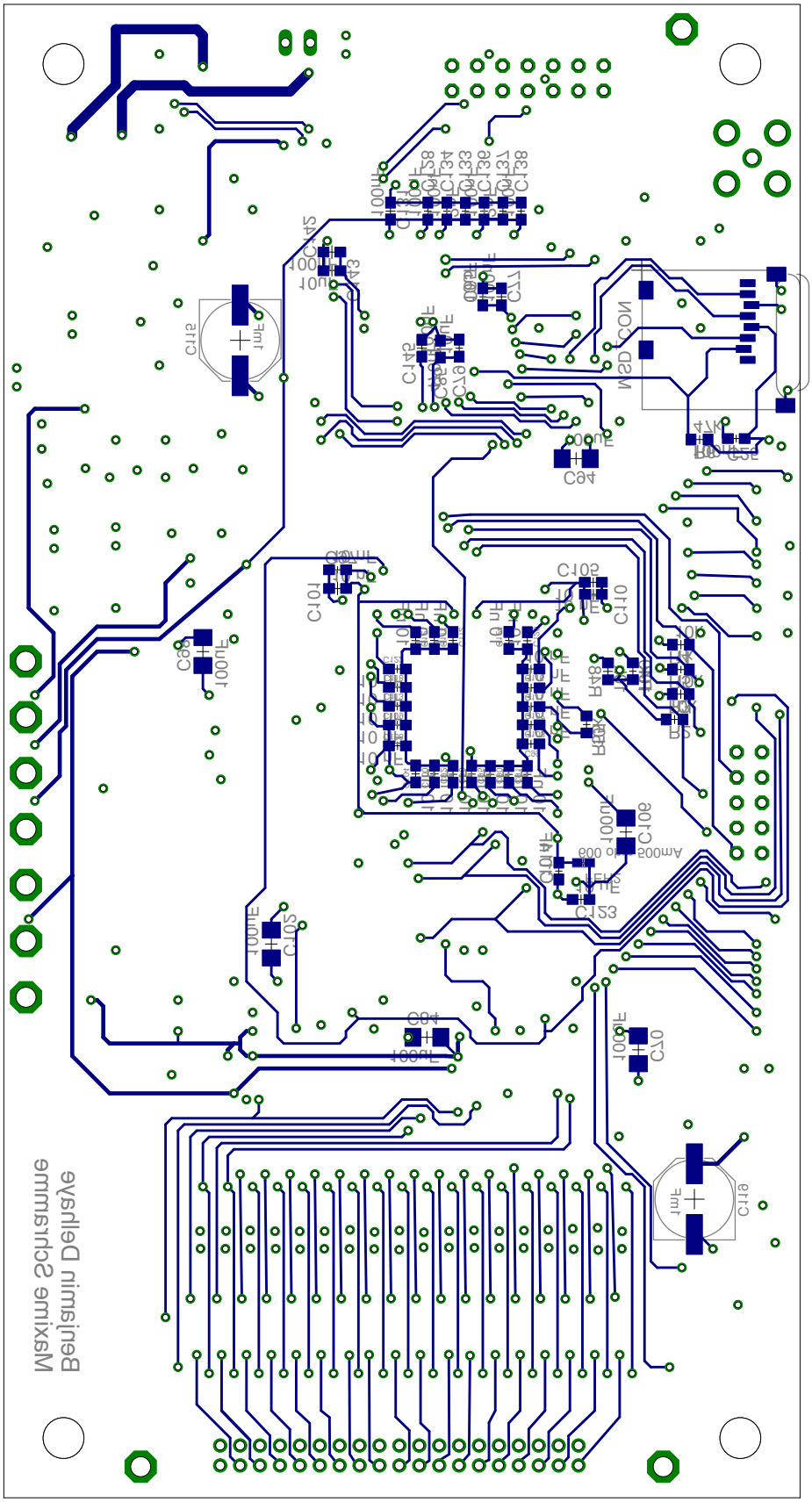


Figure A.11: Board - Layer 4 - Bottom

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**