



NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY

Internship report

Flying a drone using EEG-based brain
signals

Thomas Chamelot

August 16, 2018

Abstract

Brain Computer Interface is a new Human Machine Interface using electroencephalography to measure brain activity. Such interfaces are created in a way that is neither well known nor defined. The event related to the eye movement in an electroencephalogram is often considered as an artifact (undesired event). Yet, the eye blink can be used as a powerful event to control a system. This project tries to propose an efficient and reliable real time method to detect eye blinks in an electroencephalogram in order to control a drone.

Through literatures search and experimentation, an EEG-based real time drone controller was build. A pre-detection algorithm was created in order to facilitate the classification. The Empirical Mode Decomposition was used to break the signal into Intrinsic Mode Functions and two features extraction methods were tested to train the classifier. One method was based on energies and fractal dimensions while the other one was based on statistical values. Comparisons were made between different parameters of the Empirical Mode Decomposition and between the two features extraction methods. The best result provided a system with 0.875 accuracy with a dataset using a single subject. For a transfer learning purpose, those tests where repeated with a dataset containing 5 subjects. The accuracy increased up to 0.975 with this new dataset. A drone controller was designed based on a final state machine.

The final design of this project showed its efficiency in real conditions during a presentation that can be found at the following URL <https://photos.app.goo.gl/XNuoZMikvvwzF9vp9>. Around 10 people were able to control the drone through the system without the need for training.

Contents

1	Introduction	2
1.1	Electroencephalography	2
1.2	Brain Computer Interface	3
1.3	Eye blink paradigm	4
1.4	Empirical Mode Decomposition	5
1.5	Machine learning	6
2	System Design	8
2.1	System Overview	8
2.2	Open BCI	10
2.3	Pre-detection	11
2.4	Creating dataset	12
2.5	Features extraction	13
2.5.1	Statistic based features extraction	13
2.5.2	Energy based features extraction	14
2.6	Machine learning	14
2.7	Real time acquisition	15
2.8	Online prediction	15
2.9	Drone controller	15
3	Results and Analysis	17
3.1	Results with the one subject dataset	17
3.2	Results with the five subjects dataset	18
3.3	Results analysis	19
4	Conclusion	21
	Bibliography	23

Acknowledgment

I would like to express my greatest appreciation to Pr. Olav B. Fosso and Pr. Marta Molinas for making this internship possible and welcoming me in their team. Their willingness to give their time so generously has been considerably appreciated. I would like to offer my special thanks to Luis Alfredo Moctezuma who shared his knowledge with me and spent a lot of time helping me over the course of this project. His previous work and his explanations were essential for its success. I would also like to thank the other interns who worked in the same office. Their participation in this project was really useful and their company was pleasant.

Chapter 1

Introduction

The Electroencephalography (EEG) is a method for the measure of brain activity. It is commonly used in medicine in order to diagnose brain related diseases. EEG requires a specific equipment which used to be very expensive. This is why only structures such as medical institutions and laboratories could afford it. Such equipment has recently become more accessible to a growing audience as the price has greatly decreased.

The increasing accessibility of electroencephalography equipment led to the creation of new applications. As electroencephalography is the measure of brain activity, it is possible to extract information such as thoughts. However, deciphering thoughts from brain activity is a really complex task and not totally reliable from one subject to another. An idea is to extract strongest phenomena such as the eye movements. The eye movements can already be identified using a camera, but they are also easily detectable through an electroencephalogram. Moreover, the eye motion is something that everybody is able to do from the disabled people to the technician in a factory. For example, paralyzed people could control their wheeling chair and workers could interact with assisting systems while having their hands full. Being able to detect the movement of an eye with the electroencephalography would allow an almost unlimited number of new applications as it requires the same base theory.

With the development of new methods for signal processing, it is becoming easier to process data such as electroencephalograms. The Empirical mode decomposition is a recent signal analysis method which is a great tool for processing non stationary signals. Combined with machine learning, it allows to extract features from the data and to automate the detection of patterns without human help. Different machine learning algorithms exist to create what we call a classifier [1]. One of the challenges is to find the best combination of feature extraction and machine learning algorithm to create a reliable classifier in order to build a complete real time system.

1.1 Electroencephalography

The electroencephalogram (EEG) is the recording of brain electrical activity. Richard Caton, in 1875, discovered the electrical activity of the brain by experimenting with rabbits and monkeys.

The first human EEG record is attributed to Hans Berger, a German neurologist, in 1924. Berger published papers, in 1929, where he described EEG change associated with attention and mental effort [2].

The EEG signals (EEG potentials) are generated by the brain's neural activity. They can be observed by placing electrodes on the scalp and amplifying the resulting signal. These signals can be used to examine the subject's brain state. Scientists have proved that frequency and location of the electrodes in EEG have a lot of meaning in its analysis. The following article gives a good overview about brain waves categorizations [3].

Nowadays, the brain is identified well enough to have a normalization for the electrode positioning. The 10-20 international system illustrated in figure 1.1 defines the electrodes' placement and name [4]. The experiments and the real time system were carried out using the 10-20 international system.

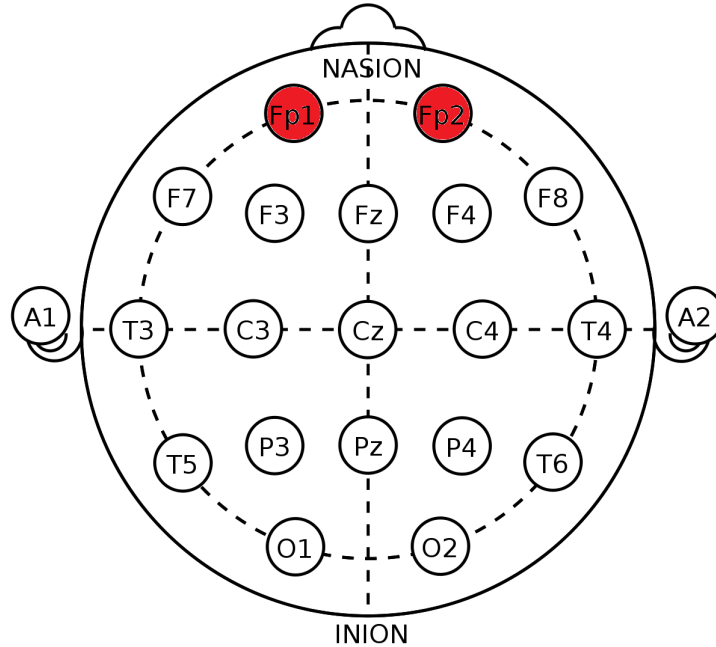


Figure 1.1: 10-20 international system with used electrodes marked in red [5]

1.2 Brain Computer Interface

Brain Computer Interface (BCI) is a new kind of Human Machine Interface (HMI) and it allows human-computer interaction using brain signals. BCI systems have the advantage of controlling complex systems without the use of muscles. The number of applications possible with this new HMI can be unlimited and it can be used in several fields such as video games, robot control, medical diagnosis and research. The main difficulty is that creating HMI with BCI requires a large range of disciplines from the understanding of the biological signals and mechanisms to the signal processing, to the programming of the interface. The BCI systems can be seen in several stages: signal acquisition, pre-processing, features extraction, classification and control

interface.

The study of BCI began in the 70s, Pr J. Vidal is considered as one of the precursors in matter of BCI and published articles in 1973 and 1977, he was using EEG which was presented in the previous part [6]. For the following research projects, the technology was mainly invasive and the applications were limited by the computational power available. The development was also limited by the cost of such an equipment. In the 90s, the research on the BCI progressed quickly. A lot of work was done using invasive BCI on animals. For example, this research showed that it was possible for a monkey to control a robotic arm with its brain. Recently, the non invasive equipment became more accessible and some equipment such as the OpenBCI set are massively used today. This change has brought new attention from various sectors. The research on BCI which was limited to a small number of laboratories 20 years ago is now a common field of research and the number of research papers regarding neural interface network has increased. The progress in signal processing and machine learning opened new possibilities.

1.3 Eye blink paradigm

The objective of this project is to detect and identify eye blinks in an EEG as a first step towards creating realistic systems able to decipher eye movements. The way eye blink produces potential variations is described in this article [7]. The eye blink is an event which occurs several times per minute. It is, most of the time, considered as an artifact. An artifact is an involuntary event, not related to the brain activity. It can be interpreted as the event to decipher and introduce mistakes in the EEG analysis [8].

The voluntary eye blink produces strong signals from the combination of the blinking intention and the muscle activity. It is possible to record signals from eye blinks using the electrodes Fp1 and Fp2 from the 10-20 international system presented in 1.1. It is also possible to detect more specific eye movements as explained in this report [9] but this was not explored in this project.

The eye blink event is remarkable by its shape in the EEG. The following pictures show the difference between blink from right, left and both eyes and also a signal not containing any of those events. The electrodes used for this recordings are Fp1 and Fp2.

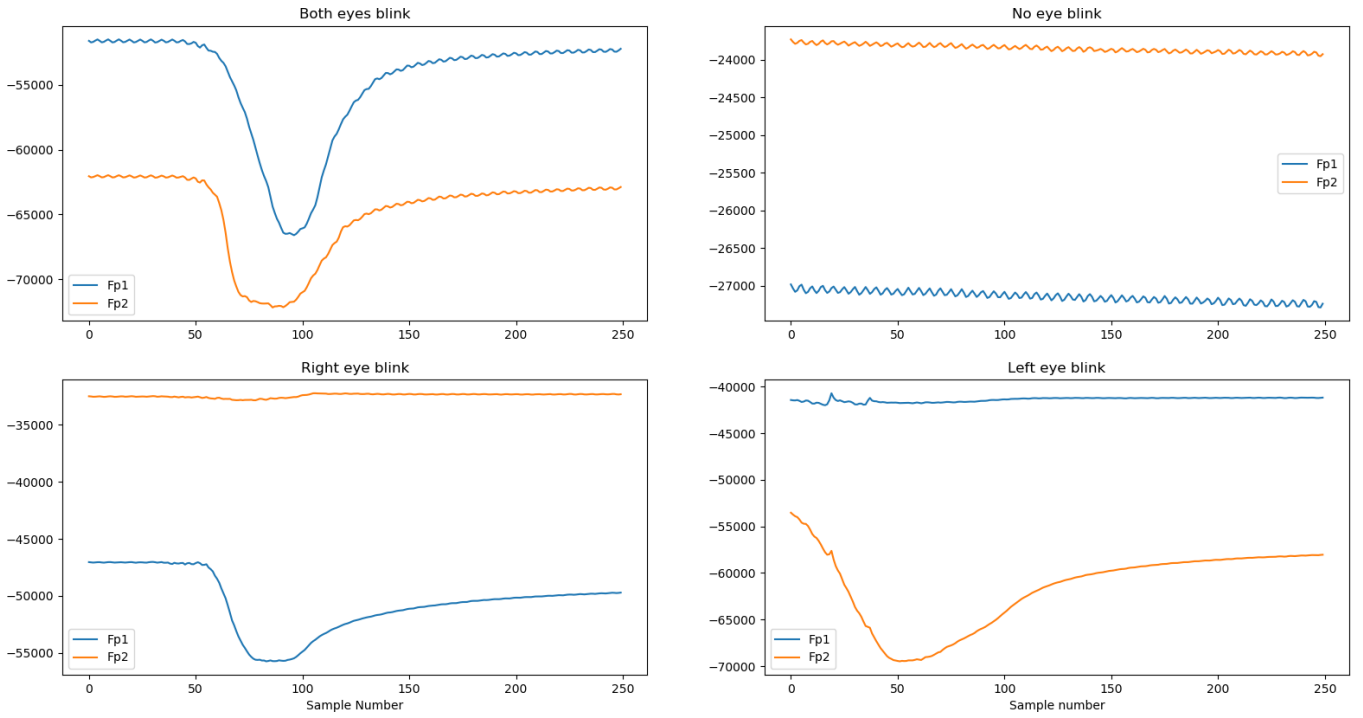


Figure 1.2: Eye blink in EEG seen from Fp1 and Fp2 over a 1 second window

1.4 Empirical Mode Decomposition

In signal analysis, it is possible to classify signals in several categories. The signals from the brain activity are known as non stationary and non linear time series. That means those signals can see their parameters, such as mean or amplitude, change with the time without any linear relation to the cause. This behavior makes brain signals difficult to study with the standard tools such as Fourier transform.

In the 90s, scientists from NASA shared their research about Hilbert Huang Transform (HHT) and the Empirical Mode decomposition (EMD), which allows to work with non stationary and non linear signals. The goal of the EMD is to decompose the signals into Intrinsic Mode Functions (IMF). An IMF is a simple oscillatory signal with a null mean value and a single frequency [10].

The EMD algorithm follows a complex process to decompose a signal. An IMF is obtained through sifting. The mean envelope of the signal is computed and subtracted from it. This step is repeated until a stopping criterion is reached. The sifting process computes one IMF. In order to compute all the IMFs, this process has to be repeated on the signal from which the already known IMFs have been subtracted. The algorithm stops when the resulting time series has only one maximum and one minimum. An example of the different IMFs obtained from the EMD on a brain signal is shown in figure 1.3.

Some interesting properties of this algorithm are that the IMFs are computed from the highest frequency to the lowest frequency. The last time series of the process (the one containing only one maximum and one minimum) is called the residue. It is often composed of the mean

value of the original signal and some algorithmic noise. The sum of the IMFs and the residue is the original signal.

Some IMFs among the last ones computed might also only contain algorithmic noise. This is why the choice of the IMFs is important while using the EMD. The IMFs selection for EEG has been studied and a method to choose the best ones has been presented in this publication [11].

Nowadays, the EMD is used in a large range of applications, including the study of brain signals. It is a powerful tool for the offline signal processing. Yet, as this algorithm is two times iterative, it is too time consuming to process all the IMFs in real time. This is why, in this project, the number of IMFs extracted was limited. Further explanations will be given in the next chapter.

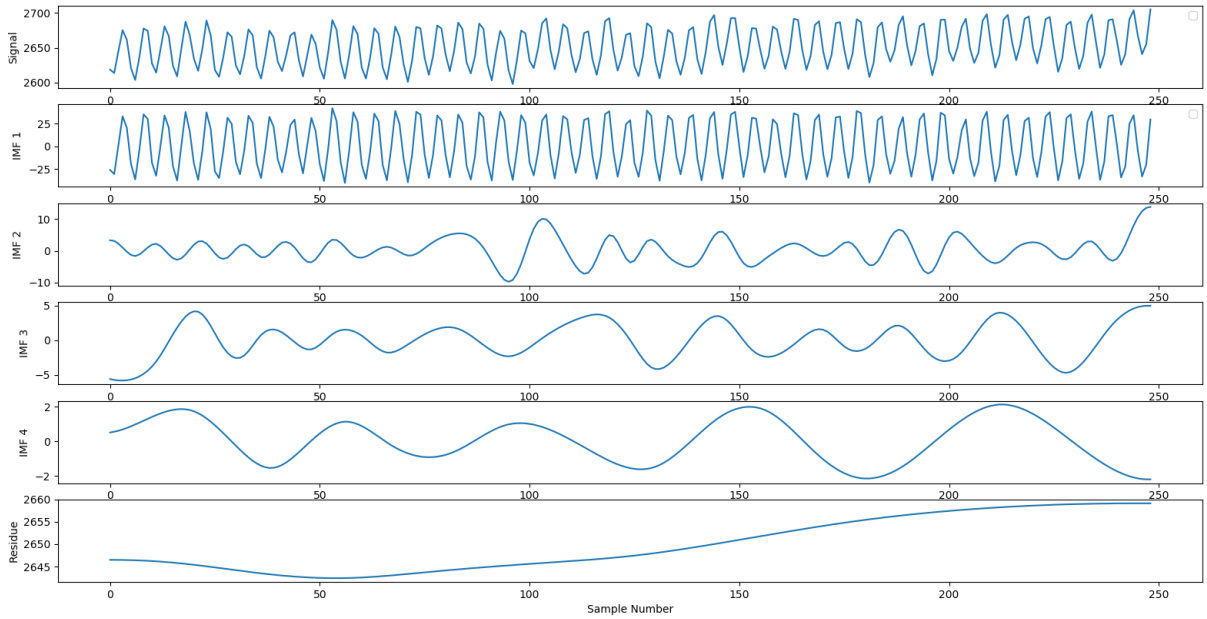


Figure 1.3: Example of EMD on an EEG

1.5 Machine learning

Machine learning is a branch of artificial intelligence. The goal of machine learning is to enable a computer to detect patterns within a dataset. It uses algorithms and statistics to help the computer to *learn*. The learning can be supervised or not and the output of the algorithm can be discrete or continuous. In this project, machine learning techniques were for supervised learning purpose (classification). The classification consists of predicting the belonging of data to a class. This takes place in two stages, the first one is the learning stage. The algorithm is fed with labeled data in order to create what is called a classifier. This is a supervised learning process which happens offline. Then the classifier is used to predict the class of unlabeled data. This stage can be either offline or online.

Machine learning simplifies the processing of complex data. It is powerful for problems

such as the classification of the brain signals. Nowadays, classification is a common way of processing brain signals because it is difficult to detect patterns with standard algorithms. For this project, several classification algorithms have been compared in order to find the best one.

Classifiers can be compared based on their accuracy. As classification uses statistics, it is possible for the algorithms to make a wrong prediction. A prediction can be divided in four categories. The true positive (TP) and the true negative (TN) are when the prediction coincides with the effective class, the false positive (FP) and false negative (FN) are when the prediction does not coincide with the effective class. The accuracy of a classifier is given by $ACC = \frac{TP}{TP+TN+FP+FN}$ and the precision is given by $PREC = \frac{TP}{TP+FP}$

Chapter 2

System Design

In this chapter, the design of the project from EEG-based brain signal is presented. It will begin with a short overview of the project and then go in depth into essential parts of the system. The main purpose of the system is to train a machine learning classifier with recorded EEG data and to use it in real time to control a drone. The classifier is trained to detect the eye blinks. This chapter will focus on the software part of the system. The resources used such as hardware and libraries are open source.

2.1 System Overview

The system designed during this internship is divided into several components. The figure 2.1 shows a high-level block diagram of the complete system, establishing naming conventions. The system will process EEG data to detect eye blink, so the first step is to get the EEG. The EEG is acquired through with an OpenBCI module, which is an open hardware and open source equipment. The electrodes used for the EEG acquisition are Fp1 and Fp2 as presented in 1.3.

Once the data is available on the computer, it is processed. The data processing begins with the pre-detection algorithm. It tries to select the part of the signals from each electrode that contains eye blink events. Then, the signals are decomposed into IMFs on which the feature extraction will be made. The features extraction represents the initial EEG signals in simple values usable with machine learning algorithm.

At this stage, the system is composed of two modules. The training module will merge records to create what is called a dataset. It is a set of signals already classified. This set will be used to train the machine learning algorithm by showing the correct label for each signal. Another dataset can be used to validate the resulting classifier with cross validation. The second module is the real time system. It uses the previously created classifier to predict the class of new unlabeled data. The predictions can be used to translate the class into command for the drone controller.

The main contributions of this project are the real time module and the training module. The hardware and the library used in this project are open source and will be quickly presented in this chapter.

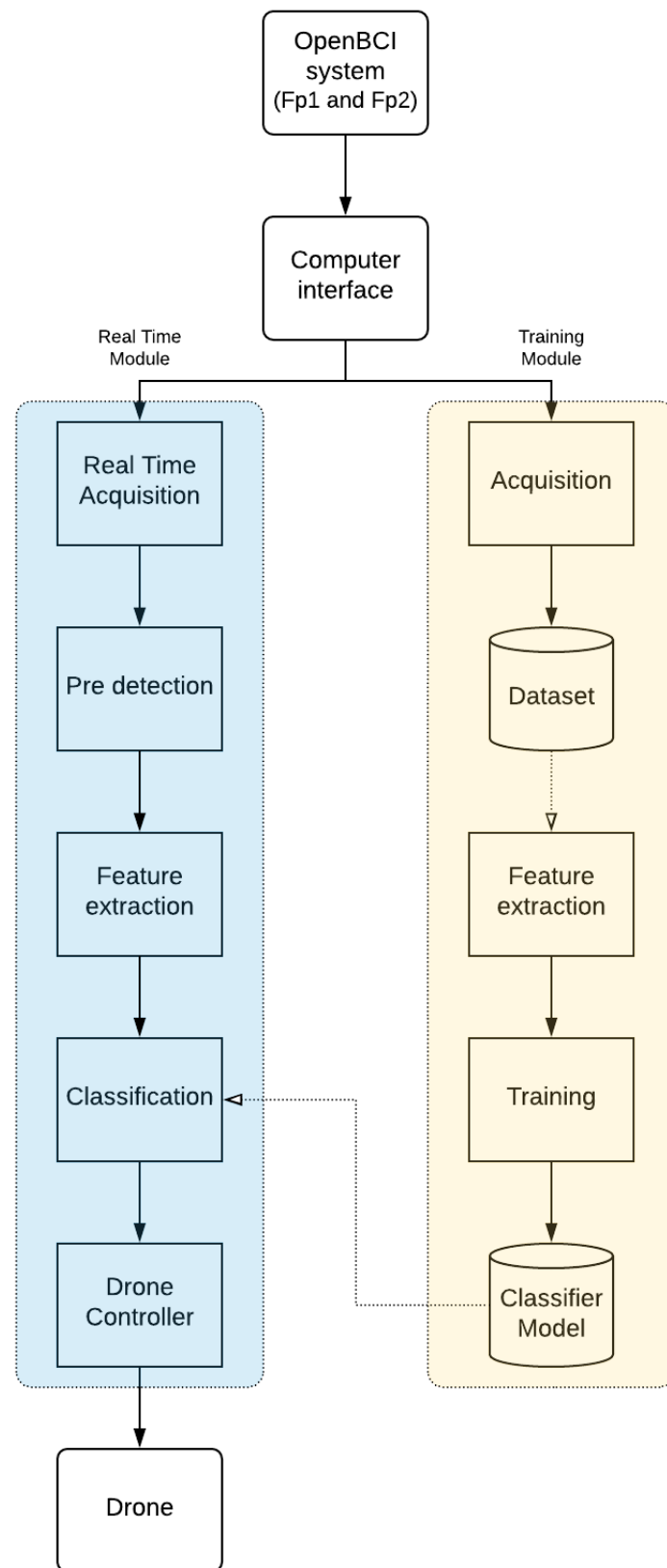


Figure 2.1: Overview of the system design

2.2 Open BCI

The Open BCI headset is a device which allows to measure brain signals through EEG using dry electrodes placed on the human scalp. It uses the combination of Ultracortex Mark IV and the Cython board which are low cost, open sources equipments. The communication between the Cython board and the computer is done through Bluetooth and a USB dongle. The hardware can be seen in the picture 2.2. The headset is on the left of the picture while the USB dongle for the communication is on the right. In the center, the Cython board is the top one while the bottom one is the daisy module which is an extension.

This module enables EEG recording from 8 channels and up to 16 using the daisy module. The BCI samples the EEG signals with a rate of 250Hz. It is then transmitted to the computer through the USB dongle. the communication between the USB dongle and the software is realized with the OpenBCI Python API [12].

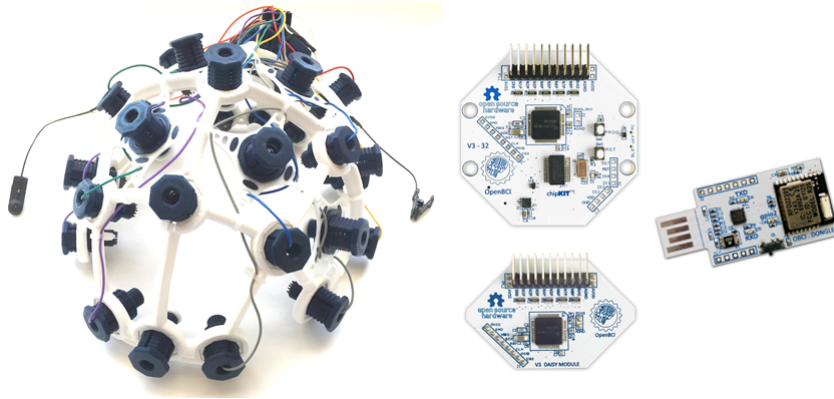


Figure 2.2: OpenBCI hardware

According to the 10-20 international system, the previous project on this subject and some experience through many manipulations during this project, the OpenBCI system was configured to use two electrodes Fp1 and Fp2. The recorded data is stored in a list which constitutes the acquisition window. The length of this window was determined experimentally. The main limitation for the maximum duration is the computing time of the features extraction which will be presented later. As this system has to run in real time, it must ensure that each part's duration does not exceed the total time allotted to the execution. A window of one second was long enough to capture the eye blink event and short enough to limit the computational time.

The acquisition module is implemented as a single Python class which is called "Recorder". This class manages the connection with the hardware, the recording of the data, the window length, and the transfer of the data to the next part of the system. The recording is done in a specific thread.

The development of this project was impacted by the handling of the OpenBCI, as a problem occurred with the recording. The measures recorded from several channels were not containing any EEG signals. Two weeks were spent identifying and correcting this issue. It was caused by a defect in the ear clips and was solved by using the ear clips from the other OpenBCI set.

2.3 Pre-detection

The system has to pre-detect events before sending them to the classifier or adding them to the dataset. The decision of adding a pre-detection algorithm was made after several tests on the real time system. At the beginning of the project, it was planned to only use a sliding window and to classify the signals every second. This solution's main issue was to be able to select the entirety of the event in a single window, instead of having an event cut into two different windows.

To solve this issue, the pre-detection algorithm provides a window to classify only when there is an event. At this point, it is possible to think that the pre-detection itself is sufficient to realize the eye blink detection. Actually this pre-detection algorithm also detects artifacts such as head movement or speaking. This is why the classification process is still needed.

The pre-detection algorithm normalizes, binarizes and filters the signals in order to find something looking like an event. As an event is just an important drop in the signal, the pre-detection is looking for a falling edge followed by a rising edge. As this kind of events occurs a lot in brain signals due to the noise, all the events with a duration below 0.2 seconds were neglected. All these steps can be seen in the picture 2.3. On the left side, the signal contains a blink while on the right side, the signals contains only noise.

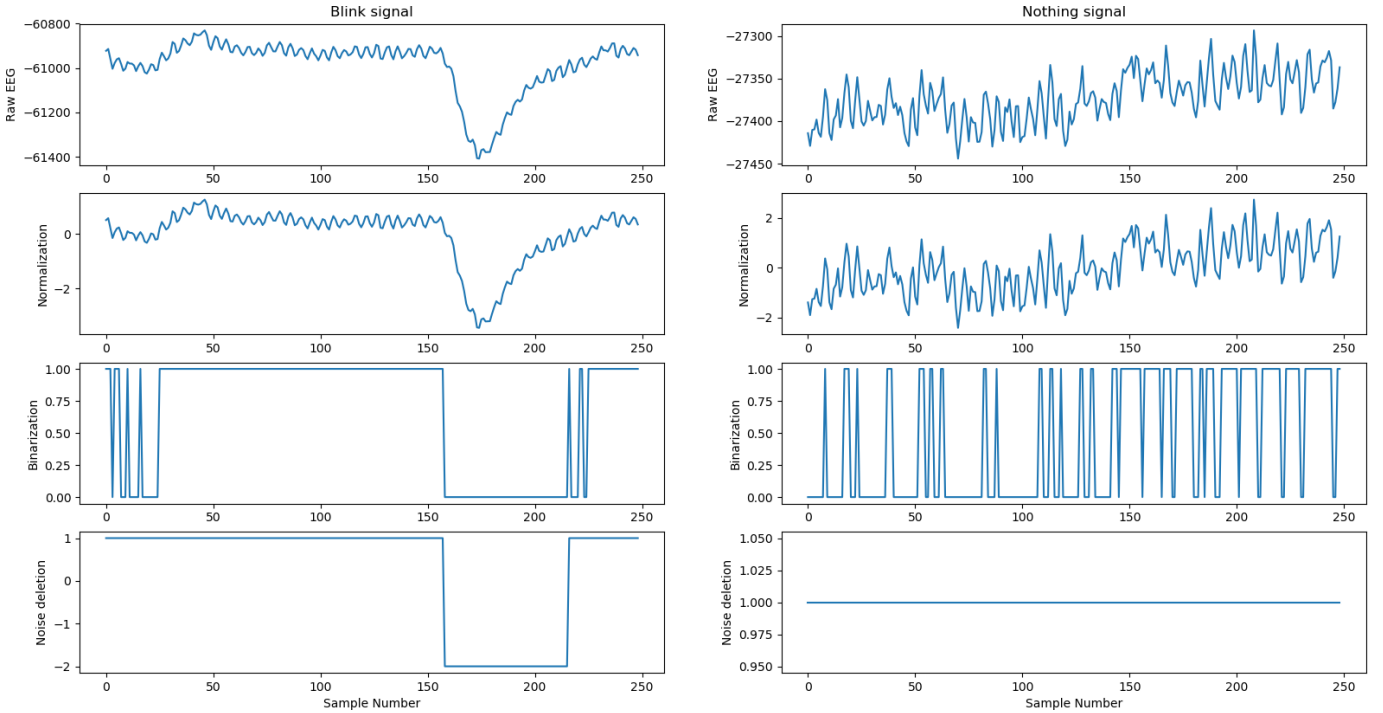


Figure 2.3: Pre-detection steps for blink related signal (left) and blink non related signal (right)

Signal normalization is done using this formula: $Snorm_k = \frac{S_k - \bar{S}}{\sigma}$ where $Snorm_k$ is the k sample of the normalized signal, S_k is the k sample of the original signal, $\bar{S}$ is its mean value and σ is its standard deviation.

Signal binarization is done by applying a threshold to the normalized signal. If a sample is

above the threshold, the resulting binarized value will be 1, whereas any value below will result in a 0. The observed values oscillated around 0, which was then selected as the binarization threshold. Thus, the ambient noise would be categorized as 1 while the blink event, being a substantial drop, would show up as 0. However, part of the ambient noise can occasionally fall below the threshold, causing noise to appear as very short drops after the binarization.

Finally, the noise deletion is done by removing all the events (falling edge then rising edge) with a duration below 0.2 seconds. This value was chosen experimentally as a human being cannot blink in less than 0.2 seconds, therefore any shorter drops can only be noise.

Once the event is detected, the pre-detection algorithm calls the prediction callback function and then locks itself for the next one second in order to let the event disappear from the window. This was made to ensure that there is only one event in each window. This solution is still problematic as the pre-detection algorithms might select an artifact. This means that the events in the next second will be ignored. To solve this behavior, one solution would be to display an information to the user when a blink is expected.

2.4 Creating dataset

A dataset needs to be created in order to perform the training stage. A dataset is a file that contains the raw EEG data and the associated label (the eye movement). The goal was to be able to differentiate the blink from left eye, right eye, or both eyes and no blink at all. This four events can be considered as the four classes used during this project. They were named using the following nomenclature:

- nothing: for an event with no blink inside
- left: for an event with a blink from the left eye
- right: for an event with a blink from the right eye
- both: for an event with a blink from both eyes

Several datasets were created with 10 repetitions per class. Two datasets were recorded with me, Thomas Chamelot, as the only subject. One dataset was used for the training stage while the other one was used for offline cross validation.

Two other datasets were created using five different subjects. Each one was represented by two repetitions per class for a total of 10 repetitions per class with all the subjects together. This was used to compare the accuracy and the reliability of the system from one person to another.

The datasets were recorded using a console interface. The pre-detection algorithm was used in order to have the same behavior for the recorded window in realtime and in the dataset. After the acquisition of each repetition, the signals were displayed and a graphical inspection was made to validate the data.

2.5 Features extraction

The features extraction is used to represent the initial EEG data with few values in order to process the classification.

The first step in features extraction was the use of the EMD. The goal is to decompose the signal in IMFs. As explained previously in 2.2, the acquisition window was limited by the computational time. The same problem occurs with the EMD. This algorithm being time consuming, the number of IMFs extracted has to be limited in order to respect real time constraints. During this project, the number of IMFs was limited to 2 plus the residue. This research publication [11] has proposed a method to select the best IMFs, however, due to time constraints, I was unable to perform more advanced work on this subject.

With the IMFs, two features extraction methods were used. One was based on the extraction of statistic values, and the other one was using energies and fractal dimensions as proposed by Luis A. Moctezuma in [13]. Previous works done by NTNU's students have also been used to study the feature extraction [14] [9]. As I am not an expert in brain signals, I simply compared the efficiency of these features extraction methods and I can not prove that any of these methods is the best one.

As the feature extraction method should be easily changed, a single function is used in the main part of the system "`get_features`". This function is implemented in the `featureExtraction.py` file. This means that the modification of the features extraction method does not require to modify the full project but only this file.

2.5.1 Statistic based features extraction

The use of statistical values for the features extraction of signals is really common and can be found in many other research projects. Here, it consists in 8 values which represent the behavior of the signal of each channel Fp1 and Fp2. The values used are listed below:

- Mean
- Variance
- Standard deviation
- Kurtosis
- Skew
- Maximum
- Minimum
- Median

Those values were computed with the NumPy, statistics and ScyPy modules from Python 2.

2.5.2 Energy based features extraction

The use of Energy and fractal dimensions is common in the study of brain signals. The values used are the following:

- Instantaneous energy
- Teager Energy
- Higuchi fractal dimension (HFD)
- Petrosian fractal dimension (PFD)

Fractal dimension provides an indication about the complexity and the shape of the signal which can be really powerful for brain signals. The algorithms for the HFD and the PFD come from the python module PyEEG which offers several algorithms to extract features from EEG.

2.6 Machine learning

With the data extracted from the dataset, it is now possible to work with machine learning. The goal is to train several machine learning algorithms to select the best one. This part is composed of two software components. One is the selector which picks the best machine learning algorithm with a given feature extraction method. The other classifies new entries for cross validation tests and the real time system.

The machine learning algorithms are provided by the scikit-learn Python 2 module. It is a free open source machine learning library [15], the use of other libraries might change the result of this project.

The machine learning algorithms used in this project are the following.

- Random Forest (RF)
- Naive Bayes (NB)
- K Nearest Neighbor (KNN)
- Decision Tree (DT)
- Support Vector Machine (SVM)

The comparison between algorithms was made using the "`cross_val_score()`" of scikit-learn. This function used the mean accuracy which was presented in the Machine learning introduction 1.5 and the K-fold cross validation with 10 folds. A problem that might occur with machine learning is the overfitting. It is when the classifier fits too much with the training dataset. The consequence is the incapacity of this classifier to label new unknown entries. Despite announcing a great score, the classifier will have really poor accuracy. To validate the classifier, a test script was created for this project. It uses the test dataset to double check the classifier. As the signals from this dataset are already known, it is easy to check if the result is correct. This script uses the precision as a score as presented in 1.5.

2.7 Real time acquisition

The real time acquisition has some specificity that differs from the dataset acquisition. In order to always have a 1 second window, the data is stored in a list which contains 1 second of recording. In order to keep this record consistent, each time a new point is recorded, the oldest point of the recording is deleted. Moreover, the real time acquisition has to handle the decision to classify a new entry. In order to do this, a pre-detection algorithm is used directly in the real time acquisition and will be presented in the next section. If the pre-detection algorithm reports a possible event in the signal, the window has to go through the features extraction process and the classification. This is made possible through to a callback function. A callback function might be a complex solution to this problem in comparison with a memory shared structure between the real time acquisition and the classification. However, shared memory manipulation causes issues in concurrent programming, which is avoided by the callback solution.

2.8 Online prediction

The online prediction is ensured by the Predictor class. This class will load the classifier from the training stage and use it to label new entries. The Predictor has a `predictClass()` method which will be used as a callback function by the Recorder. It will predict the eye movement contained in the data and an indication about the accuracy of the prediction. This information will be put in a Queue shared with the drone controller.

2.9 Drone controller

The drone controller is the final part of the system. It will use the information from the prediction Queue to control the drone. As the drone is an AR drone 2.0, the simplest way to remote control the drone is to use the drivers from Playsheep.com [16].

The main point about the controller is to have a simple and user friendly HMI. As the main goal of this project was using eye blinks from left, right or both eyes, the first design of the controller was using these 3 movements. Due to a lack of time, the final design was only using the both eye blink movement to control the drone. The controller is implemented with a final state machine structure. The original design of the controller is presented in the picture 2.4

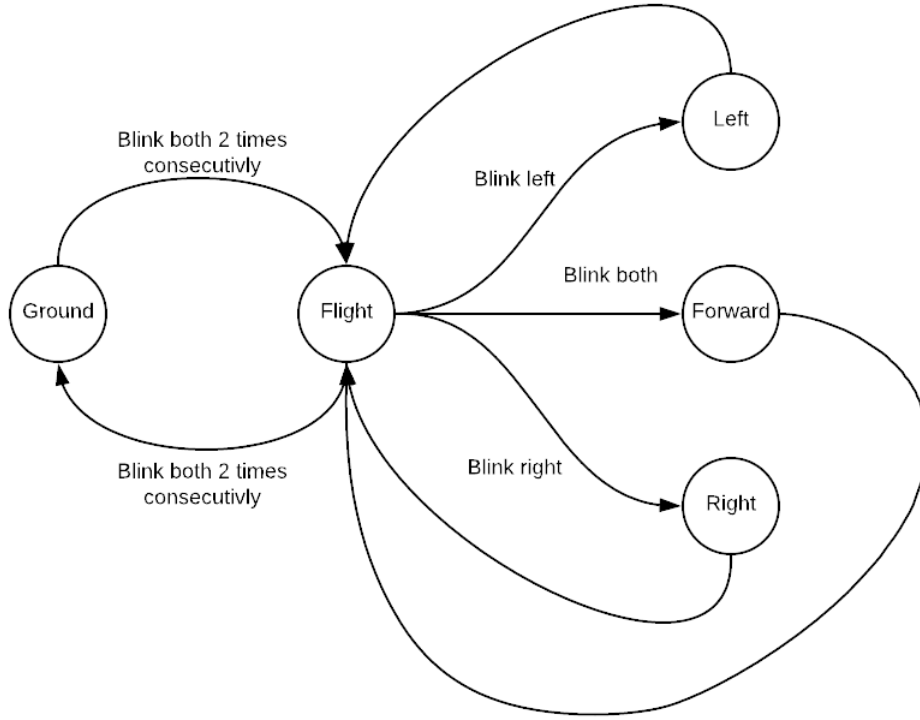


Figure 2.4: Drone controller final state machine

After several tests, the controller was slightly modified to consider the accuracy of the prediction. With this information, it is possible to discard an event with a low probability prediction. Even if the event might have been predicted right, it seems better to do nothing when the result of the classification is unsure. This part of the controller was not explored enough to present results, yet it might be interesting to study this parameter to avoid false prediction dues to artifacts. For the final design of the controller, the accuracy was checked with a simple threshold arbitrarily placed at 0.5.

The controller is contained in its own thread and will read the prediction Queue in a loop. Each event in the Queue will increment a counter that will be reset after a one second timeout.

Chapter 3

Results and Analysis

This chapter will present project results and their analysis. It is mainly a comparison between the accuracy of the classifier using different combinations. As explained in the chapter 2, two types of dataset, two features extraction methods and 5 machine learning algorithms were tested.

The comparison for this project was done between the features extraction method, and the machine learning algorithm was selected automatically by the software. In the first part, this comparison was made using the dataset with only one subject. It is only at the end of the project that the dataset with 5 subjects has been tested. The results will be presented in this order as it represents the evolution of this project.

The score used for comparison are the scikit-learn score obtained through k-fold cross-validation, the training score, and the score from the project's test script which uses the precision. The datasets used for the test script are composed of the same amount of samples as the training dataset and the same amount of subjects.

Even if the final design of the drone controller only uses blink from both eyes, the four classes of the dataset were used for this part.

3.1 Results with the one subject dataset

The features extraction methods were presented in 2.5, but the pre-processing of the signals (EMD) was also compared. The number of IMFs extracted from the signals has been tested from 0 which mean using the raw signals to 2 (2 IMFs + 1 residue). The results are presented in the tables 3.1 and 3.2.

The choice for the IMFs selection was arbitrary. The test without using the EMD was made just to verify that the use of EMD increases the accuracy of the classification. Nothing else was expected from this choice.

Limiting the study to the last IMF and the residue was done in order to use the EMD as a filter. As explained in 1.4, the EMD decomposes the signal from the highest frequency in the first IMF to the lowest one in the last. It can be noted that, in most of the applications, the

last IMFs are just algorithmic noise and do not provide any relevant information. However, for this project, the number of IMFs extracted was limited to two, which solves this problem. In this table, res represents the residue of the EMD.

Finally, the use of one IMF plus the residue was done to check the impact of the EMD as a filter for this application.

Method	Energies and fractal dimension			
IMFs used	without EMD	residue (2 IMFs)	2nd IMF and residue (2 IMFs)	all (1 IMFs)
Algorithm	RF	RF	SVM	RF
Training score	0.825	0.75	0.875	0.85
Test score	0.65	0.625	0.8	0.725

Table 3.1: Results for the one subject dataset with energies and fractal dimension features extraction

Method	Statistical values			
IMFs used	without EMD	residue (2 IMFs)	2nd IMF and residue (2 IMFs)	all (1 IMFs)
Algorithm	RF	RF	RF	DT
Training score	0.85	0.875	0.825	0.875
Test score	0.64	0.65	0.775	0.85

Table 3.2: Results for the one subject dataset with statistical features extraction

3.2 Results with the five subjects dataset

This dataset was created to study the impact of the dataset on the classification. The subjects were composed of 2 males and 3 females who recorded 3 repetitions per movement. 2 repetitions were used to build the training dataset and 1 for the test dataset. The age of the subjects was between 21 and 23 years old. Here, the test without the use of EMD and with only the residue were removed as the previous results showed that it was less efficient.

Method	Energies and fractal dimension		Statistical values	
IMFs used	2nd IMF and residue (2 IMFs)	all (1 IMFs)	2nd IMF and residue (2 IMFs)	all (1 IMFs)
Algorithm	KNN	KNN	DT	RF
Training score	0.75	0.85	0.8	0.95
Test score	0.8	0.825	0.775	0.975

Table 3.3: Results for the five subjects dataset

3.3 Results analysis

The previous sections presented results with the one subject dataset and five subjects dataset. Before comparing the datasets, the tables 3.1 and 3.2 will be analyzed together.

The tables 3.1 and 3.2 show a good comparison between the EMD parameters choice. The first thing which comes out is the low score from the case using only the residue and the raw signal. The scikit-learn score does not help differentiate cases as it is calculated based on the training dataset. Therefore the score difference can be explained by the changes from the training dataset to the test dataset. This is because the mean value of the signals or the noise can change between two recordings. When using the residues or the original signal, the number of features extracted is low (8 for the energy based features extraction and 16 for the statistics based one). Because of this low number of features, small changes have a strong impact on the classification. This is why these methods were not kept for the test with the second dataset.

As for the case using 2 IMFs and the case using all of them, they look similar. It is the same for the features extraction method used in both cases. As I discovered machine learning during this internship, I can not analyze the reason why one algorithm is better than the other one. It can still be noted that the random forest algorithm is most often the best one.

Even if the accuracy announced offline seems enough to perform real time classification, it was really difficult to get any concrete results. Most of the time, the "nothing" event was well predicted and the other ones were confused for each other. As it is uncommon to blink with only one eye, it is difficult to do it without impacting the other eye. Furthermore, the electrodes Fp1 and Fp2 are really close. This is why it is difficult to have an accurate prediction between those events. A solution could be to use more electrodes to base the prediction on more information. Moreover, the system did not work at all with a different subject from the one used in the dataset. This is what motivated the use of the second dataset.

The table 3.3 shows the results for the five subjects dataset. Here, the difference between the use of the IMF 2 plus the residue and using everything is remarkable. The use of all IMFs significantly increases the accuracy of the classification. This means that the use of the EMD as a filter is not interesting, and even though the first IMF might contain some noise, the

information contained inside is still relevant for the classification. Then, inside the cases using everything from the EMD, the statistic features extraction stands out with a score substantially higher than any other case. An explanation for this behavior is the difference between the signal from one subject to another. The number of features used with the statistic method is higher so these changes have a limited impact. Furthermore, the energies are sensitive to change from the potential more than the statistical values. With this, we can conclude that the use of statistics might be better for classification for several subjects. As only two features extraction methods were tested, better methods might exist to classify eye blink events in an EEG.

Comparing 3.1, 3.2 and the 3.3, it stands out that the use of several subjects in the dataset offers a better accuracy for the prediction. Using several subjects for the training allows for a better generalization of the prediction and reduces the risk of overfitting. Even with such an improvement, the use of the system in real time is not reliable and the "right" and "left" events are still confused. As I did not have enough time to work on this issue, I limited the classification to the events "nothing" and "both". The accuracy was then around 0.9 in real time and more than 10 people successfully managed to fly the drone during a presentation. A video summing up this presentation can be found at the following URL : <https://photos.app.goo.gl/XNuoZMikvvwzF9vp9>.

Chapter 4

Conclusion

One of the objectives was detecting eye blinks using EEG and EMD in order to control a drone. Due to time constraints, the research could not be finalized in this internship's allotted time. Nevertheless, the results obtained were promising as they show that this method can detect eye blinks, even though it is not possible yet to differentiate between right and left eye blink.

Indeed, the tests revealed that EEGs differ from one individual to the other, therefore, the best dataset for the training stage should contain various subjects. The resulting classifier would give better generalizations and be usable by a broader audience. Moreover, for this application, statistical features extraction proved to be the most reliable method. As for the use of the EMD, it significantly increases the accuracy of the classification. Further research is required in order to select the best IMFs. This project has shown that a pre-detection algorithm is required to perform realtime classification. A pre-detection algorithm has been proposed during this project. It uses normalization and binarization to detect events with a small amount of calculus. Furthermore, the use of the normalization could be considered as a pre-processing stage for the features extraction. Moreover, although several machine learning algorithms were tested, it can not be excluded that a more efficient one exists. For example, neural networks could not be tested due to a lack of sufficient data.

For the drone controller, in order to detect whether the drone has to take off or land, two consecutive eye blinks are required. This is done by waiting for one second after a blink in order to check for a second one. This causes the controlling experience to be delayed by that one second when trying to direct the drone on flight. Perhaps, take off and landing could be controlled by a much longer blink instead of two so that one short blink would prompt the system to react instantly. The intensity of the blink could also be used to differentiate actions. It must be kept in mind that this system is designed to simplify the control of a drone so these solutions have to be tested on several subjects to be validated.

The software interfaces were implemented in a console only. For general use, a graphical interface would simplify the dataset creation and modification. The real time part should be made more interactive by giving more information to the user, for example, when a blink is expected or has been detected.

In order to allow additional complex interaction, this study could be combined with eye movement detection such as the research of Adrian Ribe and Kristian Husevåg Krohn [9].

Bibliography

- [1] Sickit-learn. Supervised learning. http://scikit-learn.org/stable/supervised_learning.html#supervised-learning. Visited 16/10/2018.
- [2] L F Haas. Hans berger (1873–1941), richard caton (1842–1926), and electroencephalography. *Journal of Neurology, Neurosurgery & Psychiatry*, 74(1):9–9, 2003.
- [3] J. Satheesh Kumar and P. Bhuvaneswari. Analysis of electroencephalography (eeg) signals and its categorization—a study. *Procedia Engineering*, 38:2525 – 2536, 2012. INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.
- [4] Trans Cranial Technologies ltd. *10/20 System positioning*, 2012.
- [5] the free encyclopedia Wikipedia. 10–20 system (eeg). https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/21_electrodes_of_International_10-20_system_for_EEG.svg/1200px-21_electrodes_of_International_10-20_system_for_EEG.svg.png. Visited 16/10/2018.
- [6] J J Vidal. Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2(1):157–180, 1973.
- [7] Patrick Berg and Michael Scherg. Dipole models of eye movements and blinks. *Electroencephalography and Clinical Neurophysiology*, 79(1):36 – 44, 1991.
- [8] James C. Corby and Bert S. Kopell. Differential contributions of blinks and vertical eye movements as artifacts in eeg recording. *Psychophysiology*, 9(6):640–644.
- [9] Adrian Ribe and Kristian Husevåg Krohn. Eeg-based brain computer interface for controlling a drone. 2018.
- [10] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998.

- [11] Daoud Boutana, Messaoud Benidir, and B. Barkat. On the selection of intrinsic mode function in emd method: Application on heart sound signal. *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*, pages 1–5, 2010.
- [12] OpenBCI. Openbci_python. https://github.com/OpenBCI/OpenBCI_Python/, 2014.
- [13] Luis Alfredo Moctezuma and Marta Molinas. Eeg-based subjects identification based on biometrics of imagined speech using emd. 2018.
- [14] Sumit Kumar Ram, Marta Molinas, and Geir Kulia. Experience designing a bci for flying a drone based on the use of empirical mode decomposition, 11 2016.
- [15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [16] Playsheep. <http://www.playsheep.de/drone/>. Accessed: 09/08/2018.
- [17] M Abo-Zahhad, Sabah Ahmed, and Sherif Nagib Seha. A new eeg acquisition protocol for biometric identification using eye blinking signals. *International Journal of Intelligent Systems and Applications (IJISA)*, 07:48–54, 05 2015.