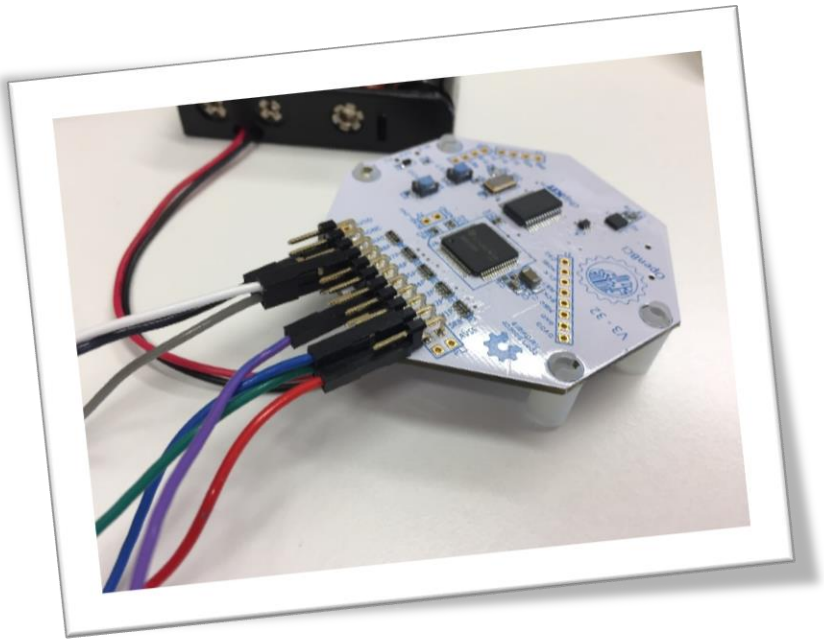
Victor CHARNET

Polytech Lille – 5ème année

Informatique Microélectronique Automatique

Développement d'une Interface Cerveau-Ordinateur

Projet de fin d'études



Rapport final – Février 2017

Tutrice : LECOCQ Claudine

Responsable laboratoire : CABESTAING François

Remerciements

Je tiens tout particulièrement à remercier :

- **Madame Lecocq**, tutrice de ce projet pour son accompagnement, ses conseils et son aide très précieuse lors de ces premiers mois de projet.

- **Monsieur Cabestaing**, responsable du pôle BCI du laboratoire CRIStAL, pour avoir accepté de lancer ce projet et d'en avoir défini les objectifs.

- **Monsieur Duprès**, doctorant au pôle BCI du laboratoire CRIStAL pour le temps qu'il a su prendre pour m'expliquer le principe de sa thèse et le fonctionnement de son système.

- **Messieurs Casier, Claverie, Verdonck** qui ont accepté de tester régulièrement mes applications.

- **Monsieur Engels**, pour avoir tourné et monté la vidéo du projet.

Table des matières

Introduction

Ce projet est réalisé dans le cadre des projets de fin d'études de cinquième année à l'école d'ingénieur Polytech Lille en formation IMA (informatique, Microélectronique, Automatique).

Le sujet est développé en partenariat avec le pôle BCI faisant parti du groupe thématique Interaction et Intelligence Collective du laboratoire CRIStAL (Centre de Recherche en Informatique, Signal et Automatique de Lille) se situant à l'Université Lille 1, dirigé par François Cabestaing.

Le lien entre le laboratoire est assuré par la tutrice du projet, Claudine Lecocq, maître de conférences à Polytech Lille.

L'objectif principal de ce projet est de développer une Interface Cerveau-Ordinateur à l'aide d'un kit Recherche & Développement open source, du nom d'OpenBCI. Le principe d'une Interface Cerveau-Ordinateur est de mesurer des signaux physiologiques d'une personne, qui peuvent être des signaux cérébraux ou musculaires, de les traiter et d'effectuer une classification des caractéristiques obtenues dans le but de contrôler et piloter un système, qui peut être physique (fauteuil roulant, drone, bras robot…) ou logiciel (P300 dont le fonctionnement est expliqué dans la partie I.b).

Il est important de préciser que les BCI sont développées principalement pour des personnes lourdement handicapées au niveau moteur pour qu'elles puissent communiquer ou interagir avec leur environnement par la pensée ou grâce à leur motricité restante

Ce rapport final présentera le contexte du projet et son cahier des charges, le déroulement du travail et les applications qui ont été développées et testées. Il sera conclu par une synthèse de performances, une présentation des perspectives du projet ainsi que d'un bilan de connaissances & compétences.

Lexique

BCI : Brain-Computer Interface : Abréviation courante utilisée pour Interface Cerveau-Ordinateur.

EEG : Électroencéphalographie : Méthode d'exploration cérébrale qui mesure l'activité électrique du cerveau par des électrodes placées sur le cuir chevelu.

EMG : Électromyogramme : Examen qui permet d'enregistrer l'activité spontanée d'un muscle ou d'un nerf.

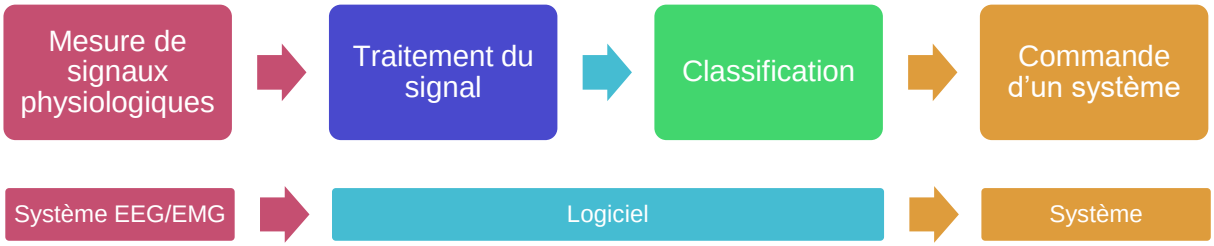
ECG : Électrocardiographie : Représentation graphique de l'activité électrique du cœur.

GUI : Graphical User Interface : Logiciel doté d'une interface graphique.

I. Contexte du sujet

a) Principe des BCI

L'architecture classique d'une BCI est la suivante :



Sur le premier graphique, on retrouve l'analyse fonctionnelle du système. Chaque bloc représente une fonction principale réalisée par le système. Il s'agit dans un premier temps de mesurer les signaux physiologiques de l'utilisateur. Il faut ensuite réaliser une opération de traitement de signal, où les données mesurées seront principalement filtrées dans le but d'obtenir des caractéristiques. Une étape de classification permet ensuite de classer les valeurs obtenues des caractéristiques. Enfin, les données, sous la forme de consignes (signaux logiques par exemple), sont transmises à un système.

Le second graphique correspond à l'architecture matérielle d'une BCI. Chaque bloc correspond donc au système nécessaire pour réaliser la fonction correspondante au bloc supérieur. Ainsi, la mesure des signaux physiologiques est assurée par un système EEG et/ou EMG (OpenBCI par exemple). Le traitement du signal et la classification sont réalisés par une partie logicielle (OpenVibe, le GUI d'OpenBCI, Matlab), puis le système à commander, qui peut être physique (fauteuil roulant, drone, etc..) ou logiciel (P300 dont le fonctionnement est expliqué dans la partie I.b).

a.1) Acquisition

Dans une interface cerveau-ordinateur, il est capital d'analyser les ondes cérébrales, mais il est également intéressant d'étudier les signaux issus des muscles. En effet, certains patients disposent encore d'une motricité sur certains de leurs membres, il est important de l'exploiter car il est plus facile de détecter un signal venant d'un muscle que du cerveau.

Il existe différents moyens de réaliser de l'acquisition de signaux cérébraux. La méthode la plus classique est la méthode dite « non-invasive ». Cela consiste à installer un système EEG muni d'électrodes sur la surface du crâne. Ce dispositif est relativement simple à mettre en œuvre et est le plus généralement utilisé. Néanmoins, les signaux sont très bruités car les potentiels mesurés sont extrêmement faibles (généralement inférieur à la dizaine de micro volt). En effet, le potentiel électrique mesuré par les électrodes d'un système EEG est lié à l'activité électrique des neurones situés dans le cerveau lorsqu'ils sont excités. On appelle ce phénomène : « Potentiel d'action ». L'idéal serait de pouvoir mesurer le potentiel d'action et de repos de chaque neurone présent, mais c'est technologiquement impossible en méthode « non-invasive ». Les systèmes EEG actuels mesurent donc l'activité électrique d'une population de neurones dans des zones spécifiques du cerveau (voir paragraphe suivant). Les BCI actuelles se basent donc sur une mesure globale d'activité électrique et ne permettent donc qu'une interprétation grossière de l'activité cérébrale, rendant ainsi difficile la commande

d'un système très complexe. C'est pourquoi les BCI se contentent d'environ 3 à 4 opérations simples au maximum sur les systèmes à commander.

Il n'est donc pas possible de mesurer l'activité électrique de chaque neurone, c'est pourquoi les systèmes EEG mesurent l'activité d'une population de neurones. Néanmoins il est connu que le cerveau est décomposé en différentes zones, « cortex » qui sont dédiées aux très nombreuses fonctions de l'Être Humain (Figure 1 : Représentation des différents cortex**Erreur ! Source du renvoi introuvable.**). On retrouve ainsi le cortex moteur, regroupant les fonctions appartenant aux mouvements du corps. Il existe également par exemple le cortex visuel, se situant à l'arrière du cerveau, dédié aux fonctions visuelles.

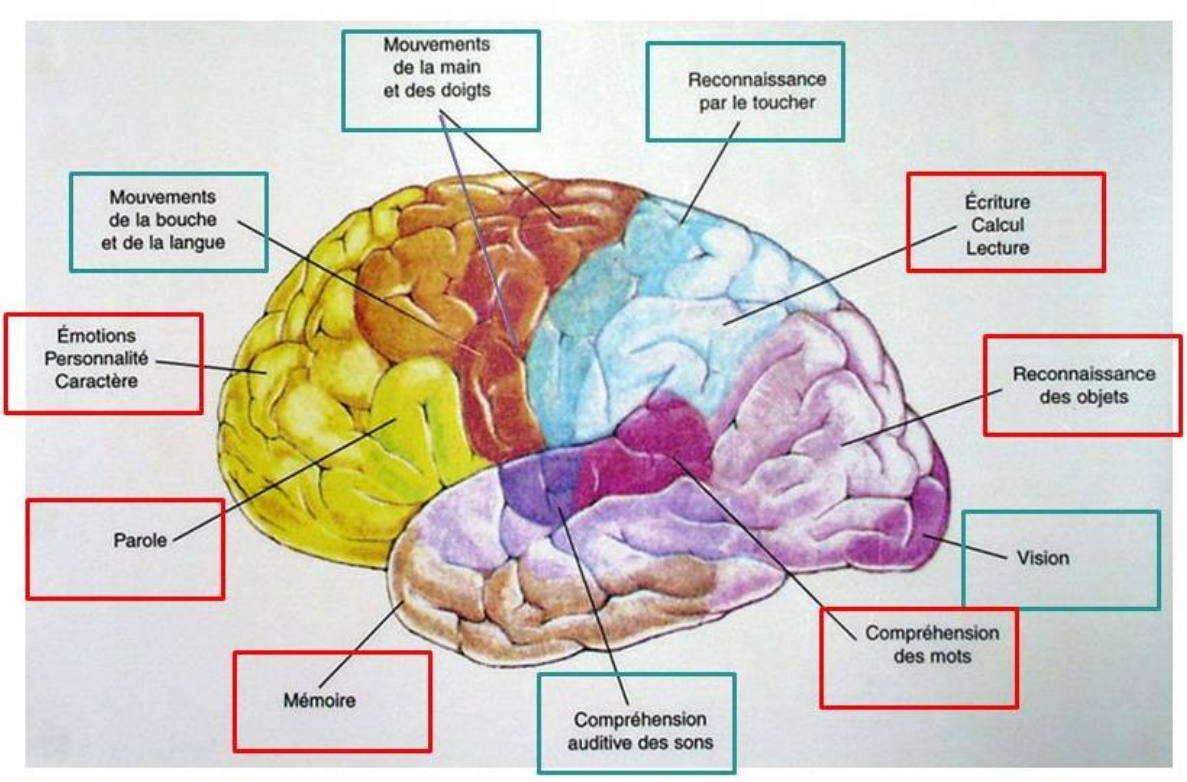


Figure 1 : Représentation des différents cortex

Afin de pouvoir mesurer correctement l'activité cérébrale dans ces zones, une méthode reconnue internationalement propose une disposition optimale des électrodes d'un système EEG. Il s'agit du « système 10-20 » et permet de normaliser la répartition des électrodes sur la surface de la tête d'un patient. (Figure 2 : Système 10-20)

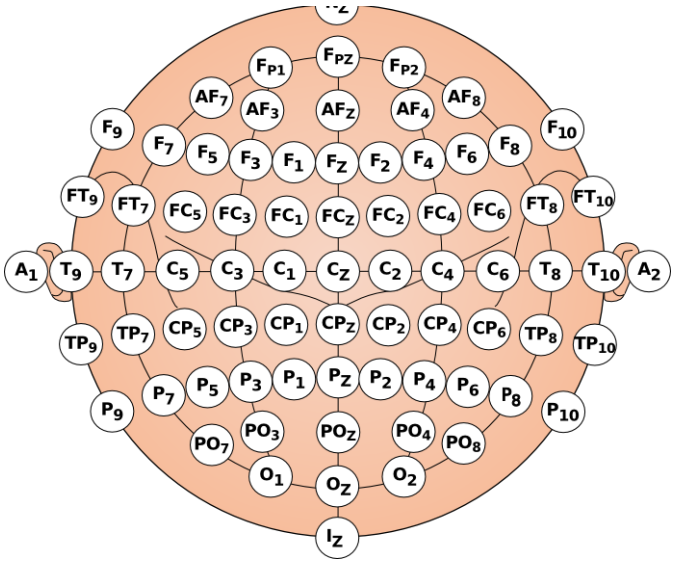


Figure 2 : Système 10-20

L'acquisition des ondes cérébrales peut également être réalisée par une méthode semi-invasive. Une grille d'électrodes est installée à la surface du cortex cérébral. Le procédé permet d'obtenir des signaux beaucoup moins bruités, mais nécessite une intervention chirurgicale, il est donc utilisé en association avec des traitement thérapeutiques.

La méthode pour obtenir les signaux de la meilleure qualité est la méthode invasive. Lors d'une opération, des électrodes vont être placées directement dans le cerveau. Les signaux sont alors très peu bruités, très fiables et très précis car on mesure l'activité d'une synapse. Néanmoins, la durée de vie des électrodes dans le milieu cérébral est très courte et leur installation provoque de graves séquelles irréversibles au cerveau.

La mesure d'activité musculaire s'effectue en plaçant une électrode sur le muscle à étudier et une électrode de référence à un endroit où est présent le rythme cardiaque (poignet ou tempe par exemple). Au repos le système mesure donc l'ECG, et lorsque le muscle est contracté, le système mesure l'ECG et l'EMG. Cela se traduit par une forte augmentation de la fréquence du signal.

a.2) Traitement du signal et classification

L'étape de traitement du signal est essentielle dans le bon fonctionnement d'une BCI. Elle permet d'extraire ce que l'on appelle des « caractéristiques ». L'objectif est dans un premier temps d'éliminer ce que l'on appelle les artefacts (perturbations du signal EEG), qui peuvent être de type électrique, comme la présence du « 50Hz » issu de l'alimentation électrique du secteur, ou de type physiologique comme les clignements des yeux ou bien le pouls. C'est une étape de prétraitement qui utilise généralement des filtres. Les plus courants sont :

- **Filtrage spatial :** On utilise les informations des électrodes voisines. Un filtre spatial commun est le Laplacien. On va soustraire à une électrode la valeur moyenne des électrodes voisines. Si on se place dans la configuration de l'illustration ci-dessous, l'expression de la valeur de l'électrode centrale Cz après un filtrage Laplacien est la suivante :

$$Cz_{Laplacien} = Cz - \frac{Fz + Pz + C4 + C3}{4}$$

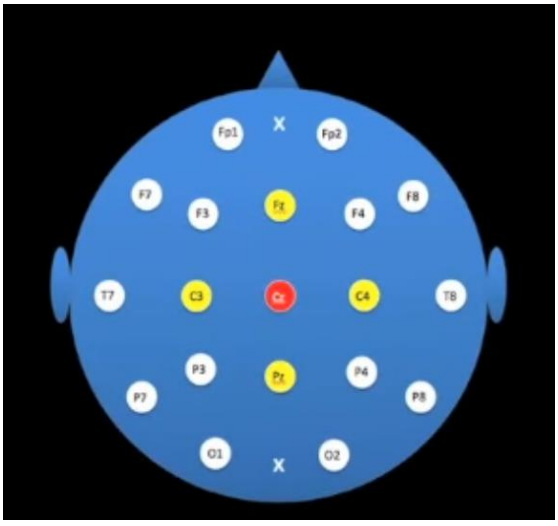


Figure 3 : Illustration du filtrage Laplacien sur un 10-20 system

- **Filtrage fréquentiel :** Le filtrage fréquentiel permet de sélectionner les bandes de fréquence utiles en fonction du paradigme de l'interface cerveau-ordinateur. En effet on peut classer les activités électriques cérébrales rythmiques en fonction de leur fréquence. Par exemple les ondes α sont généralement comprises entre 9 et 13Hz, elles correspondent à un état physiologique de repos et se manifestent généralement à la fermeture des yeux. L'analyse fréquentielle permet ainsi de remarquer la présence ou non de ces ondes. Le graphique ci-dessous (Figure 4 : Mise en évidence des ondes alpha sur une FFT) représente mes propres ondes alpha mesurées lors d'une acquisition sur le GUI OpenBCI.

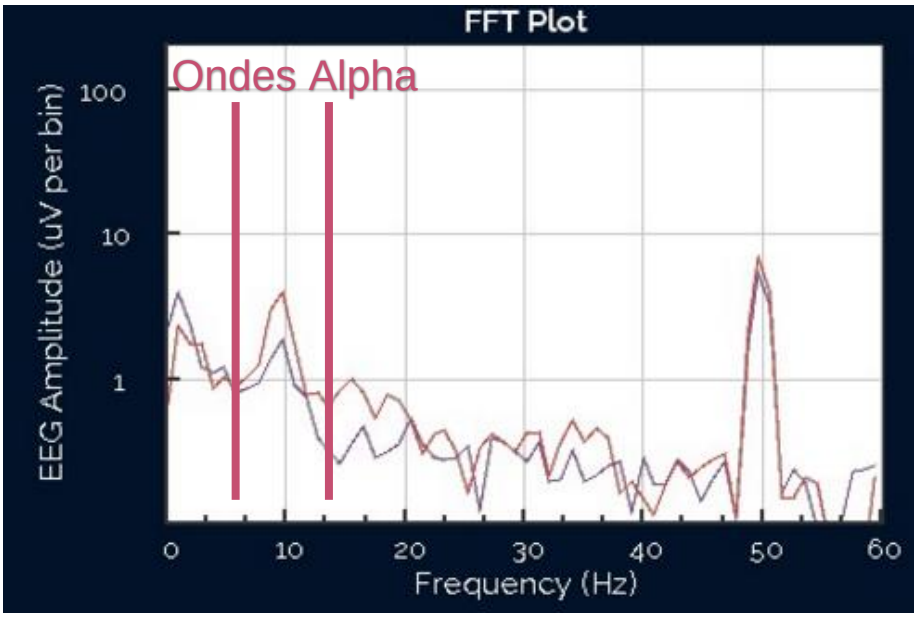


Figure 4 : Mise en évidence des ondes alpha sur une FFT

L'étape de classification a pour but d'attribuer une classe aux caractéristiques extraites lors de l'étape de traitement du signal. Cette classe va représenter le type de la tâche mentale effectuée par l'utilisateur de l'interface, par exemple : bouger la main droite. La classification est réalisée à l'aide d'algorithmes appelés « classifieurs ». Les classifieurs apprennent à identifier la classe des caractéristiques à l'aide d'un ensemble d'apprentissage. Ces ensembles sont composés de caractéristiques auxquelles sont attribuées les classes d'appartenance. *[Réf. 1]* Les classifieurs les plus utilisés sont les suivants :

- Classifieur LDA (Analyse linéaire discriminante) : Il s'agit d'un classifieur linéaire dont les coefficients définissent un hyperplan, permettant de discriminer les signaux EEG appartenant à deux classes. *[Réf. 2]*

- Réseaux de neurones : chaque neurone effectue une combinaison linéaire et permet d'attribuer à chaque neurone une classe en fonction de la combinaison. Ce type de classifieur permet notamment de séparer des classes non séparables par un hyperplan. *[Réf. 3]*

- Séparateurs à Vastes Marges (SVM) : Il s'agit en réalité d'un classifieur linéaire, mais qui permet de choisir l'hyperplan le plus éloigné des nuages de points des deux

classes. Cet hyperplan est alors optimal en termes de généralisation. Ce type de classifieur permet d'obtenir des pourcentages de réussite de l'ordre de 80% à 96,8% pour une classification main gauche / main droite. *[Réf. 3]*

Cette étape de classification est très souvent rattachée à une étape de machine learning, notamment dans le cas des interfaces asynchrones.

Il est également important de préciser que même avec un faible nombre de classes, le taux de classification correcte d'une interface n'atteint que très rarement les 100%.

b) Etat de l'art

b.1) Applications actuelles

En fonction du traitement accordé aux signaux d'entrée, les systèmes BCI peuvent être classés en tant que systèmes synchrones ou systèmes asynchrones. Les systèmes synchrones vont analyser les signaux cérébraux durant des fenêtres temporelles précises, basées sur des stimulations. Les signaux en dehors de ces fenêtres temporelles sont ignorés. L'utilisateur ne peut donc envoyer des commandes que lors de durées spécifiques déterminées par le système. Les systèmes asynchrones analysent en continu les signaux cérébraux, peu importe quand l'utilisateur agit. Ce système permet donc une interaction plus naturelle entre l'homme et le système. Cependant les systèmes asynchrones sont plus complexes à mettre en œuvre. *[Réf. 4]*

Actuellement, la plupart des interface cerveau-ordinateur sont basées sur des signaux relativement simples à détecter, traiter et classifier. Les applications les plus courantes sont les suivantes :

- **Utilisation d'un muscle (EMG) :** C'est un système relativement simple à mettre en place, avec deux électrodes, une sur le muscle et une de référence, il est possible de mesurer l'activité du muscle. Ce système est asynchrone.

- **Détection des ondes alpha :** Comme expliqué dans la partie I a.2, lorsque nous fermons les yeux et que nous sommes en état de repos, nos ondes cérébrales vont se mettre à osciller sur une gamme de fréquences comprise entre environ 9Hz et 13Hz. En recherchant une augmentation de la puissance de signal dans cette bande de fréquence on peut donc détecter la présence d'ondes alpha.

- **SSVEP (Steady-State Visual Evoked Potential) :** Lorsque nous regardons une source lumineuse clignoter (par exemple matrice de LED) à une certaine fréquence (de l'ordre de la dizaine de hertz), il est possible de retrouver cette fréquence en observant les ondes cérébrales de la zone située au niveau du lobe occipital. L'inconvénient de ce paradigme est qu'il s'agit d'une réaction non volontaire de l'utilisateur. L'avantage est que cela ne nécessite pas de phase d'apprentissage.

- **Moyen de communication, « Clavier virtuel » :** Il s'agit du P300 speller, qui est couramment utilisé dans le domaine des BCI. Il s'agit d'un système synchrone qui se présente sous la forme d'un tableau où dans chaque case se trouve une lettre. Si l'utilisateur veut écrire « bonjour » par exemple. Il devra se concentrer sur la lettre B dans un premier temps. Le système va ensuite mettre en surbrillance alternativement les différentes colonnes puis les différentes lignes. En se concentrant sur la lettre

voulue, à chaque fois que la bonne colonne ou la bonne ligne où se trouve la lettre sera en surbrillance, il y a aura une augmentation du potentiel électrique sur l'électrode Cz du patient 300ms après le stimulus sous réserve que l'utilisateur exerce à ce moment-là une activité cérébrale cognitive.

- **Imagerie motrice, détection des mouvements de pieds :** Il est possible en analysant les ondes cérébrales de retrouver l'activité du corps humain. Lorsque l'utilisateur va bouger les pieds, une diminution de la puissance dans la fréquence β (18-24Hz) va être mesurée en EEG. Cette diminution de la puissance est appelée ERD (Event Related Desynchronisation). Lorsque le mouvement est terminé, une augmentation de la puissance dans cette gamme de fréquence peut être mesurée, on l'appelle ERS (Event Related Synchronisation). Cette phase d'augmentation est appelée « Rebond Beta ». En mesurant la puissance dans cette gamme de fréquence, il est possible de détecter si l'utilisateur a réalisé un mouvement des pieds (réel ou imaginé). *[Réf. 5]* L'INRIA a développé une application qui consiste à faire décoller virtuellement un vaisseau spatial en se basant sur ce paradigme.

- **Imagerie motrice, discrimination main gauche / main droite :** En effectuant une phase d'apprentissage, il est possible d'entrainer un classifieur pour déterminer si l'utilisateur bouge sa main droite ou sa main gauche. La plupart des applications actuelles ont un taux de réussite d'environ 80% sur ce type d'applications.

b.2) Limites

Il est important de contredire certaines idées reçues sur l'étude des signaux cérébraux, il est impossible de lire dans les pensées des gens avec de tels systèmes, les rêves, ou alors de détecter les mensonges. Les moyens d'acquisition utilisent une détection de l'activité électrique globale, par zones. On ne peut donc distinguer et utiliser que des fonctions simples.

L'acquisition de signaux se fait actuellement majoritairement en surface, le dispositif est simple mais les signaux sont extrêmement bruités. Les acquisitions invasives ou semi-invasives sont très difficiles à mettre en place et sont réservées à des patients atteints de pathologies lourdes étant donné les séquelles irréversibles que peuvent causer des électrodes dans la masse cérébrale.

c) Objectifs du projet

Les objectifs principaux du projet sont issus d'une collaboration entre le laboratoire CRIStAL (représenté par Mr Cabestaing), Mme Lecocq et moi-même. Il fallait choisir des objectifs réalisables dans le temps dédié à ce projet de fin d'études, et qui aient un réel apport pour le laboratoire.

- **Analyse des possibilités du kit OpenBCI :** L'intérêt premier du projet est de tester les possibilités du kit OpenBCI à travers plusieurs applications classiques des BCI.

- **Documentation technique pour utilisation OpenBCI :** il s'agit là de produire une documentation précise des câblages et des caractéristiques du kit afin qu'une personne l'utilisant puisse rapidement trouver les informations dont elle a besoin.

- **Mise en œuvre du système d'acquisition OpenBCI sur OpenVibe** : OpenVibe est le logiciel majoritairement utilisé par le laboratoire CRIStAL. Il est donc très important de rendre le plus rapidement possible le matériel OpenBCI compatible avec ce logiciel.

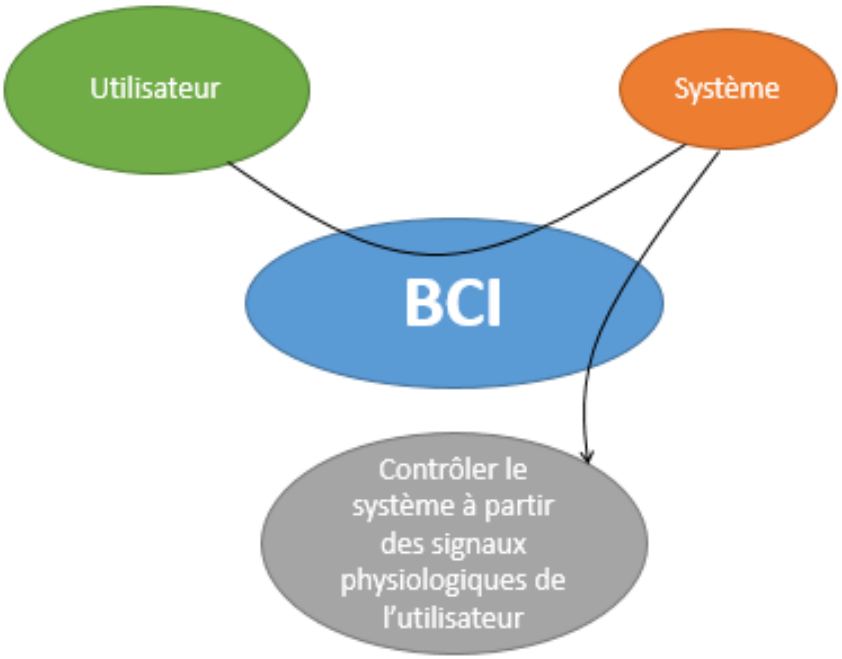
- **Intégration du kit dans une Interface Cerveau-Ordinateur :** L'objectif à terme de ce projet est de pouvoir mettre en œuvre la commande d'un système, qu'il soit logiciel ou physique.

d) Modélisation du cahier des charges

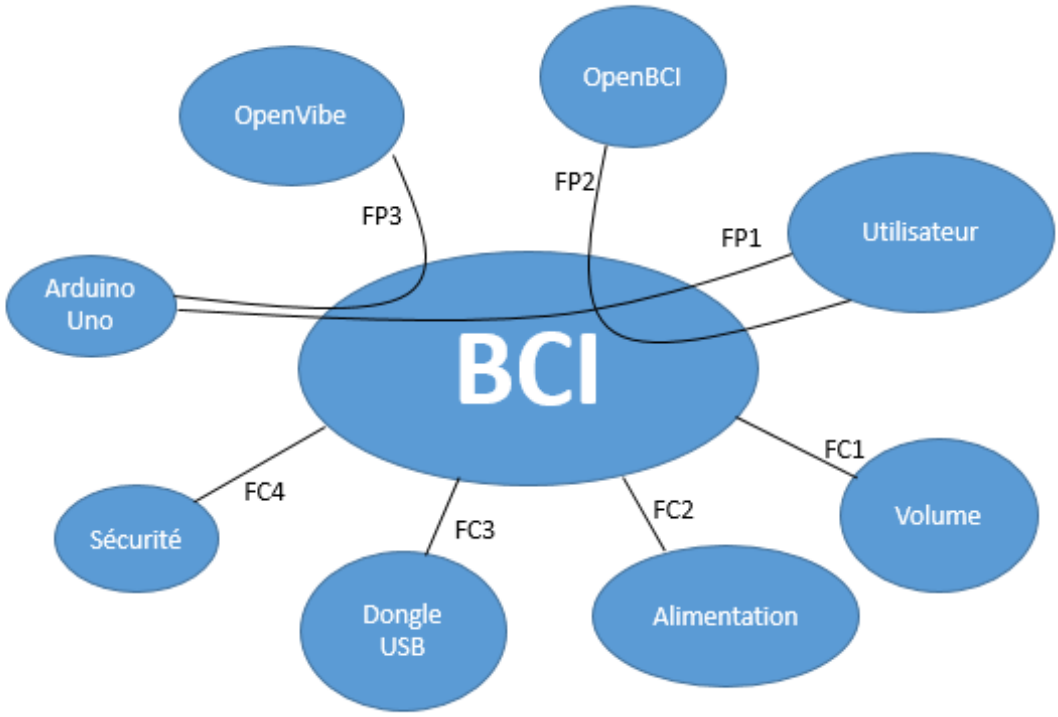
Dans le cadre de ce projet, il est important de savoir modéliser le cahier des charges à l'aide outils méthodologiques.

L'analyse bête à cornes, permet de mettre en avant l'expression du besoin :

Une BCI a pour but de rendre service à un utilisateur en agissant sur un système à commander. Le but est de contrôler le système à partir des signaux physiologiques de l'utilisateur.



Le diagramme pieuvre permet de mettre en avant les fonctions principales et contraintes du projet :



Fonctions Principales :

- **FP1 :** Permettre à l'utilisateur de contrôler un système physique
- **FP2 :** Effectuer la mesure des signaux physiologiques de l'utilisateur
- **FP3 :** Assurer la communication entre le BCI et le système physique

Fonctions Contraintes :

- **FC1 :** Le système ne doit pas être encombrant
- **FC2 :** L'alimentation doit être réalisée par des piles pour éviter le 50Hz du réseau
- **FC3 :** La communication entre le BCI et le PC doit être en Bluetooth et non en filaire
- **FC4 :** Le système ne doit pas être dangereux pour l'utilisateur

II. Déroulement du projet

a) Gestion du projet

Dans le cadre du projet, plusieurs outils ont été mis en place pour s'assurer du bon déroulement de celui-ci :

a.1 Plan d'action & livrables associés

Une fois les objectifs définis, il était important de définir le plan d'action à suivre dans ce projet et de définir pour chacune des tâches un livrable.

Tâche	Livrable associé
Documentation sur les BCI & gestion du projet	
Documentation et état de l'art des BCI	Diaporama de connaissances sur le sujet BCI
Utilisation d'outils de gestion de projet	Documents (Analyse des risques, indicateurs etc..)
Mise en œuvre du kit OpenBCI	
Etude du kit OpenBCI	Documentation technique du kit
Prise en main des logiciels OpenBCI et OpenVibe	Documentation sur l'utilisation d'OpenBCI et OpenVibe
Mise en place d'une procédure de câblage	Document décrivant les étapes pour la mise en place du système d'acquisition
Assurer la compatibilité du kit avec OpenVibe	Document décrivant les étapes pour configurer OpenBCI avec OpenVibe
Analyse des performances d'OpenBCI	
Sélection des applications à tester	Liste d'applications et paradigmes
Choix des câblages pour chaque application	Document précisant les câblages et montages utilisés
Elaboration de scénarios OpenVibe	Scénarios OpenVibe
Synthèse des performances	Document récapitulatif
Intégration d'OpenBCI dans une interface	
Développement d'une BCI de démonstration	Interface simple opérationnelle
Permettre une communication entre l'interface cerveau-ordinateur et un système physique	Mise en place d'une liaison série

a.2) Indicateurs de résultats

Afin de s'assurer du bon déroulement du projet, il a fallu mettre en place des indicateurs de résultats permettant de mesurer la pertinence et l'avancée de chaque tâche.

Indicateurs de résultat liés aux actions

Documentation sur les BCI & gestion du projet

Temps : Gain de temps pour les prochains utilisateurs du kit OpenBCI. L'analyse des risques permet d'éviter d'éventuelles pertes de temps

Sécurité : Informations concernant les BCI et notamment à usage médical

Qualité : Informations regroupées avec sources. Utilisation des outils de gestion de projet

Profitabilité : Fiabilité, pertinence et justification du projet. Organisation réutilisable

Mise en œuvre du kit OpenBCI

Temps : Gain de temps pour les prochains utilisateurs du kit OpenBCI, que ce soit vis-à-vis de la document technique (gain estimé à 2 semaines) mais également au niveau de l'installation du kit (gain estimé à 1 mois)

Sécurité : Informations concernant le matériel et les manipulations (câblage, manipulations à éviter etc…) Sécurité du système vis-à-vis des patients

Qualité : Garantir la fiabilité de la mesure et obtenir de mesures exploitables sur d'autres logiciels (MATLAB). Identifier les mesures représentatives et exploitables

Profitabilité : Mise en place d'une procédure de mesure. Système compatible avec OpenVibe

Analyse des performances d'OpenBCI

Temps : Document exploitable par les prochains utilisateurs du kit OpenBCI (gain estimé à 1 mois).

Sécurité : Prise de conscience des avantages et inconvénients d'OpenBCI en fonction des différentes applications

Qualité : Document exploitable par les prochains utilisateurs. Mise en place de scénarios réutilisables

Profitabilité : Tableau de comparaison des différentes applications

Intégration d'OpenBCI dans une interface

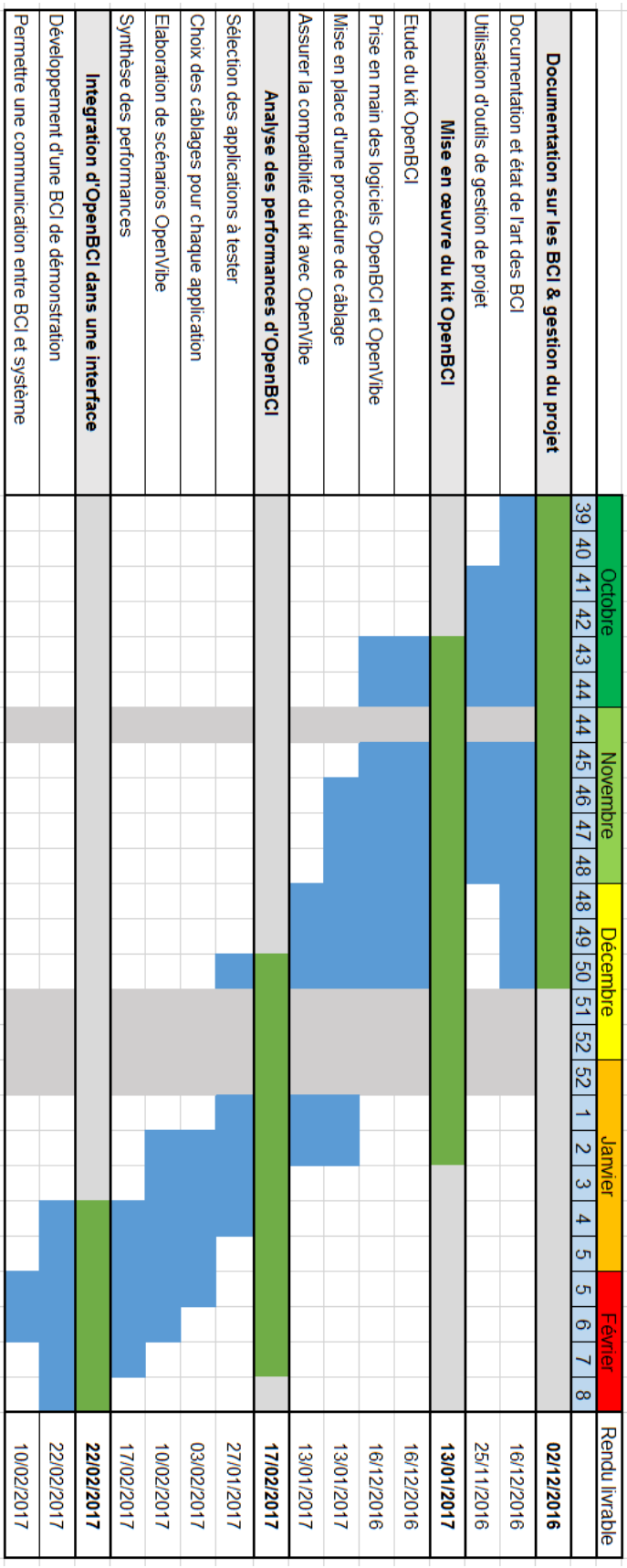
Temps : Gain de temps pour le laboratoire (gain estimé à 4 mois). Mise en place d'un protocole et projet réutilisable

Sécurité : Informations disponibles pour le laboratoire

Qualité : Fichiers exploitables. Absence de bugs sur les réalisations. Système fiable et utilisable par le maximum de personnes (avec la campagne de tests)

Profitabilité : Possibilité de présenter une BCI de démonstration

a.3) Planning du projet



a.4) Analyse des risques du projet

1) Risques techniques

- Défaillance ou détérioration du matériel (kit OpenBCI)
    - Probabilité : Extrêmement rare
    - Effet : Défaut, Destruction ou système inopérant
    - Détection : Immédiate
    - Protection : Matériel rangé dans casier fermé / Manipulations « délicates »
    - Réaction : Réparation ou retour
- Manque de ressources disponibles (documentations, études etc…)
    - Probabilité : Rare
    - Effet : Délai ou annulation d'une tâche, voir action
    - Détection : Quand la ressource est nécessaire
    - Protection : Plan d'action prévu à l'avance, prévoir une aide extérieure
    - Réaction : Modification du planning et du plan d'action, utilisation de l'aide extérieure
- Casque MarkIV non commandable dans les délais
    - Probabilité : Fortement envisageable
    - Effet : Délai ou modification d'une tâche
    - Détection : Prise de décision
    - Protection : Prévoir une solution de secours (impression de casque sur un autre modèle ? Rendre opérationnel sur bonnet CRIStAL ?)
    - Réaction : Mettre en place la solution de secours choisie

2) Risques humains

- Compétence non maîtrisée
    - Probabilité : Rare
    - Effet : Délai d'une tâche
    - Détection : Quand la compétence est nécessaire
    - Protection : Anticipation des tâches et du plan d'action, prévoir une aide extérieure
    - Réaction : Modification du planning et du plan d'action, utilisation de l'aide extérieure
- Créneaux disponibles pour effectuer des mesures
    - Probabilité : Occasionnelle
    - Effet : Délai d'une tâche
    - Détection : Chaque semaine lors du bilan
    - Protection : Anticiper les séances de projet et planifier les séances de mesures

- Réaction : Mobiliser les ressources et améliorer l'efficacité du processus de mesures

3) Risques liés à la sécurité

- Système peu fiable pour la sécurité du patient

  - Probabilité : Extrêmement rare
  - Effet : Danger pour le patient
  - Détection : A la manipulation
  - Protection : Etude de la documentation technique et évaluation des risques (mise en place d'un protocole « d'urgence ») Se référer aux normes médicales ?
  - Réaction : Application du protocole « d'urgence »

4) Risques sur les délais

- Mauvaise estimation de la durée des tâches et actions

  - Probabilité : Fréquente
  - Effet : Délai ou modification d'une tâche, voir action
  - Détection : Lors des bilans projets
  - Protection : Anticiper la charge de travail, efficacité
  - Réaction : Modification du planning et du plan d'action. Mobiliser les ressources

b) Présentation du kit OpenBCI

OpenBCI est un système d'acquisition des signaux EEG à bas coût. C'est une plateforme open-source qui donne accès très facilement à la mesure des signaux cérébraux. Il est développé en open source dans le but d'accélérer l'innovation des neurosciences. Le kit R&D en ma possession est composé d'un lot d'électrodes (type goldcup, passives) et d'une carte d'acquisition basée sur un convertisseur analogique/numérique Texas Instruments ADS1299 à faible bruit. Ce convertisseur est conçu pour travailler sur la mesure de faibles signaux EEG (de l'ordre du micro volt). La carte de base permet de fonctionner sur 8 canaux et donc d'utiliser 8 électrodes.

b.1) Etude du kit

Ce document est une synthèse d'informations techniques concernant le kit de développement « Cyton Biosensing Board 8-channel » développé par OpenBCI.

Référence : http://shop.openbci.com/collections/frontpage/products/openbci-16-channel-r-d-kit?variant=785215991

Il est important de rappeler que ce kit n'est pas du matériel médical et ne doit donc pas être utilisé dans un but de diagnostic médical.

Fonctionnalités du kit

Le kit Cyclon OpenBCI est utilisé pour la mesure d'EEG ou d'EMG.

Spécifications techniques

OpenBCI 32bit Board :

- 8 canaux d'entrée, différentiels, haut gain et faible bruit
- Compatible avec des électrodes passives et actives
- Convertisseur Analogique/Numérique « Texas Instruments ADS1299 ADC »
- Microcontrôleur « PIC32MX250F128B »
- Communication Low Power Bluetooth « RFD22301 »
- Accéléromètre « LIS3DH »
- Port pour cartes micro SD
- 5 pin GPIO, dont 3 analogiques

OpenBCI Dongle USB

- Communication Low Power Bluetooth « RFD22301 »
- Convertisseur USB/série depuis FTDI « FT231X »

Module Daisy OpenBCI

- 8 canaux d'entrée supplémentaire, différentiels, haut gain et faible bruit
- Compatible avec des électrodes passives et actives
- Convertisseur Analogique/Numérique « Texas Instruments ADS1299 ADC »

Câblages pour les mesures.

Le kit Cyton OpenBCI dispose de 8 canaux d'entrée qui fonctionnent en mode différentiel. Chaque canal peut utiliser deux broches N et P comme le représente le schéma ci-dessous.

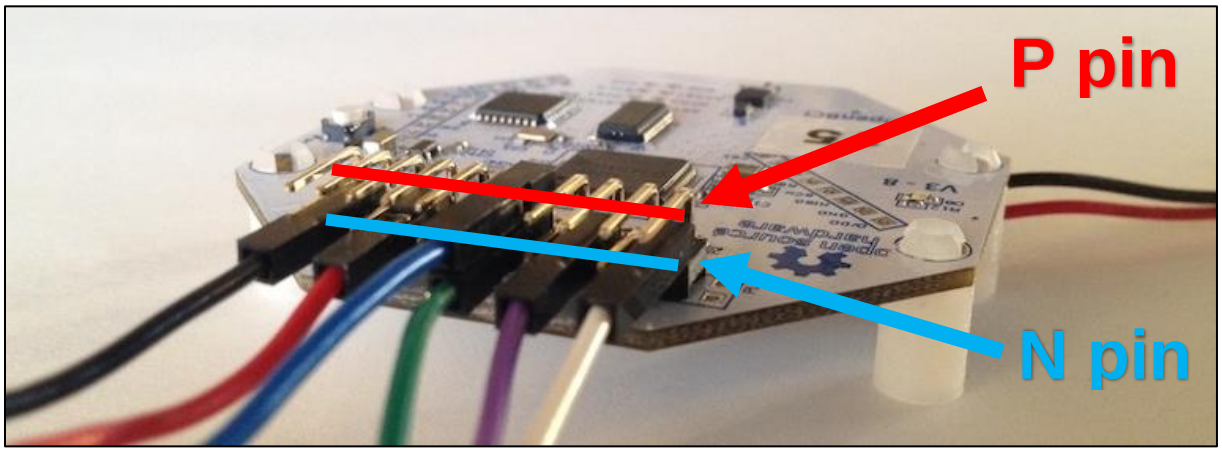


Figure 5 : Pins P & N de la carte Cyton OpenBCI

La mesure de potentiel électrique s'effectue en différentiel, mais en fonction des applications, les câblages peuvent être différents.

Câblage N_P

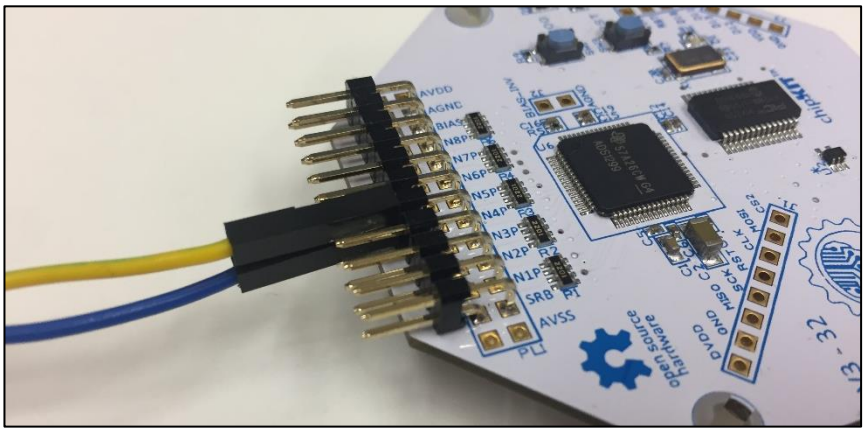


Figure 6 : câblage N_P

Dans cet exemple, le câble jaune est sur la pin 4P et le câble bleu sur la pin 4N. Sur ce montage, la mesure du potentiel est effectuée sur la pin 4N et la broche 4P est la référence. Dans ce cas, la valeur mesurée sur le canal 4 est la suivante :

$$V_{N4P} = V_{4N} - V_{4P}$$

Avec V : tension mesurée sur une PIN

Montage classique à une électrode

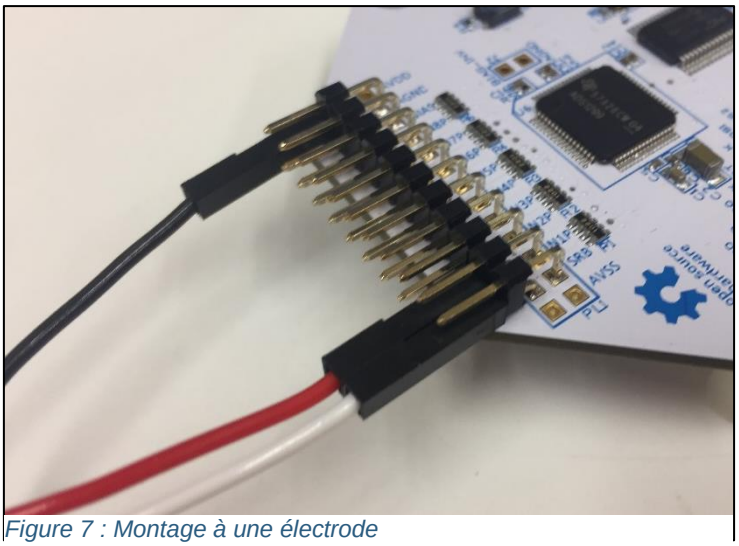


Figure 7 : Montage à une électrode

Voici une proposition de câblage classique pour la mesure des EEG avec une électrode.

Sur ce montage, nous effectuons la mesure sur une électrode placée sur la pin N1 (câble rouge). Le câble blanc est branché sur la pin SRB2, il servira de référence pour la mesure et son électrode associée devra généralement être placée sur le lobe d'une oreille. En général, il est judicieux de ne pas placer l'électrode de référence trop proche de la zone d'intérêt. Le câble noir est câblé sur la PIN du BIAS. Cette électrode sera souvent placée sur l'autre lobe d'oreille. Son intérêt est de venir réduire le bruit des signaux. En effet, lors des mesures, le corps humain se comporte comme une antenne pour les rayonnements à 50Hz

venant du réseau électrique. L'électrode placée sur le BIAS va donc venir « copier » les rayonnements sur cette zone de fréquence et les soustraire du signal. (voir Annexe 1)

Utilisation des PIN SRB1 et SRB2

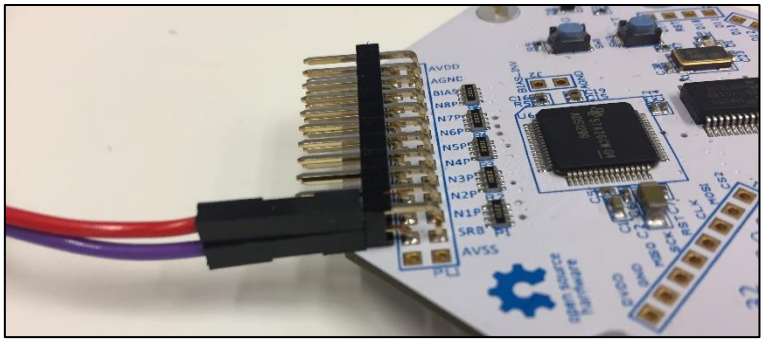


Figure 8 : PIN SRB1 et SRB2

Sur cette image, le câble rouge est sur la pin SRB1 (dessus) et le câble violet sur la pin SRB2 (dessous). Dans tous les cas, les pin SRB sont utilisées en tant que référence pour les mesures, mais on des utilisations différentes en pratique :

- **SRB1 :** est la référence pour les pin P et N. SRB1 est utilisé soit pour TOUS, soit AUCUN des canaux.
- **SRB2 :** Est le plus souvent utilisé, il s'agit de la référence pour les pin N. Il est possible de choisir individuellement sur quels canaux SRB2 est utilisé en référence. (Par exemple si on veut réaliser une mesure de potentiel entre N et P)

b.2) Graphical User Interface – GUI OpenBCI

Les chercheurs qui développent le kit OpenBCI fournissent également une interface graphique pour les utilisateurs qui permet d'effectuer quelques opérations de traitement de signal simples, mais surtout de visualiser les signaux. Cette interface est disponible en open source, ce qui permet aux personnes qui en ont le désir de rajouter des modules en fonction de leurs applications.

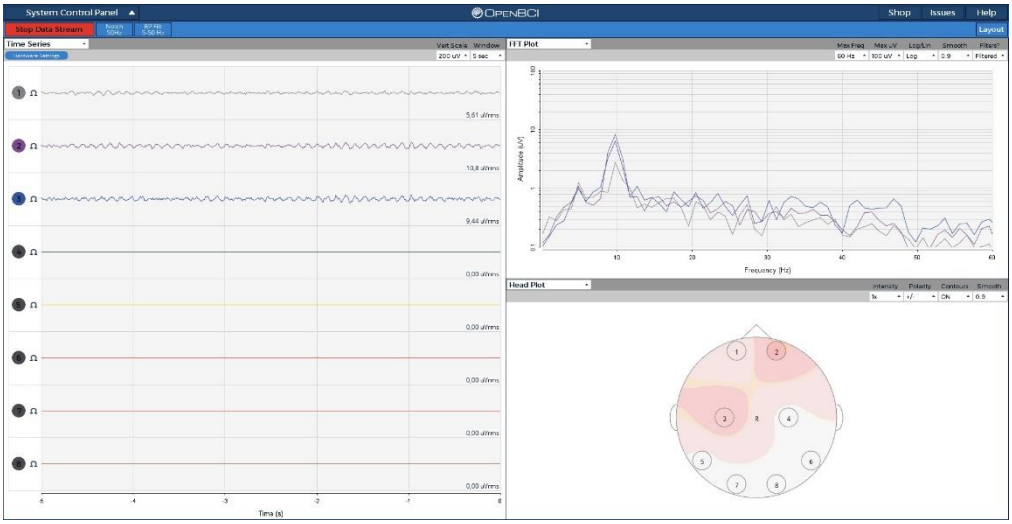


Figure 9 : GUI d'OpenBCI

b.3) Performances actuelles :

Pour l'instant la plupart des utilisateurs du kit OpenBCI l'utilisent pour 3 applications classiques :

- Détection de mouvements en EMG (Electrodes placées sur les muscles)
- Détection d'ondes alpha
- Application utilisant les SSVEP (en disposant plusieurs matrices de LED à des fréquences différentes)

c) Utilisation d'OpenBCI avec OpenVibe

c.1) Présentation d'OpenVibe

OpenVibe est un logiciel entièrement gratuit et open source développé par l'INRIA (Institut National de Recherche en Informatique et en Automatique). Il est spécialement conçu pour la création, le test et l'utilisation d'interfaces cerveaux-ordinateur. OpenVibe est un logiciel utilisé pour les applications en temps réel des neurosciences (et tout particulièrement pour les signaux cérébraux). Il peut être utilisé pour l'acquisition de ces signaux, le traitement, la classification, mais également pour les visualiser, en temps réel.

OpenVibe fonctionne sur un système de scénarios qui peuvent être élaborés à partir de différentes boites outils spécifiques.

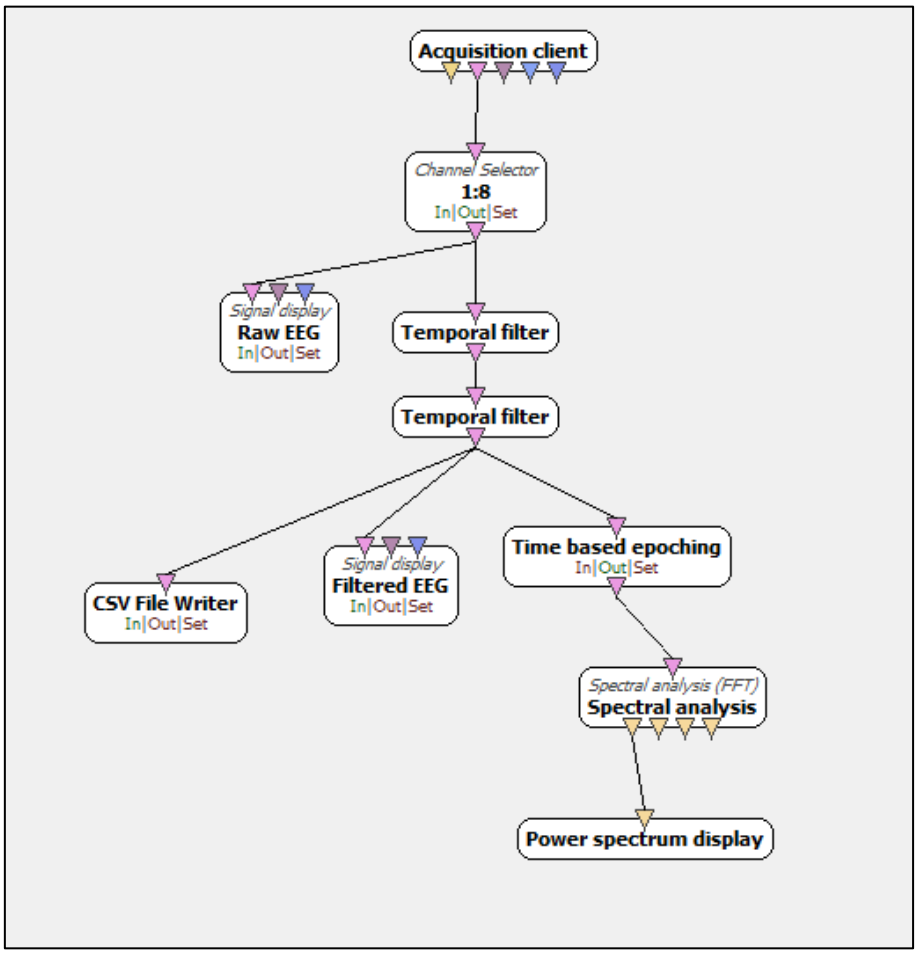


Figure 10 : Exemple de scénario sur OpenVibe (extrait d'une application OpenBCI)

Dans cet exemple d'application, on peut retrouver des éléments essentiels d'un scénario OpenVibe :

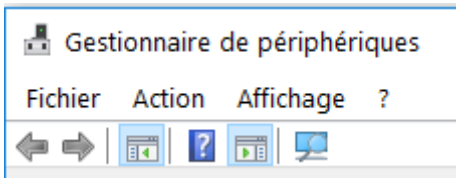
- **Acquisition client :** permet l'acquisition des données à partir du casque EEG.
- **Channel Selector :** l'utilisateur peut choisir les canaux qu'il utilisera dans l'application
- **Signal Diplay :** permet l'affichage des signaux dans une fenêtre dédiée
- **Temporal filter :** l'utilisateur peut utiliser filtres temporels classiques (Passe-bas, passe-haut etc… Un coupe-bande entre 48Hz et 52Hz est fréquemment utilisé pour retirer la fréquence liée au 50Hz du réseau électrique)
- **Time base epoching :** sert à définir une fenêtre glissante pour des calculs par exemple
- **Spectral analysis :** permet d'effectuer une analyse spectrale de type FFT
- **Power spectrum display :** permet l'affichage d'une FFT
- **CSV File Writer :** enregistrement des données sous le format .csv (Comma-separated values) qui permet d'obtenir les données sous forme d'un tableau facilement exploitable (avec des logiciels comme MATLAB par exemple)

c.2) Configuration pour OpenBCI

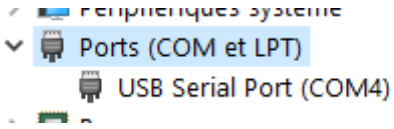
Les informations suivantes sont extraites du tutoriel donné sur le site de l'INRIA pour l'utilisation d'OpenBCI avec OpenVibe. Adresse du site : http://openvibe.inria.fr/drivers-openbci/

Cette configuration est nécessaire à chaque fois qu'on utilisera un autre port série (COM) sur Windows avec le dongle USB.

- Ouvrir le gestionnaire de périphériques Windows (Clic droit sur l'ordinateur dans l'explorateur windows, puis *Propriétés*)
- Ouvrir ensuite les trois panneaux (gauche et droite) du gestionnaire



- Double cliquer sur le périphérique correspondant (dans mon cas, COM4)



- Cliquer sur *Paramètres du port*, puis *Avancé*

- Paramétrer le périphérique de cette manière :

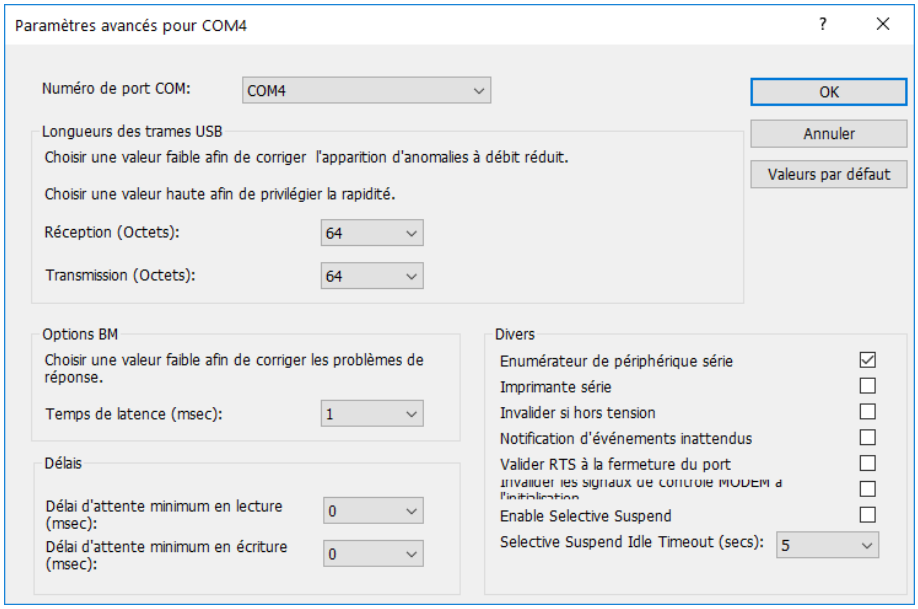


Figure 11 : Paramètres pour fonctionnement OpenBCI

c.3) Programmation de la carte

Pour paramétrer la carte OpenBCI Cyton sous OpenVibe en fonction des utilisations voulues, il faut communiquer avec en ASCII à travers l'interface du serveur d'acquisition d'OpenVibe.

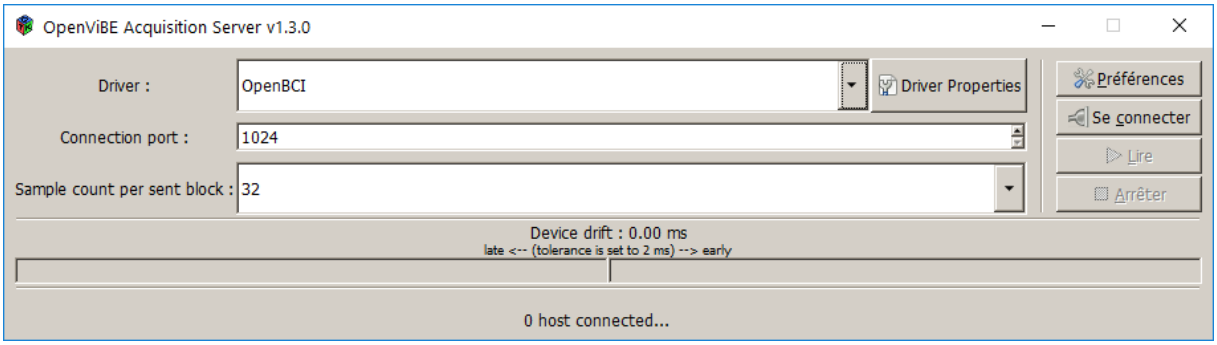


Figure 12 : Interface OpenVibe Acquisition

Pour rentrer des commandes personnalisées, il faut cliquer sur « *Driver Properties* » puis les rentrer dans l'onglet « *Custom Commands on Initialization* ».

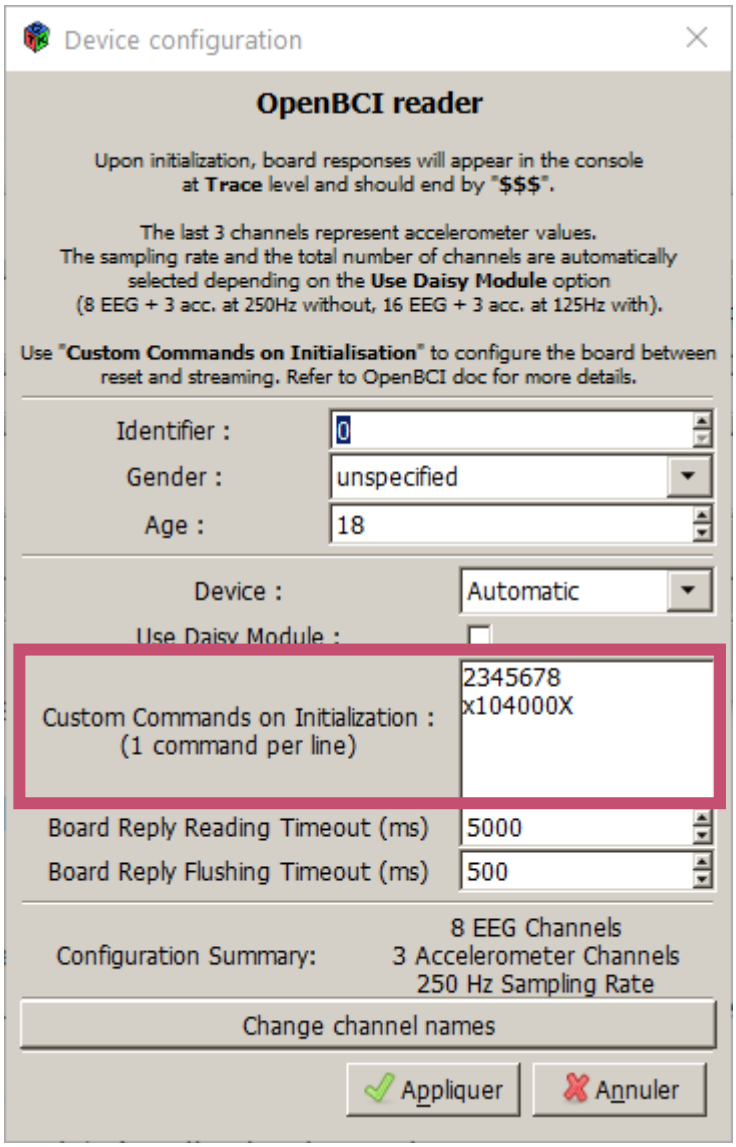


Figure 13 : Localisation de la ligne de commandes

Les commandes à rentrer s'écrivent de la manière suivante (Une ligne = Une commande) :

Eteindre un canal (Rentrer un chiffre éteindra le canal correspondant) :
1 2 3 4 5 6 7 8

Allumer un canal (Rentre un caractère éteindra le canal correspondant) :
! @ # $ % ^& *

Paramétrer un canal :
x(CHANNEL, POWER_DOWN, GAIN_SET, INPUT_TYPE_SET, BIAS_SET, SRB2_SET, SRB1_SET)X

Avec :

CHANNEL

- 1 2 3 4 5 6 7 8 pour les canaux de la carte seule
- Q W E R T Y U I pour les canaux du module Daisy

POWER_DOWN
- 0 = ON (par défaut)
- 1 = OFF

GAIN_SET
- 0 = Gain 1
- 1 = Gain 2
- 2 = Gain 4
- 3 = Gain 6
- 4 = Gain 8
- 5 = Gain 12
- 6 = Gain 24 (par défaut)

INPUT_TYPE_SET

Sélectionne la source du canal d'entrée du convertisseur
- 0 ADSINPUT_NORMAL (par défaut)
- 1 ADSINPUT_SHORTED
- 2 ADSINPUT_BIAS_MEAS
- 3 ADSINPUT_MVDD
- 4 ADSINPUT_TEMP
- 5 ADSINPUT_TESTSIG
- 6 ADSINPUT_BIAS_DRP
- 7 ADSINPUT_BIAS_DRN

BIAS_SET

Permet de choisir si le canal utilisé est relié au BIAS
- 0 = Remove form BIAS
- 1 = Include in BIAS (par défaut)

SRB2_SET

Permet de connecter ce canal à la PIN SRB2
- 0 = Disconnect this input from SRB2
- 1 = Connect this input to SRB2 (par défaut)

SRB1_SET

Permet de connecter toutes les PIN à SRB1
- 0 = Disconnect all N inputs from SRB1 (par défaut)
- 1 = Connect all N inputs to SRB1

Exemples :

La commande : **2345678** désactive tous les canaux sauf le 1.
La commande : **x104000X** permet d'effectuer une mesure en EMG sur le canal 1 en paramétrant le gain du canal à 8 et en déconnectant le canal 1 du BIAS et de SRB2. En effet, l'ordre de grandeur des signaux EMG est plus grand que les signaux EEG, ils risquent donc de perturber les autres canaux.

Les autres commandes sont disponibles sur le lien :
https://github.com/OpenBCI/Docs/blob/master/OpenBCI%20Software/04-OpenBCI_Cyton_SDK.md

d) Utilisation de scripts Matlab

Dans OpenVibe, il est possible d'utiliser des scripts « .m » développé avec Matlab. Cependant, cela nécessite de posséder une version légale de Matlab sur son ordinateur et en

version 32 bits obligatoirement (pour informations, Mathworks ne développe plus de version 32 bits, la toute dernière version possible utilisable est donc la 2015b)

Quand on utilise la box Matlab sous OpenVibe, il faut préciser plusieurs paramètres :

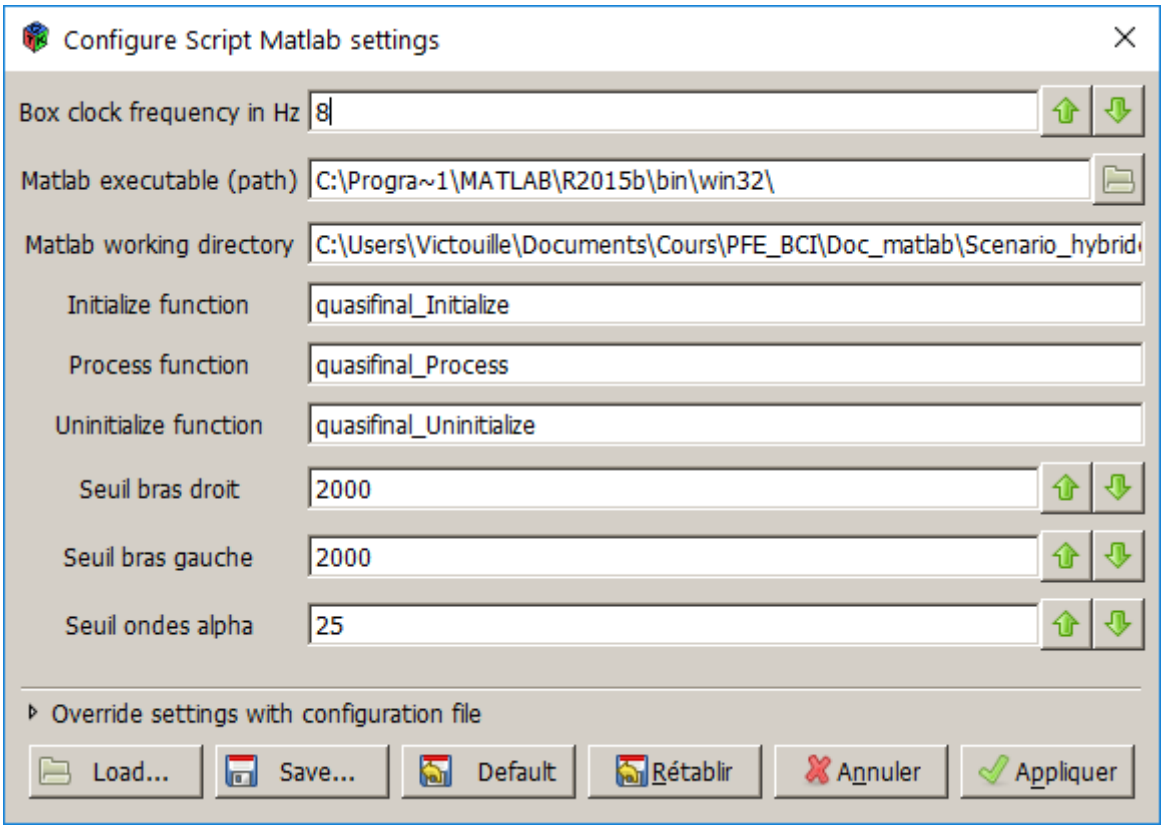


Figure 14 : Box Matlab dans OpenVibe

- Le chemin menant à l'exécutable Matlab (OpenVibe supporte moyennement les espaces dans les noms de chemins)
- Le chemin où se situe les scripts Matlab à utiliser pour la box (les 3 scripts doivent être situés dans le même répertoire)
- Les noms des trois fichiers .m (Initialize, Process, Uninitialize)

d.1) Architecture du code

Comme expliqué précédemment, l'utilisation de scripts matlab nécessite 3 fichiers .m qui doivent se situer dans le même répertoire.

Le fichier **Initialize.m** s'exécute lorsque le bouton « play » d'OpenVibe est appuyé. Il doit comporter au minima :

- La définition des canaux
- Le paramétrage des sorties de la box

Le fichier **Process.m** s'exécute en boucle pendant le déroulement du scénario. Il doit comporter au minima :

- La lecture des signaux d'entrée
- Le transfert des données aux sorties

Le fichier **Uninitialize.m** s'exécute lorsque le bouton « Stop » d'OpenVibe est appuyé (ou que le scénario est arrêté). Il doit comporter au minima :

- Libération des variables globales s'il y en a

d.2) Fonctions utilisées dans ce projet

Dans cette partie je vais détailler les points importants du code utilisé dans ce projet. Le reste des fonctions nécessaire au bon fonctionnement d'un script Matlab sur OpenVibe sont disponibles à cette adresse : http://openvibe.inria.fr/tutorial-using-matlab-with-openvibe/

- Chaque fichier .m doit être rédigé de la manière suivante :

```
function box_out = nomdelafonction(box_in)
end
```

- Une expression comme ci-dessous permet de récupérer une valeur rentrée directement dans les paramètres de la box Matlab d'OpenVibe (il suffit de faire clic droit, *modify settings*, *new*). A noter que l'indice entre parenthèses correspond au premier paramètre rajouté.

```
NomDeLaVariable = box_in.settings(1).value ;
NomDeLaVariable2 = box_in.settings(2).value ;
```

- L'expression pour récupérer le premier signal de la box est la suivante. Les données sont alors stockées dans le vecteur ou matrice « matrix_data ». (Pour récupérer le deuxième signal de la box, changer l'indice « 1 » par « 2 »)

```
[box_in, start_time, end_time, matrix_data] = OV_popInputBuffer(box_in,1);
```

L'intégralité des codes sont disponibles en annexes et sont commentés.

e) Communication en liaison série

Dans le but de communiquer avec un système physique, j'ai choisi de mettre en place une liaison entre mon interface cerveau ordinateur sur OpenVibe et une carte Arduino Uno. Cette communication est établie depuis les 3 scripts Matlab, de la manière suivante :

- Dans le script initialize.m, on déclare l'arduino en variable globale pour qu'il puisse être utilisé par les 3 scripts, on déclare ensuite une liaison série qu'on ouvre ensuite avec la fonction fopen :

```
global arduino_martin; % Déclaration de la variable globale de
l'arduino
arduino_martin=serial('COM5','BaudRate',9600); % Création d'une
communication série sur le port COM5
fopen(arduino_martin); % Ouverture de la liaison série avec
l'arduino
fprintf(arduino_martin,'%s',char(2)) % Dans ce cas là, on éteint la
LED (voir programme arduino)
```

- Dans le script Process.m, on déclare toujours la variable globale, puis on envoie le caractère voulu en fonction de notre application. Dans mon cas, le caractère 1 allume une LED sur l'arduino, alors que le caractère 2 l'éteint.

```
global arduino_martin; % Déclaration de la variable globale de
l'arduino
```

```
        fprintf(arduino_martin,'%s',char(1)) % LED allumée (voir
selon code arduino)
```

- Dans le script Uninitialize.m, on déclare toujours la variable globle, puis on ferme la liaison série avec fclose. Enfin, on libère la variable globale.

```
global arduino_martin; % Déclaration de la variable globale de
l'arduino
fclose(arduino_martin); % Fermeture de la communication série avec
l'arduino
clear arduino_martin; % Clear de la variable globale
```

Le code qui doit être sur la carte Arduino Uno est disponible en annexe 4 du rapport. Il suffit d'utiliser le logiciel Arduino pour le transférer sur la carte.

Dans cet exemple on se contente d'allumer une LED, mais on peut très bien imaginer contrôler des moteurs ou d'autres systèmes à partir de la carte.

III. Applications réalisées :

L'objectif du projet est de déterminer les possibilités d'OpenBCI, c'est dans cette optique que j'ai réalisé plusieurs des applications classiques des BCI, présentées en partie I.b.1. J'ai donc testé le kit OpenBCI sur un P300 speller, sur une interface devant détecter les mouvements des pieds et sur un système utilisant à la fois des EMG et une détection des ondes alpha.

a) P300 Speller

a.1) Montage

Pour le P300 speller, nous allons chercher à détecter une activité cognitive dans le signal, comme expliqué en partie I.b.1. Pour se faire, l'activité cérébrale est mesurée à 3 endroits. Au niveau de l'électrode Cz qui est la zone la plus adaptée pour détecter une activité cognitive, mais également aux électrodes O1 et O2, qui sont situées au niveau des lopes

occipitaux, qui correspondent aux zones visuelles ; et le P300 speller est une application principalement visuelle. On retrouve également la référence SRB2 sur une oreille et le BIAS chargé d'éliminer les interférences liées au 50Hz sur l'autre.

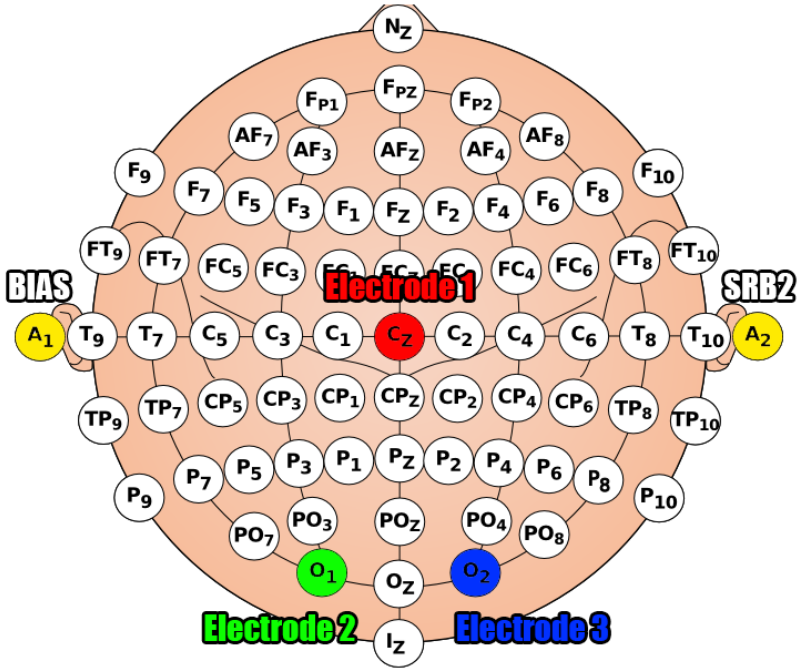


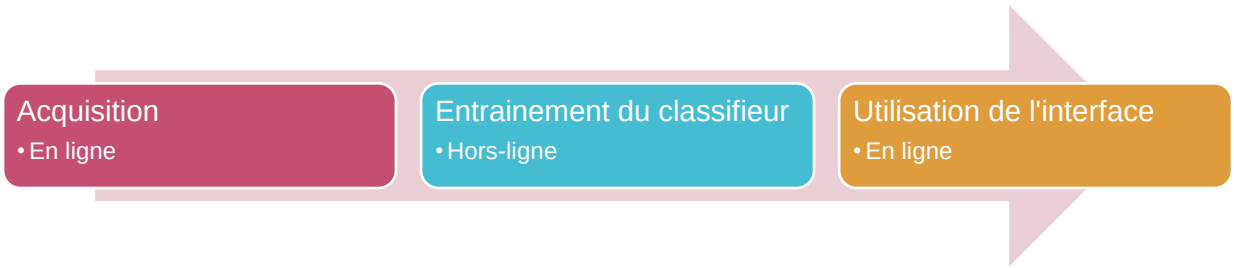
Figure 15 : Montage utilisé pour le P300 Speller

a.2) Traitement du signal utilisé

Figure 16 : Visuel du P300 Speller

Dans cette application, nous recherchons une action cognitive qui se détecte dans les fréquences 1 à 20Hz. Pour ce faire nous utilisons un passe-bande. Ce signal filtré va ensuite être utilisé pour entrainer un classifieur (LDA) a deux classes. En effet, l'objectif du P300 speller est de pouvoir retrouver la lettre sur laquelle l'utilisateur se concentre. Les deux classes issues de l'entrainement du classifieur sont donc : « La lettre est ciblée par l'utilisateur » et « la lettre n'est pas ciblée par l'utilisateur ».

a.3) Scénarios OpenVibe



Cette application utilise 3 scénarios sur OpenVibe. Le premier sert à faire une acquisition dans le but de pouvoir par la suite entrainer le classifieur. Ce scénario s'exécute donc « en ligne », l'utilisateur doit se concentrer pendant plusieurs minutes sur différentes lettres dans le but d'étalonner le système. Le second scénario utilise les données obtenues du premier scénario pour entrainer un classifieur. Ceci se déroule « hors-ligne », l'utilisateur n'a rien à faire. Le dernier scénario, en ligne, correspond à l'interface « finale » en quelque sorte. L'utilisateur peut se concentrer sur les lettres qu'il désire, et grâce au classifieur entrainé, le système est capable de déterminer quelles sont les lettres sélectionnées.

- **1er scénario :**

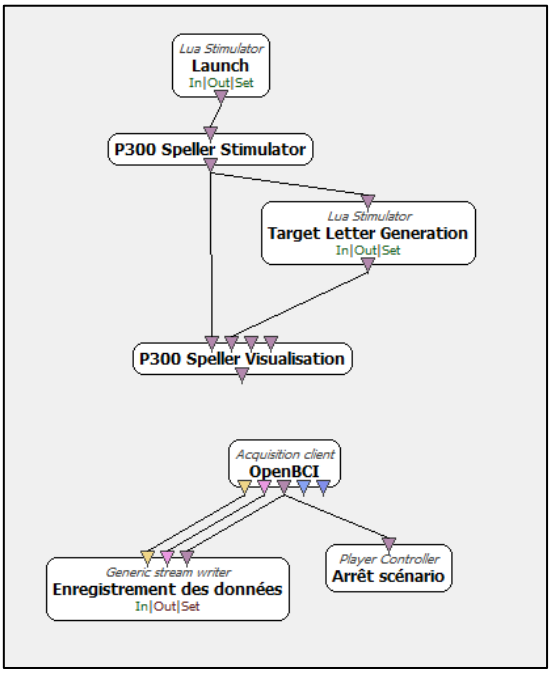


Figure 17 : 1er scénario P300 Speller

Ce scénario se décompose en deux parties. La partie supérieure correspond à l'aspect visuel du système. Les scripts génèrent les lettres choisies aléatoirement sur lesquelles l'utilisateur devra se concentrer. Cela permet également de générer les « stimulations » sur OpenVibe. Ce sont en quelque sorte des repères temporels utilisés pour calibrer le classifieur dans le temps.

La seconde partie du scénario consiste tout simplement à l'enregistrement des données issues du système OpenBCI dans un fichier. La box « Arrêt scénario » sert à arrêter automatiquement le scénario une fois l'enregistrement terminé.

- **2nd scénario :**

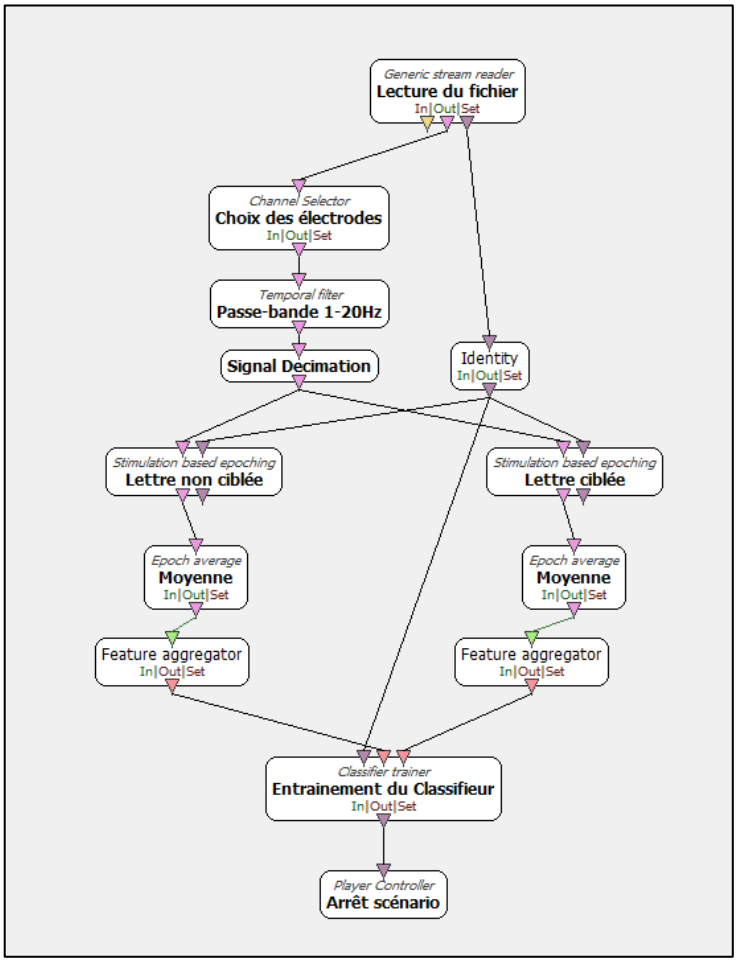


Figure 18 : 2nd scénario P300 Speller

Dans ce scénario, OpenVibe va aller lire les données du fichier issu de l'acquisition. On va ensuite faire une sélection des électrodes à étudier (Cz,O1,O2). Un passe-bande est ensuite utilisé pour sélectionner les fréquences d'intérêt (1-20Hz). Le signal est ensuite divisé en deux parties selon les stimulations enregistrées. Cela va servir à répartir les signaux entre « la lettre est ciblée » et « la lettre n'est pas ciblée » par l'utilisateur.

Ces données sont ensuite réunies pour entrainer le classifieur.

Une fois cette phase d'entrainement terminée, le scénario est automatiquement arrêté.

- **3ème scénario :**

Le troisième scénario, qui correspond à l'interface dans sa version « finale » ressemble énormément au second. Mais est en ligne. On n'utilise plus un fichier mais directement les données envoyées par OpenBCI. Cette fois, les données sont gérées en direct et on utilise le classifieur (un fichier) qui a été entrainé lors du second scénario. On peut décider de choisir soi-même les lettres sur lesquelles on veut se focaliser ou alors permettre au logiciel de nous en proposer pour tester les performances du classifieur et du scénario.

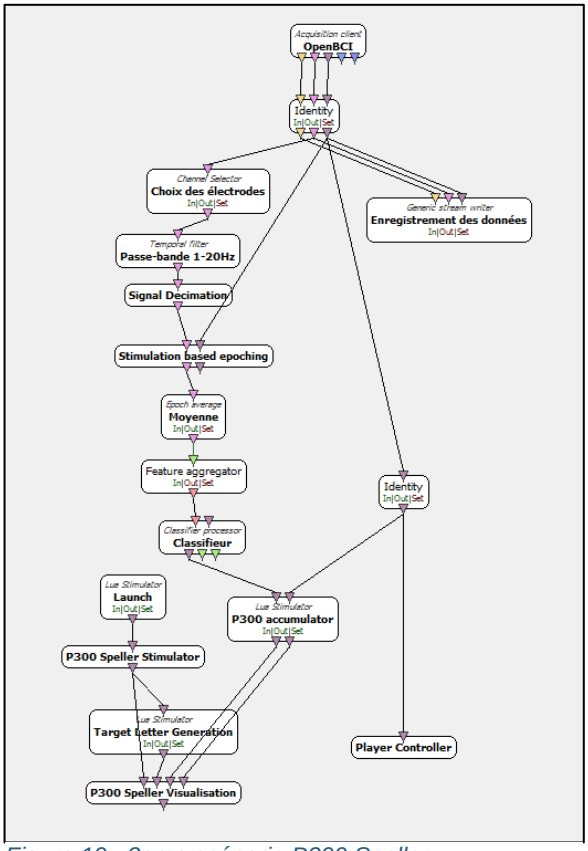


Figure 19 : 3eme scénario P300 Speller

a.4) Résultats

Lors des premiers tests du scénario, j'obtenais des résultats d'exactement 0% de réussite avec ce scénario. J'ai donc cherché à trouver l'origine du problème. J'ai testé mon scénario avec des fichiers de données d'autres personnes. Et le scénario fonctionnait très bien (plus de 80% de taux de réussite à la classification). Le problème venait donc de l'acquisition. Début janvier, l'INRIA a proposé une mise à jour d'OpenVibe comprenant notamment une correction de certains problèmes pouvant survenir avec OpenBCI au niveau de l'acquisition. J'ai donc pu relancer plusieurs campagnes en obtenant des résultats d'environ 30% de réussite, ce qui est plutôt encourageant même si le taux reste assez faible. Je pense qu'il serait possible d'obtenir de meilleurs résultats en utilisant plus d'électrodes et un filtrage spatial Laplacien pour obtenir des valeurs plus fiables. L'utilisation d'un casque EEG est donc fortement préconisée pour permettre l'utilisation de plus d'électrodes et une meilleure stabilité de celles-ci.

b) Détection des mouvements de pieds

b.1) Montage

Dans le cadre de l'imagerie motrice des pieds, l'intérêt est de détecter une augmentation de la puissance dans une bande de fréquences spécifique, entre 16 et 24Hz. Pour cela nous allons placer une électrode au niveau du cortex moteur qui correspond au mouvement des pieds. Cette électrode se place donc en Cz, au sommet au crâne. On utilise toujours deux électrodes au niveau des oreilles, en tant que référence (SRB2) et pour éliminer le bruit issu du 50Hz (BIAS).

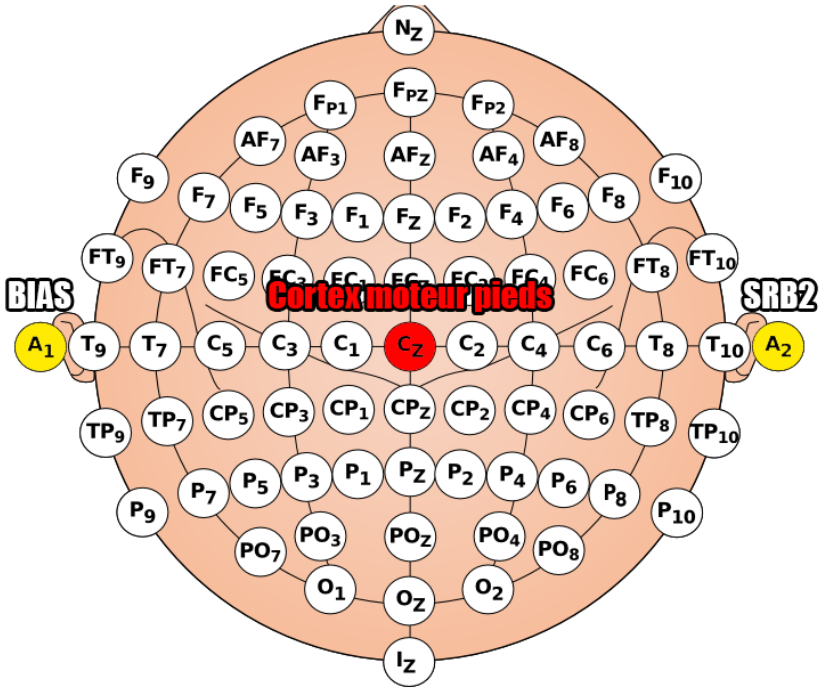


Figure 20 : Montage imagerie motrice des pieds

b.2) Traitement du signal utilisé et scénario OpenVibe

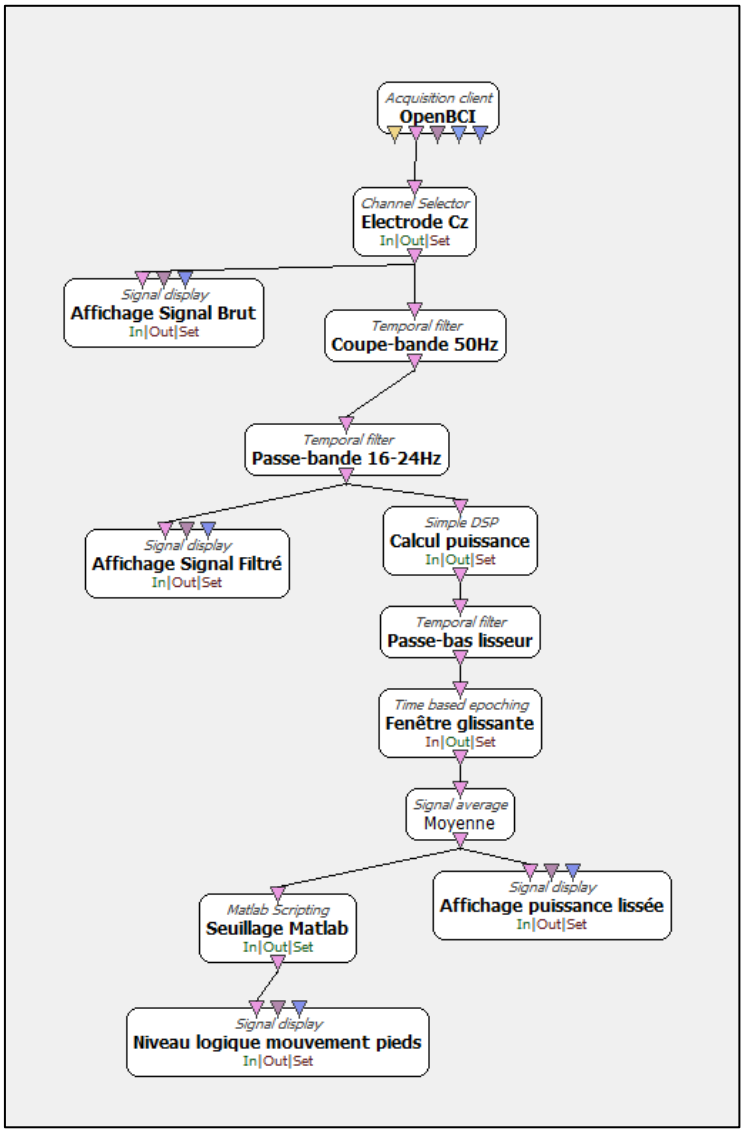


Figure 21 : Scénario imagerie motrice

Dans ce scénario, nous nous intéressons à détecter une augmentation de puissance dans une certaine bande de fréquences. Pour cela nous récupérons le signal issu de l'électrode Cz, qui est tout d'abord filtré pour éliminer le 50Hz à l'aide d'un coupe-bande. Un passe-bande est ensuite utilisé pour ne sélectionner que les fréquences intéressantes, ici entre 16 et 24Hz. Nous calculons ensuite la puissance de ce signal que nous « lissons » avec passe-bas dans le but d'obtenir un signal qui pourra ensuite être seuillé. Actuellement le seuil qui permet de déterminer s'il y a mouvement des pieds ou non est décidé arbitrairement. Mais il est tout à fait envisageable de sélectionner ce seuil comme étant X fois l'écart quadratique du signal lorsque la personne ne bouge pas les pieds.

En sortie du scénario, nous obtenons un signal logique 0 ou 1 qui permettrait de commander un système. Une liaison série est par exemple mise en place et permet d'allumer une LED sur Arduino.

b.3) Résultats

Concernant cette application, le scénario OpenVibe est fonctionnel, et répond correctement au besoin. En revanche, en pratique, le système ne permet pas de détecter les mouvements de pieds lorsqu'ils sont en l'air, ni lorsqu'ils sont imaginés. Cette application ne peut être utilisée qu'en bougeant les pieds au niveau du sol. Après quelques recherches, j'ai constaté que d'autres personnes rencontraient le même problème pour faire ce genre d'applications avec le système OpenBCI et OpenVibe. De mon point de vue, le problème vient soit du système d'acquisition qui n'est pas adapté à ce genre d'application, soit d'une mauvaise configuration de la carte. Il est en effet possible que certains canaux viennent perturber la mesure. Il serait intéressant de reprendre les mesures avec une autre configuration de la carte.

c) EMG & Ondes alpha

c.1) Fonctionnement de l'application BCI et montage

Dans cette application, nous allons nous réaliser ce qui correspond à une interface cerveau-ordinateur « hybride ». En effet, cette BCI mêle des mesures EMG avec des mesures EEG.

Cette interface permet de détecter le mouvement de deux muscles, dans notre cas nous allons nous focaliser sur les bras, mais il est possible d'imaginer placer les électrodes au niveau de muscles encore valides d'une personne à mobilité réduire (les doigts par exemple).

Il est également possible de détecter la présence d'ondes alpha, qui sont déclenchées lorsque l'utilisateur du système ferme les yeux et est au repos.

Les électrodes EMG sont donc placées au niveau des bras, sur le muscle bachio-radial de chaque bras (à l'avant-bras donc).

Figure 22 : Placement de l'électrode sur le l'avant-bras

Les références de chaque bras sont placées au niveau des tempes (T7 et T8) opposées (dans le but de récupérer le rythme cardiaque comme référence).

La mesure du signal pour détecter les ondes alpha se fait sur la zone du lobe occipital gauche (à l'arrière de la tête) Une seule électrode est utilisée mais on peut imaginer en placer plusieurs et effectuer un filtrage spatial.

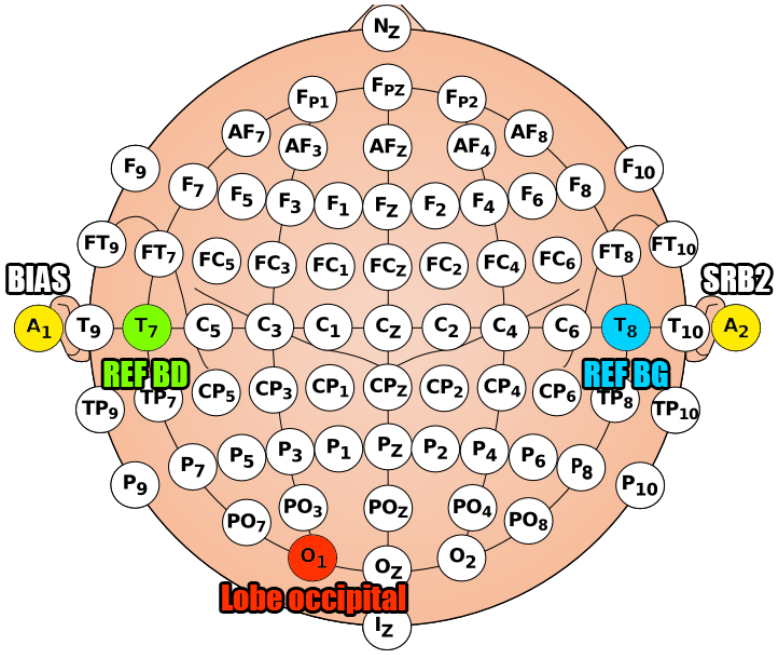


Figure 23 : Montage interface EMG & ondes alpha

c.2) Visualisation des signaux

Au repos, la représentation fréquentielle des signaux est celle-ci :

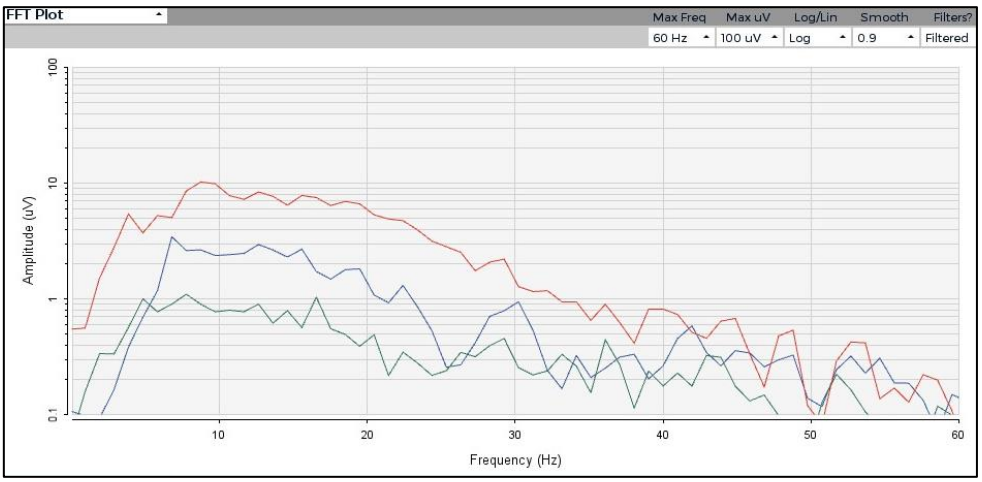


Figure 24 : FFT au repos

- En rouge : le signal issu du bras gauche
- En bleu : le signal issu du bras droit
- En vert : le signal issu du lobe occipital gauche

Lorsque l'utilisateur va contracter un de ces bras, on peut très nettement remarquer une augmentation de la puissance du signal dans les fréquences supérieures à 30Hz.

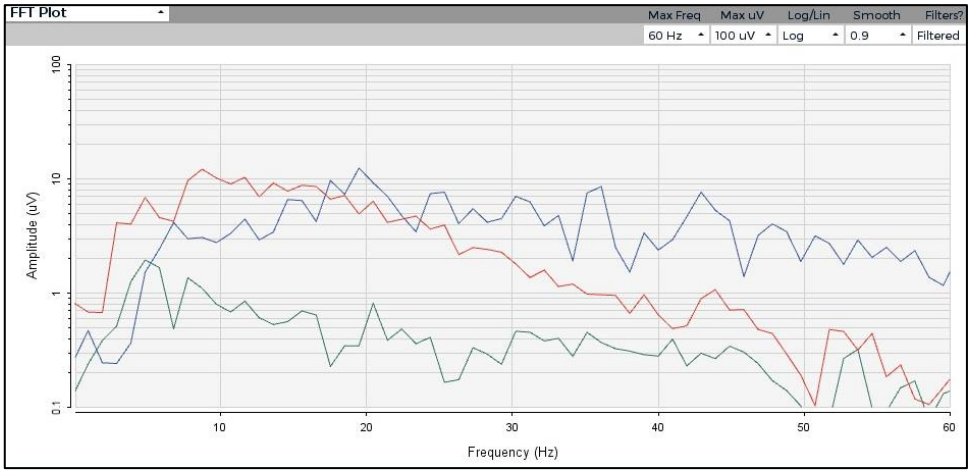


Figure 25 : FFT lorsque bras contracté (courbe bleue)

Au niveau du signal en lui-même, cela se caractérise parce que l'on appelle une « bouffée EMG » qui correspond à une augmentation de la fréquence du signal :

C'est donc cette caractéristique d'augmentation d'une bande de fréquence que le scénario cherchera à identifier.

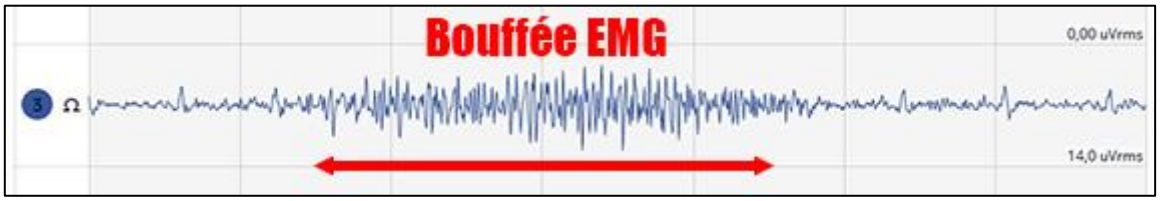


Figure 26 : Bouffée EMG sur signal temporel

Concernant les ondes alpha, lorsque nous sommes au repos, avec les yeux fermés, nos signaux cérébraux vont se mettre à osciller entre 9 et 13Hz. Cela peut être facilement identifiable par un pic sur cette gamme de fréquences.

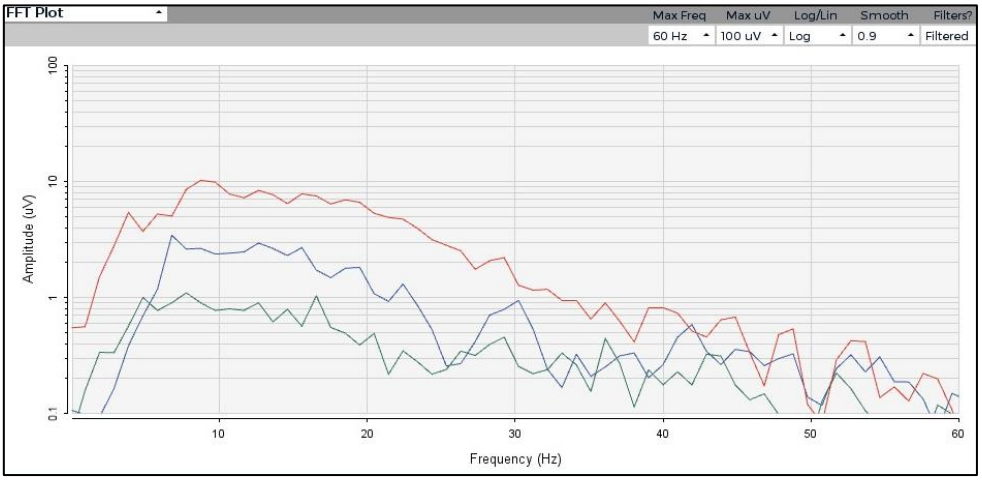


Figure 27 : FFT au repos

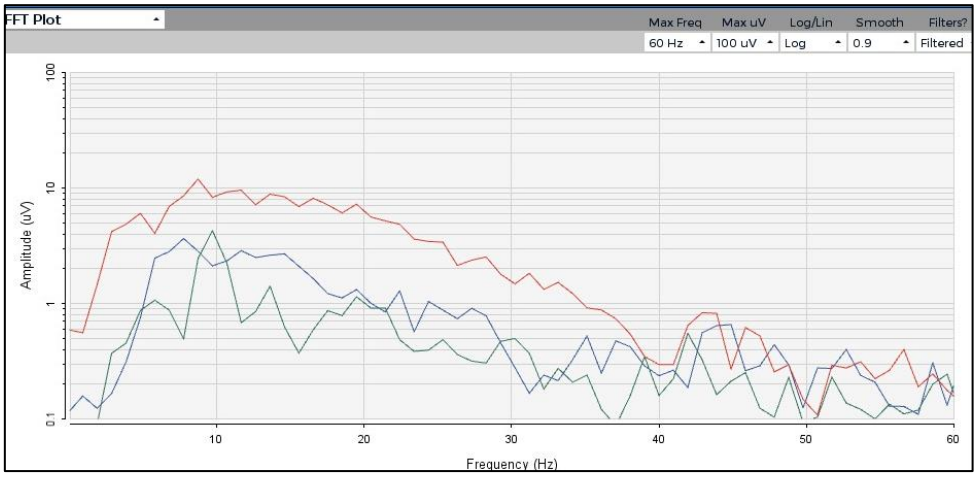


Figure 28 : FFT avec ondes alpha (courbe verte)

Au niveau du signal d'un point de vue temporel, il est également facile de repérer l'oscillation des ondes.

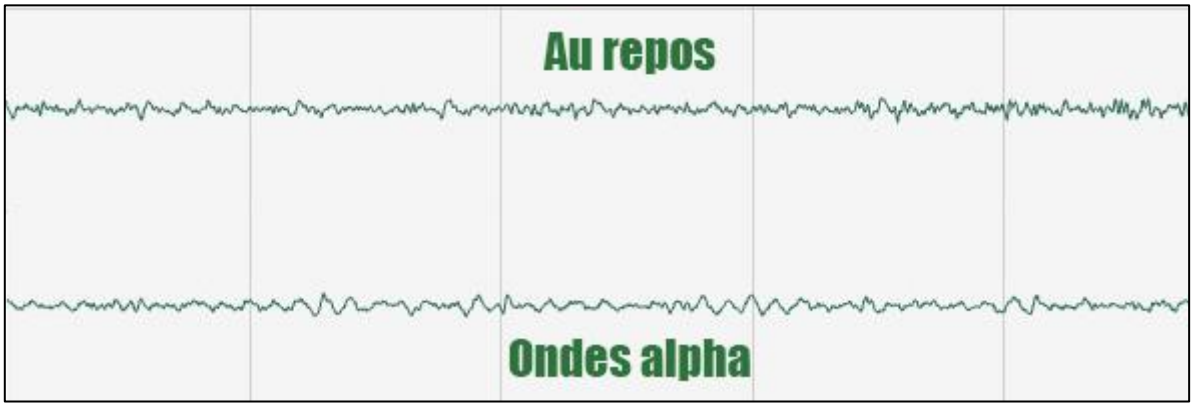


Figure 29 : Signaux issus du lope occipital au repos et avec présence d'ondes alpha

c.3) Scénario OpenVibe

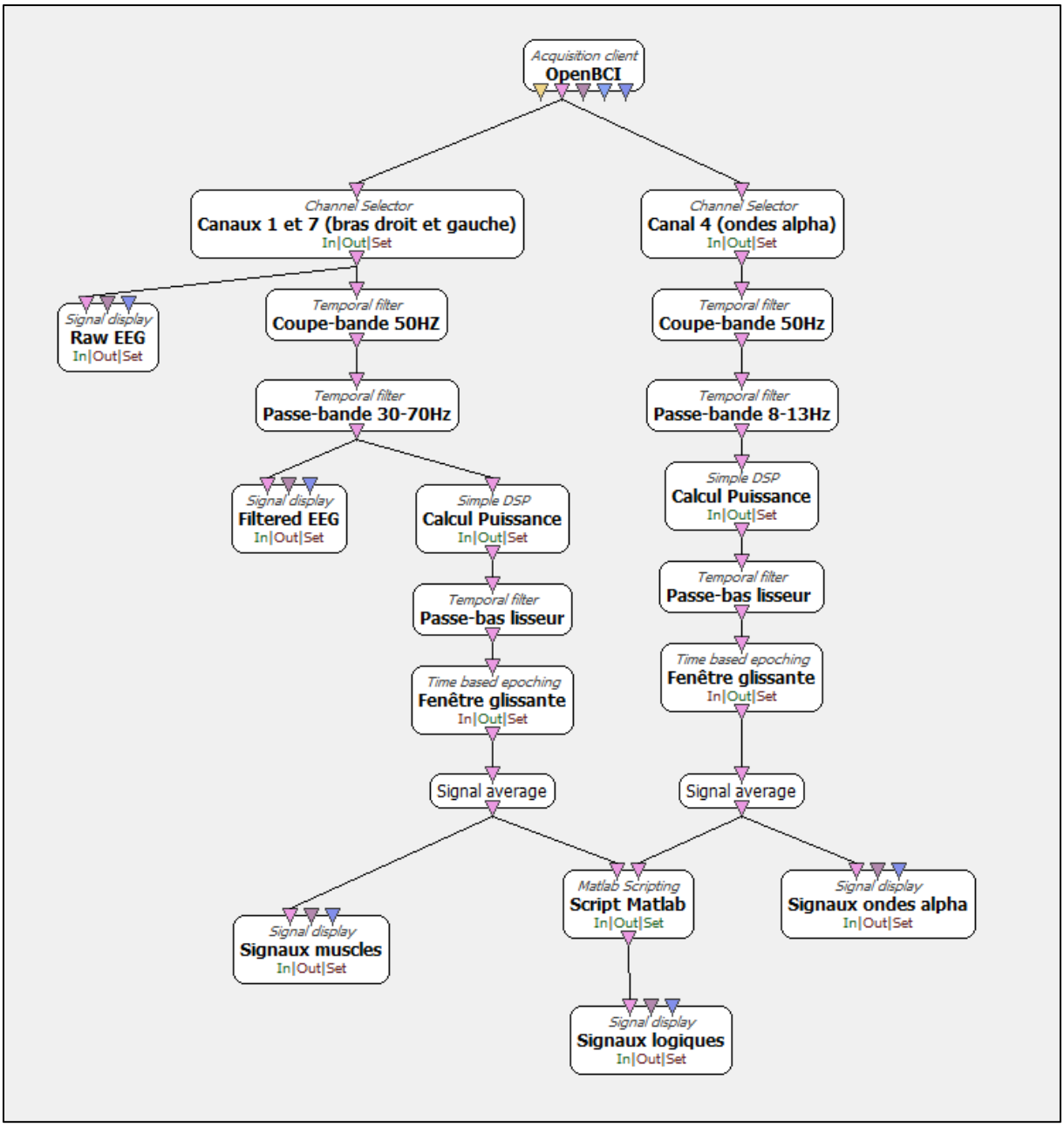


Figure 30 : Scénario complet de l'application

Le scénario de cette application est assez similaire à celui de l'imagerie moteur. Tout d'abord les signaux sont récupérés d'OpenBCI. On va ensuite sélectionner les 3 canaux utiles. 1 et 7 correspondent aux bras (partie gauche du scénario) alors que le canal 4 correspond aux ondes alpha (partie droite). On va donc d'abord supprimer le bruit lié au 50Hz pour chaque signal, il s'agit ensuite de choisir les fréquences d'intérêt de chaque paradigme étudié. 30-70Hz pour les EMG et 8-13Hz pour les ondes alpha. On calcule ensuite la puissance en fenêtre glissante des signaux qu'on va ensuite lisser pour être seuillée.

Les seuils sont directement paramétrables depuis la box Matlab. Il suffit de rentrer les valeurs voulues pour faire une interface plus ou moins réactive.

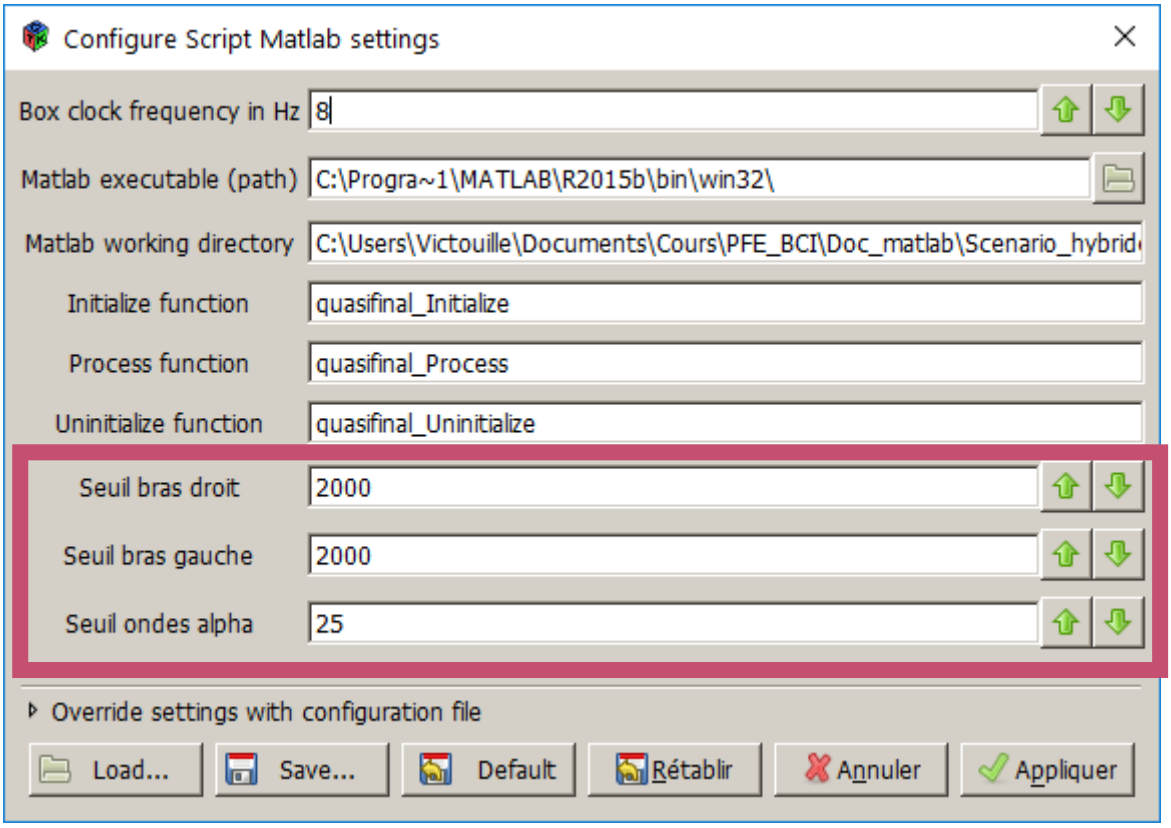


Figure 31 : Paramétrage des seuils sur OpenVibe

A la fin du scénario, l'interface délivre 4 signaux logiques, qui correspondent à :

- Contraction du bras droit
- Contraction du bras gauche
- Contraction des deux bras
- Présence d'ondes alpha

On peut donc ensuite imaginer contrôler un système à partir de ces quatre informations.

La fenêtre finale de visualisation est la suivante :

Lorsque les niveaux logiques sont à 1, il y a détection de contraction de bras ou détection d'ondes alpha.

Lorsque le niveau logique des ondes alpha est à 1, un signal sonore est produit par l'interface pour informer l'utilisateur, qui a les yeux fermés.

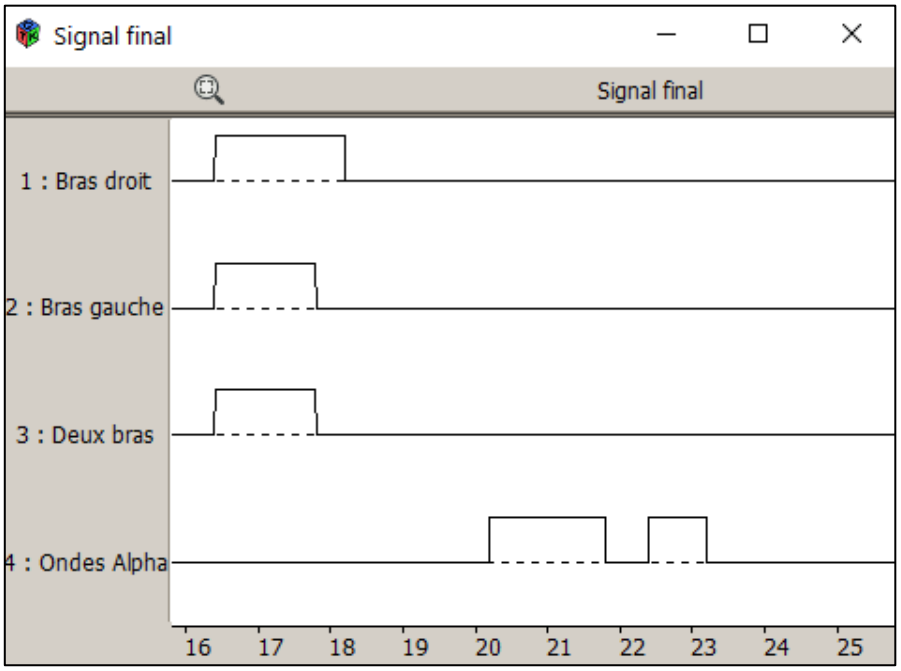


Figure 32 : Fenêtre finale avec niveaux logiques

c.4) Résultats

Les résultats de cette application sont très concluants. La détection d'activité au niveau des muscles de bras ne pose aucun problème une fois le bon seuil fixé. On peut très bien imaginer placer les électrodes sur d'autres muscles et prendre des seuils plus bas pour détecter des contractions plus faibles (on se projetant par exemple sur une application pour des personnes à mobilité très réduite). La détection des ondes alpha fonctionne également très bien avec un temps de réaction du système de l'ordre de la seconde (induit principalement par le traitement du signal qui fonctionne en moyenne glissante). En revanche, il faut que l'utilisateur reste relativement fixe car des mouvements de la tête peuvent perturber le signal et créer des artefacts au niveau du signal.

d) Synthèse des applications

Pour faire la synthèse des performances testées d'OpenBCI durant ce projet, j'ai réalisé un tableau synthétique reprenant les différentes applications que j'ai « notées » selon plusieurs critères.

- La difficulté d'utilisation, c'est-à-dire si le système est utilisable directement ou non, s'il faut entrainer un classifieur ou non.
- La pertinence des résultats, c'est-à-dire si les résultats sont cohérents avec les attentes.
- Portabilité, c'est-à-dire la difficulté à faire fonctionner l'application entre des personnes différentes.
- Applications par la communauté OpenBCI, le système étant Open source, il est intéressant de voir à si les applications sont réalisées ou non par d'autres personnes.

Application	**EMG**	**Ondes alpha**	**P300 Speller**	**Imagerie motrice des pieds**
Difficulté d'utilisation	Simple	Simple	Lourde	Simple
Résultats	Efficace	Efficace	A améliorer	A améliorer
Portabilité	Simple	Moyenne	Lourde	Moyenne
Applications par la communauté OpenBCI	Très commun	Commun	Rare	Très rare

On peut donc aisément remarquer qu'il est très facile de mettre en place et d'utiliser des mesures EMG avec le système OpenBCI. Les ondes alpha sont également quelque chose de relativement simple à mettre en place, même si il est parfois délicat de détecter les ondes sur certaines personnes. En revanche, l'imagerie motrice des pieds reste quelque chose d'encore expérimental avec le kit OpenBCI, ainsi que le P300 Speller qui est très difficile à mettre en place.

Cependant, on peut constater avec la dernière application (EMG & ondes alpha) qu'il est possible de mettre en place une application BCI simple et fonctionnelle à l'aide du kit OpenBCI.

Perspectives du projet

L'objectif de ce projet était principalement de comprendre et décrire le fonctionnement du kit OpenBCI, mais également ses possibilités, et si possible l'intégrer dans une interface complète. La personne qui prendra la suite du projet aura donc accès à toute la documentation de ce projet, sur le fonctionnement du kit et des moyens pour le mettre en œuvre. Il est donc totalement possible de faire évoluer ce projet et les applications vers des utilisations de classifieurs plus poussés.

Une autre amélioration serait possible, En effet, le système d'acquisition OpenBCI mis en œuvre dans ce projet utilise des électrodes passives. Il faut donc appliquer du gel sur ces électrodes qui se placera entre celles-ci et le cuir chevelu. Il faut ensuite placer les électrodes sur la tête de l'utilisateur à l'aide de scotch médical. Le temps de cette préparation est estimé à 10 minutes pour une électrode et il est raisonnable de rajouter 5 minutes de préparation par électrode. Il faut également le même temps pour retirer le système, nettoyer le cuir chevelu du patient et les électrodes pour éviter qu'elles se détériorent. Ce temps de préparation et de rangement du matériel a été une contrainte assez importante dans ce projet, car cela implique de devoir trouver des créneaux horaires suffisamment grands afin d'effectuer la manipulation. De plus, une fois le système installé au patient, il devient très difficile de se déplacer.

Le système OpenBCI est sensé fonctionner avec un casque OpenBCI spécifique conçu pour pouvoir accueillir la carte d'acquisition et plusieurs électrodes (16 si on utilise le module Daisy fourni avec le kit). Néanmoins il n'était pas possible de l'obtenir pour des questions logistiques. En revanche, il serait particulièrement intéressant de l'obtenir pour la suite du projet, cela permettrait à la fois de résoudre le problème d'installation du matériel (les électrodes étaient directement fixées sur le casque, de plus, il s'agit d'électrodes sèches, ne nécessitant pas de gel). Cela permettrait également d'augmenter la fiabilité des mesures car les électrodes seront moins susceptibles de bouger, et donc d'induire des erreurs.

Dans ce projet, une liaison série a été mise en place entre le système BCI (en passant par un script Matlab dans OpenVibe) avec une Arduino Uno, il serait intéressant de faire évoluer

cette partie du projet pour rendre le système plus démonstratif avec par exemple la commande de servomoteurs dans une application ludique.

Enfin, un autre point possible du projet serait de tester une interface développée avec OpenBCI sur des patients. En effet, lors de ce projet, la totalité des essais ont été réalisés sur des personnes saines, en pleine possession de leurs moyens.

Bilan personnel

Je suis particulièrement heureux d'avoir pu participer à ce projet qui me tenait à cœur. Cela a été un réel plaisir de pouvoir travailler sur une technologie innovante et à destination de personnes dans le besoin. Cela a également été l'occasion d'effectuer un projet de recherche, ce qui est particulièrement enrichissant, d'un point de vue technique et humain.

Au cours de ce stage, j'ai utilisé et acquis de nombreuses connaissances et compétences, voici un tableau avec une liste non exhaustives de celles-ci.

	CONNAISSANCES	**COMPETENCES**
UTILISEES	**Electronique :** Connaissances sur les circuits embarqués, convertisseurs analogique/numérique **Analyse numérique :** Calcul matriciel	**Mesures et expérimentation :** Analyse des performances d'un système **Conception logicielle :** Développement de scénarios et programmes
ACQUISES	**Neurosciences :** Fonctionnement du cerveau et de caractéristiques physiologiques **Traitement du signal :** Découverte de méthodes de filtrage de signaux pour extraire des caractéristiques précises	**Gestion de projet :** Organisation du projet, création de documents spécifiques **Application BCI :** Création d'une application BCI complète, acquisition, traitement, classification et commande de système

Conclusion

Ce projet, réalisé dans le cadre de ma cinquième année à Polytech Lille a été mené à son terme.

J'ai été en mesure de proposer une documentation technique du kit OpenBCI et de fournir des indications précises sur son utilisation avec le logiciel OpenVibe. Une procédure et des codes ont été réalisés pour faire communiquer OpenVibe avec des scripts Matlab.

Le kit a été testé sur différentes applications, et une synthèse des performances a été fournie. Il apparait clairement que les mesures des EMG et des ondes alpha sont aisées avec OpenBCI. En revanche il est plus délicat de l'utiliser pour de l'imagerie motrice ou sur un système comme le P300 Speller.

J'ai également été capable de réaliser une application BCI complète, en partant du système d'acquisition, OpenBCI, en réalisant toute la partie traitement du signal et en créant une liaison série avec un Arduino Uno pouvant servir de système à piloter avec l'interface.

Le projet OpenBCI vient d'être lancé, tout reste à faire, et tout est possible. Je suis particulièrement fier d'avoir pu y apporter ma contribution et j'espère qu'il se poursuivra et sera perfectionné.

Victor Charnet

Références :

[1] Fabien Lotte, Anatole Lécuyer, Bruno Arnaldi. *Les Interfaces Cerveau-Ordinateur : Utilisation en Robotique et Avancées Récentes.* Journées Nationales de le Recherche en Robotique, Oct 2007, Obernai, France. 2007.

[2] Duprès Alban. *Interface cerveau-machine hybride pour pallier le handicap causé par la myopathie de Duchenne*. Traitement du signal et de l'image. Université Lille 1 - Sciences et Technologies, 2016. Français.

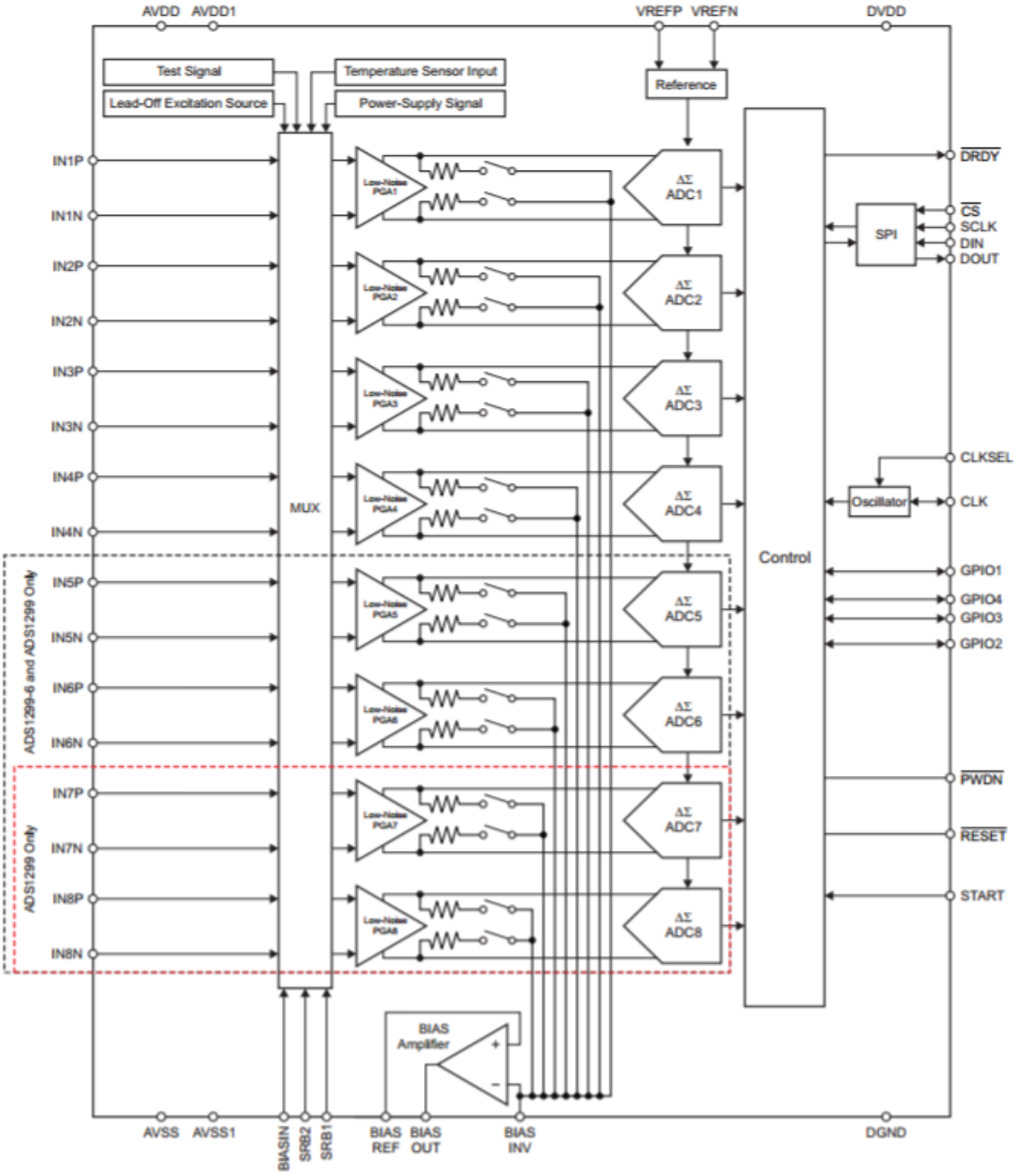
[3] Fabien Lotte, Anatole Lécuyer, Bruno Arnaldi. *Classification de données pour l'utilisation des brain computer interfaces en réalité virtuelle*, 2004-2005, IRISA, France

[4] Nicolas-Alonso LF, Gomez-Gil J. Brain Computer Interfaces, a Review. *Sensors (Basel,Switzerland)*. 2012;12(2):1211-1279. doi:10.3390/s120201211.

[5] Joan Fruitet. *Interfaces Cerveau-Machines basées sur l'imagination de mouvements brefs : vers des boutons contrôlés par la pensée*. Traitement du signal et de l'image. Université Nice Sophia Antipolis, 2012. Français.

Table des illustrations

ANNEXE 1 / Datasheet ADS1299



http://www.ti.com/lit/ds/symlink/ads1299.pdf

ANNEXE 2 / Initialize.m

```
% quasifinal_Initialize.m
% ------------------------------
% Author : Victor Charnet (Polytech)
% Date   : 23 February 2017
%
% La fonction quasifinal_Initialize se lance en appuyant sur le
bouton
% play sur OpenVibe
%
function box_out = quasifinal_Initialize(box_in)
disp('Initialisation du script Matlab')

% global arduino_martin; % Déclaration de la variable globale de
l'arduino
% arduino_martin=serial('COM5','BaudRate',9600); % Création d'une
communication série sur le port COM5
% fopen(arduino_martin); % Ouverture de la liaison série avec
l'arduino
% fprintf(arduino_martin,'%s',char(2)) % Dans ce cas là, on éteint
la LED (voir programme arduino)

nb_channels = 4; % Définition du nombre de canaux en sortie
channel_names = cell(1,nb_channels); % Création de la matrice
contenant le nom des canaux

% PERMET DE NOMMER AUTOMATIQUEMENT LES CANAUX
% for chan = 1 : nb_channels
%     channel_names{chan} = sprintf('channel %i',chan);
% end

% ICI ON NOMME LES CANAUX MANUELLEMENT
channel_names{1} = sprintf('Bras droit',1);
channel_names{2} = sprintf('Bras gauche',2);
channel_names{3} = sprintf('Deux bras',3);
channel_names{4} = sprintf('Ondes Alpha',4);

% PARAMETRES
nb_samples_per_buffer = 32;
sampling_rate = box_in.clock_frequency * nb_samples_per_buffer;

% Parametrage du signal de sortie de la box Matlab
box_in = OV_setSignalOutputHeader(box_in, 1, nb_channels,
nb_samples_per_buffer, channel_names, sampling_rate);

% On passe la box modifiée en sortie pour continuer le process
box_out = box_in;
end
```

ANNEXE 3 / Process.m

```
% quasifinal_Process.m
% ------------------------------
% Author : Victor Charnet (Polytech)
% Date   : 23 February 2017
%
% La fonction quasifinal_Process s'execute en boucle
%

function box_out = quasifinal_Process(box_in)
%% INITIALISATION

% global arduino_martin; % Déclaration de la variable globale de
l'arduino

Chan_count = 4; % Nombre de canaux de sortie
samples_per_buffer = 32; % Nombre d'échantillons par buffer
signal_de_sortie = zeros(Chan_count, samples_per_buffer); % On
initialise le signal de sortie de la box MATLAB

% On va lire dans les paramètres de la box Matlab sur OpenVibe les
valeurs des seuils
Seuil_bras_droit = box_in.settings(1).value;
Seuil_bras_gauche = box_in.settings(2).value;
Seuil_alpha = box_in.settings(3).value;

%% SEUILLAGE DES SIGNAUX
for i = 1: OV_getNbPendingInputChunk(box_in,1)

    % On récupère les signaux de l'entrée n°1 de la box Matlab dans
une matrice "matrix_data"
    [box_in, start_time, end_time, matrix_data] =
OV_popInputBuffer(box_in,1);
    % On récupère le signal de l'entrée n°2 de la box Matlab dans
une matrice "matrix_data2"
    [box_in, start_time, end_time, matrix_data2] =
OV_popInputBuffer(box_in,2);

    % ------------------------------------------------------------
    % SEUILLAGE DU SIGNAL BRAS DROIT
    if(matrix_data(1,:) >= Seuil_bras_droit) % On compare le signal
d'entrée filtré au seuil
        signal_de_sortie(1,1:samples_per_buffer)=1; % On met le 1er
signal de sortie à 1
        % fprintf(arduino_martin,'%s',char(1)) % LED allumée (voir
selon code arduino)

    else
        signal_de_sortie(1,1:samples_per_buffer)=0; % On met le 1er
signal de sortie à 0
        % fprintf(arduino_martin,'%s',char(2)) % LED éteinte (voir
selon code arduino)
```

```
    end
    % ---------------------------------------------------------------
    % SEUILLAGE DU SIGNAL BRAS GAUCHE
    if(matrix_data(2,:) >= Seuil_bras_gauche) % On compare le signal
d'entrée filtré au seuil
        signal_de_sortie(2,1:samples_per_buffer)=1; % On met le 2eme
signal de sortie à 1

    else
        signal_de_sortie(2,1:samples_per_buffer)=0; % On met le 2eme
signal de sortie à 0

    end
    % ---------------------------------------------------------------
    % SEUILLAGE DES DEUX BRAS
    if(matrix_data(2,:) >= Seuil_bras_gauche) && (matrix_data(1,:)
>= Seuil_bras_droit) % On compare les signaux en entrée aux seuils
        signal_de_sortie(3,1:samples_per_buffer)=1; % On met le 3eme
signal de sortie à 1

    else
        signal_de_sortie(3,1:samples_per_buffer)=0; % On met le 3eme
signal de sortie à 0

    end
    % ---------------------------------------------------------------
    % SEUILLAGE DES ONDES ALPHA
    if(matrix_data2 >= Seuil_alpha) % On compare le signal d'entrée
filtré au seuil
        signal_de_sortie(4,1:samples_per_buffer)=1; % On met le 4eme
signal de sortie à 1
        beep % On emet un signal sonore pour avertir l'utilisateur

    else
        signal_de_sortie(4,1:samples_per_buffer)=0; % On met le 4eme
signal de sortie à 0

    end
    % ---------------------------------------------------------------

    % On transfere les signaux "logiques" à la sortie n°1 de la box
Matlab sur OpenVibe
    box_in =
OV_addOutputBuffer(box_in,1,start_time,end_time,signal_de_sortie);

end

% On passe la box modifiée en sortie pour continuer le process
box_out = box_in;

end
```

ANNEXE 3 / Uninitialize.m

```
% quasifinal_Uninitialize.m
% ------------------------------
% Author : Victor Charnet (Polytech)
% Date   : 23 February 2017
%
% La fonction quasifinal_Uninitialize se lance en appuyant sur le
bouton
% stop sur OpenVibe
%
function box_out = quasifinal_Uninitialize(box_in)
disp('Fermeture du script Matlab')

% global arduino_martin; % Déclaration de la variable globale de
l'arduino
% fclose(arduino_martin); % Fermeture de la communication série avec
l'arduino
% clear arduino_martin; % Clear de la variable globale

% On passe la box modifiée en sortie pour arrêter le process

box_out = box_in;
end
```

ANNEXE 4 / Code_arduino.ino

```
int ledPin=13;
int matlabData;

void setup()
{
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
}

void loop()
{

   if(Serial.available()>0) // if there is data to read
   {
    matlabData=Serial.read(); // read data
    if(matlabData==1)
      digitalWrite(ledPin,HIGH); // turn light on
    else if(matlabData==2)
      digitalWrite(ledPin,LOW); // turn light off
  }
}
```

ANNEXE 5 / Compte-rendu de réunion avec Alban Duprès (06/12/2016)

Personnes présentes :

Victor Charnet (élève)
Alban Duprès (thésard au laboratoire CRIStAL)

Synthèse de la rencontre :

- Récupération des fichiers
  - Applications (Labyrinthe, karting etc…) développées sous UNITY
  - Scénarios OpenVibe
  - Exemples de classifieurs LDA (OpenVibe)
  - Fichiers Python (lien Classifieur/VRPN)
- Explications
  - Fonctionnement entre OpenVibe et Unity
  - Principe d'une interface cerveau ordinateur (d'un point de vue logiciel)
  - Scénario OpenVibe
- Impossible d'emprunter une électrode « oreille » car celles du laboratoire ont deux connecteurs
- Discussion à propos de l'application à un système réel

Informations :

FONCTIONNEMENT APPLICATIONS UNITY

Les applications développées par Alban dans le cadre de sa thèse fonctionnent sur deux ordinateurs. L'ordinateur A est destiné à l'acquisition des signaux EEG (et EMG), et exécute principalement les logiciels OpenVibe et MATLAB. L'ordinateur B est utilisé comme retour visuel pour les patients. Il exécute les applications développées sur le logiciel UNITY (Labyrinthe, Karting etc…). La communication entre A et B s'effectue à l'aide d'un câble Ethernet (il est important de bien paramétrer les deux ordinateurs afin d'avoir l'accès aux fichiers).

Attention : La version des fichiers donnée par Alban est faite pour fonctionner sur les deux ordinateurs du laboratoire. Pour faire fonctionner le tout, il faut lancer UNITY sur l'ordinateur B avec les fichiers sources. Dans les paramètres %VRPN%, il faudra rentrer l'adresse IP de l'ordinateur A puis recompiler le .exe associé.

L'ordinateur A doit exécuter l'OpenVibe acquisition server & l'OpenVibe Designer avec le scénario voulu. Pour l'ordinateur B, il faut d'abord lancer l'exécutable « UniVRPNityServer.exe » (dans le dossier VRPN2Unit), puis l'exécutable « appli.exe » (dans le dossier Beta1.4 – Dernière version du logiciel).

Une fois l'application UNITY lancée et le thème choisi, la touche H permet d'afficher le panneau de contrôle avec les différentes commandes possibles.

OPENVIBE

Le bon format de fichier à utiliser pour les acquisitions (tout du moins celui utilisé par Alban) est « .csv ». Il permet de facilement jongler entre acquisition / Matlab / OpenVibe.

Lors d'une acquisition avec OpenVibe server, il est important d'identifier et de nommer correctement les électrodes servant à l'acquisition. On peut retrouver les index dans l'interface d'acquisition.

Schéma classique d'un scénario BCI sous OpenVibe :

Etapes d'une application BCI