ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISEE
M.Ing.

PAR
Alexandre JUMELINE

UTILISATION DE SIGNAUX ÉLECTROMYOGRAPHIQUES POUR LE CONTRÔLE D'UN BRAS ROBOTIQUE POUR USAGERS DE FAUTEUILS ROULANTS MOTORISÉS.

MONTRÉAL, LE "DATE DU DÉPÔT AU BUREAU DES CYCLES SUPÉRIEURS"

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Rachid Aissaoui, directeur de mémoire
Département de Génie de la Production Automatisée

M. Prénom Nom, président du jury
département et institution

M. Prénom Nom, examinateur externe
département et institution

Mme. Prénom Nom, membre du jury
département et institution

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE "DATE DE SOUTENANCE"

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS

Ce projet a été réalisé en partenariat avec la société Canadienne Kinova dans le but d'ajouter une interface de contrôle à leur bras robotique Jaco contrôlable par le biais d'un Joystick.

Le but principal était de prendre la suite du projet de maîtrise de Véronique Maheu qui consistait en l'étude de la performance de différents classificateurs dans le cadre de la classification de mouvements à partir de huit canaux électromyographiques.

Le projet devait aussi se servir des paramètres cepstraux du signal électromyographiques. Pendant la réalisation, il s'est avéré que des caractéristiques du signal beaucoup plus simples permettent de discriminer suffisamment les mouvements effectués par l'utilisateur. De ce fait l'utilisation des paramètres cepstraux du signal a été abandonnée au profit d'une détection d'activité musculaire simple.

REMERCIEMENTS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque justo justo, porta sagittis feugiat eget, ornare rhoncus ligula. Nunc non odio sed lacus rutrum rhoncus. Mauris non congue arcu. Cras quis quam tortor. In ultrices tincidunt magna sed suscipit. Curabitur vel tellus sapien, ut tincidunt arcu. Maecenas dapibus ullamcorper urna, ut mollis mi tincidunt a. Nam eu orci nec lacus consectetur commodo. Donec purus tellus, consectetur at feugiat quis, scelerisque congue nibh. Aliquam urna dolor, congue nec euismod eget, convallis vitae libero. Sed vel magna suscipit leo suscipit porta quis et nunc. Nullam ante tellus, tincidunt a fringilla vel, rutrum non tellus. In volutpat consectetur purus, in euismod lorem feugiat vel. Aliquam sodales nisl eget sapien ullamcorper posuere consectetur orci bibendum. Vestibulum pulvinar viverra auctor. Vivamus ac sem et enim sodales dictum.

UTILISATION DE SIGNAUX ÉLECTROMYOGRAPHIQUES POUR LE CONTRÔLE D'UN BRAS ROBOTIQUE POUR USAGERS DE FAUTEUILS ROULANTS MOTORISÉS.

Alexandre JUMELINE

RÉSUMÉ

Les blessés médullaires de haut niveau lésionnel n'ont plus l'usage de leurs bras, mais conservent des capacités de mouvements résiduelles variant selon la hauteur de leur lésion. Ces capacités résiduelles concernent des mouvements restreints des doigts et des épaules principalement. Suivant le degré d'atteinte et la hauteur de la lésion de la moelle épinière, certaines personnes ne conservent pas suffisamment de liberté de mouvement dans leurs doigts pour utiliser ne serait-ce qu'un simple joystick. Pour ces personnes, les activités de la vie quotidienne (AVQ) deviennent des tâches très compliquées.

La société canadienne Kinova a conçu un bras robotique, JACO, adaptable à un fauteuil roulant motorisé afin d'aider les personnes à mobilité réduite à effectuer leurs actions de la vie quotidienne plus facilement. En effet, celui-ci permet la saisie d'objets ou d'ouvrir des portes de manière intuitive en n'utilisant qu'un Joystick. Cependant les personnes n'ayant plus suffisamment d'activité musculaire dans les doigts ne peuvent donc pas profiter de ce progrès. Le but de ce projet est d'adapter l'utilisation du bras JACO aux personnes blessées médullaires en utilisant les signaux électromyographiques (EMG) de 4 muscles résiduels au niveau du cou et de l'épaule.

L'acquisition des signaux EMG est faite avec des électrodes Delsys filtrées et pré-amplifiées en temps-réel à 2 kHz. Les électrodes sont placées sur quatre muscles présent chez les blessés médullaires de haut niveau : les deux trapèzes ainsi que les deux sternocléïdomastoïdiens. Ces muscles ont été choisis car ils permettent par leur seule activation de reconnaitre un mouvement donné : une élévation de l'épaule dans le cas des trapèzes et une rotation de la tête dans le cas des sternocléïdomastoïdiens.

L'énergie de Teager-Keiser est utilisée afin de déterminer si une activité musculaire est présente ou non, et chaque mouvement détecté peut être relié à une action du bras robot. Une calibration peut être effectuée pour chaque muscle afin de définir des seuils de détection personnalisés. Enfin un système de filtrage prédicitif temps réel a été intégré afin d'augmenter la sensibilité du système a des mouvements plus lents ou pour les personnes ayant moins de tonus musculaire.

Ce projet constitue une avancée supplémentaire dans l'aide technique à la manipulation apportée aux blessés médullaire dans l'accomplissement de leurs tâches de la vie quotidienne.

Mot-clés : Kinova, jaco, contrôleur EMG, temps-réel, blessé médullaire

TITRE ORIGINAL

Alexandre JUMELINE

ABSTRACT

High level spinal cord injury victims don't have the capacity to use their arms, but still have control of several muscles depending on the level of injury. Those residual capacities are mostly finger and shoulder movements. Depending of the level of the injury, some people don't have enough residual muscular activity to move their fingers in order to control a simple joystick. For those people, daily life activities can be very hard to achieve.

The Canadian corporation Kinova has developped a 7 degrees of freedom robotic arm, JACO, meant to be mounted on a motorised wheelchair in order to help spinal cord injury victims in their everyday's life. JACO can be used to grab objects or open doors only using a simple joystick. However, some poeople don't have enough musclar capacities to use this joystick, and therefor, can't use the arm. The goal of this project is to adapt the use of JACO to spinal cord injury victims using electromyographic (EMG) signals sampled on 4 muscles of the neck and shoulders.

EMG signal acquisition is performed using preamplified and filtered Delsys electrodes in real time at a sampling frequency of 2KHz. These electrodes are placed on 4 muscles still used by high level spinal cord injury victims : trapezius and sternocleidomastoid muscles. Those muscles were chosen beacause of their action in simple movements : shoulder elevation for trapeziuses and head rotation for sternocleidomastoids.

Teager-Keiser energy (TKE) is used in order to determine if a muscular onset is performed, and each move of the person can be linked to an action of the arm. A calibration can be done for each muscle in order to set a personalised detection threshold for onset detection. Finally, a real time predictive filter has been added in order to increase the sensitivity of the whole system and to be able to detect smaller and slower movements.

This project represent a step towards technical assistance brought to spinal cord injury victims and the improvement of their living conditions.

Keywords: Kinova, jaco, EMG control, real time, spinal cord injury

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

EMG	Électromyographique
BM	Blessé médullaire
AVQ	Activité de la vie quotidienne
TKE	Énergie de Teager-Keiser
ETS	École de Technologie Supérieure
CRC	Contrôle de redondance cyclique
SOF	Start of frame
EOF	End of frame
ALU	Unité arithmétique et logique

LISTE DES SYMBOLES ET UNITÉS DE MESURE

EMG	Électromyographique
α	FAM choice hyperparameter
B_t	New batch learning data block available that combines all available data up to a time t (*i.e.*,$D_1 \cup ... \cup D_t$)
β	FAM learning hyperparameter
c	Class input associated to **a** for supervised learning
C_k	Label of class k
$\|C_k\|_{\mathbf{LTM}}$	Maximal number of samples per class in the long term memory
D_t	New incremental learning data block available at a time t
D_t^{t}	Training data set at a time t
D_t^{v}	Validation data set at a time t
D_t^{f}	Fitness estimation data set at a time t
$\delta_{e_1 e_2}$	Particle diversity between two ensemble members e_1 and e_2
Δ	Distance from a local best particle within which no personal best value can be memorized by other particles among the swarm
$\Delta\theta_{e_1 e_2}$	Diversity between two classifiers e_1 and e_2 determined with a FAM specific indicator
e_1, e_2	Two ensemble members
EoFAM	An ensemble of fuzzy ARTMAP networks
ϵ	FAM match-tracking hyperparameter

$f(\mathbf{h})$	Objective function for a fuzzy ARTMAP hyperparameter vector **h** in a static optimization environment
$f(\mathbf{h}, t)$	Objective function for a fuzzy ARTMAP hyperparameter vector **h** and at a time t in a dynamic optimization environment
$f_e(\mathbf{h}, t)$	Objective function defined by the generalization error rate.
$f_o(\mathbf{h}_n, t)$	Objective function o (during MOO)
$f_s(\mathbf{h}, t)$	Objective function defined by the size of the F_2 layer (*i.e.*, number of F_2 layer nodes)
F^{ab}	FAM map field
F_1	FAM input layer
F_2	FAM competitive hidden layer
$FAM_{\text{estimation}}$	FAM network used to estimate fitness with the data set D_t^{f}
FAM_n	FAM network associated to the best position of particle n
$FAM_{n,o}$	FAM network associated to the best position of particle n for the objective o (during MOO)
FAM_{optimal}	FAM network with the highest accuracy obtained after optimization on a learning block D_t
FAM_n^{start}	FAM network that defines the initial state of the particle n prior learning data block D_t. During mono-objective optimization it corresponds to the best position of particle n, while it is associated with the current position of particle n during MOO.
FAM_{temp}	Temporary fuzzy ARTMAP network used during fitness estimation

INTRODUCTION

Avec l'avancée des technologies de l'information, le domaine du génie des technologies de la santé est aujourd'hui en mesure de venir en aide aux personnes présentant un handicap naturel, ou résultant d'une pathologie ou d'un accident, en procurant à celles-ci des dispositifs permettant l'amélioration de leurs conditions et leur qualité de vie. Ces dispositifs sont appelés les aides techniques.

Ce projet s'intéresse principalement aux aides techniques de type robotique développées pour aider les personnes paralysées et en fauteuils roulants motorisés à effectuer leurs activités de la vie quotidienne (AVQ).L'une d'entre elles, le bras robotique Jaco a été développé par la société canadienne Kinova, et est contrôlé par un joystick de manière à ce que son utilisateur puisse facilement effectuer des actions simples, comme l'ouverture d'une porte ou la préhension d'un objet.

Les blessés médullaires (BM) de haut niveau lésionnel n'ont plus l'usage de leurs bras, mais conservent des capacités de mouvements résiduelles variant selon la hauteur de leur lésion. Ces capacités résiduelles concernent des mouvements restreints des doigts et des épaules principalement. Suivant le degré d'atteinte et la hauteur de la lésion de la moelle épinière, certaines personnes ne conservent pas suffisamment de liberté de mouvement dans leurs doigts pour utiliser ne serait-ce qu'un simple joystick. Pour ces personnes, les AVQ deviennent des tâches très compliquées.

Le but du projet est donc de faire en sorte de diversifier les interfaces de contrôle du bras Jaco, pour permettre à des personnes incapables de le contrôler exclusivement au joystick de pouvoir avoir accès à cette avancée technologique majeure. Pour ce faire, l'utilisation du signal électromyographique (EMG) a été choisie.

Ce projet de maîtrise étant en grande partie de la recherche appliquée, une grande partie présentera la conception du système.

Une première partie de ce mémoire sera consacrée au fait de situer le projet dans l'univers scientifique actuel. Une mise en contexte sera premièrement effectuée dans le chapitre 1, suivie d'une revue de littérature et enfin de la problématique centrale du projet.

La partie principale du mémoire sera alors présentée dans le chapitre 4 : la méthodologie. Ce chapitre expliquera les choix matériels, logiciels et mathématiques effectués tout au long du projet. Toutes les étapes de conceptions seront détaillées ainsi que les intéractions entre les différentes parties du système.

Enfin, la dernière partie comprendra l'étape de validation du système conçu afin de déterminer l'efficacité du système et de valider la faisabilité d'un tel projet. Cette validation sera effectuée pars l'application d'un protocole de recherche et à travers une étude effectuée sur 10 sujets sains afin de récolter des données qualitatives permettant une courte étude statistique des performences du système.

En résumé cette étude présente la conception d'un système de contrôle d'une aide technique robotisée grâce aux signaux EMG d'une personne BM et à la validation d'une preuve de concept dans le but d'une potentielle commercialisation future par la societé Kinova.

CHAPITRE 1

MISE EN CONTEXTE

Les personnes BM sont aujourd'hui en mesure de pouvoir continuer à garder une autonomie partielle dans leur vie quotidienne grâce à la combinaison de plusieurs progrès technologiques. Les fauteuils roulants motorisés sont, par exemple, une part de ce progrès qui contribue à permettre au BM de garder une certaine mobilité, et sont le point d'encrage d'autres aides techniques dont fait partie le bras robotique Jaco.

Ce chapitre a pour but de présenter les conditions de vie des personnes BM d'aujourd'hui ainsi que les aides techniques fournies par l'industrie qui permettent d'améliorer leurs conditions quotidiennes de vie. Cette mise en contexte est nécessaire afin de discerner dans quelle mesure ce projet de maîtrise participe au processus d'aide aux AVQ des personnes BM.

1.1 Les Blessés Médullaires

1.2 Les aides techniques aux actions de la vie quotidienne (AVQ)

1.3 Le Signal Électromyographique (EMG)

CHAPITRE 2

REVUE DE LITTÉRATURE

La classification de mouvements grâce au signal EMG a été énormément étudiée dans la littérature. Toute fois, une partie seulement de ces études l'a traitée de manière temps réelle. Enfin un type d'application vient encore se démarquer dans ce sous ensemble : les applications embarquées.

Par application embarquée, nous entendons un système fonctionnant de manière autonome sur une carte électronique pouvant être déplacée facilement. Il faut donc enlever a ce sous ensemble toutes les applications réalisées sur ordinateurs.

La première partie de cette revue de littérature s'intéressera à la classification EMG en temps réel, tandis que la deuxième partie sera centrée sur les applications embarquées dont fait état la littérature dans son état actuel.

2.1 Classification EMG

L'analyse des signaux EMG et la détermination du mouvement effectuée à partir de ceux-ci se divise en plusieurs grandes sous-parties incluant chacune un algorithme bien précis. Chacune de ces parties peut être réalisée avec plusieurs algorithmes différents impliquant chacun une précision, et une facilité d'implémentation différente. Des compromis doivent donc être faits entre la précision engendrée par un algorithme et sa complexité d'implémentation et d'exécution qui est souvent inversement proportionnelle.

La première étape de traitement des signaux EMG consiste à déterminer quand il y a une activité musculaire et quand il n'y en a pas. Celle-ci est appelé détection d'activité, ou détection d'onset, et c'est la première partie qui sera analysée. L'étape suivante est l'extraction des caractéristiques du signal qui pourront permettre ensuite au classificateur de déterminer quel mouvement a été effectué. Enfin, une analyse de différents classificateurs sera effectuée,

se basant sur les mêmes critères que les autres algorithmes, à savoir la précision, la facilité d'implémentation et le temps d'exécution.

Un point important est à souligner, et ce point constituera un premier point de cette étude : tous ces algorithmes mis bouts à bouts doivent, une fois le choix d'architecture effectué permettre de réaliser le traitement de quatre canaux EMG en temps réel.

2.1.1 Fenêtrage du signal

Le signal électromyographique est découpé à des fins d'analyses et pour extraire les caractéristiques de celui-ci. La longueur de ces fenêtres est exprimée en temps ou en nombre d'échantillons. Un pas d'incrément est également défini, afin de choisir si les fenêtres se recouvrent ou non.

La littérature étudiée fait état de beaucoup de tailles de fenêtres et de pas d'incréments différents. La plupart sont fixés de manière heuristique. Cependant, Englehart et Hudgins (2003) évoquent le fait que cette fenêtre ne doit pas être trop grande de manière à ce que le temps de cette fenêtre et de son temps de traitement n'excède pas les 300ms de latence maximum au-delà desquels la latence est ressentie par l'utilisateur. Ainsi, leur fenêtre est fixée à 250ms avec un pas d'incrément à 125ms de manière à garder une marge suffisante pour le temps de calcul.

Une fenêtre trop petite est sujette au biais de variance généré par les variation du signal et de ces caractéristiques qui peuvent jouer sur la précision du classificateur utilisé.

Un compromis doit donc être fait sur la longueur des fenêtres utilisées entre la précision du classificateur et son temps de réponse.

2.1.2 Algorithmes de détection d'activité

Une fois une fenêtre du signal acquise, la première étape est la détection d'activité musculaire, ou détection d'"onset".

La détection d'onset est le principe qui consiste à détecter le début d'une activité musculaire sur le signal EMG de manière automatique. Pour ce faire, plusieurs algorithmes semblent bien fonctionner en combinant à la fois, une bonne précision et un nombre d'opérations flottantes à réaliser restreint. Cette détection d'onset permet d'économiser les ressources processeurs afin de n'extraire les caractéristiques du signal que lorsque c'est utile et surtout de délimiter les zones d'activité musculaire pour la détection de mouvement. Il existe ainsi plusieurs caractéristiques du signal qui peuvent être extraites afin de détecter un onset. Une première largement utilisée (Chang *et al.*, 1996; Englehart et Hudgins, 2003; Tenore *et al.*, 2007)est la méthode du Zero Crossing Rate (ZCR). Elle calcule le nombre de fois où la courbe du signal repasse par zero.

Un élément important est que Chang *et al.* (1996) ont développé cette méthode sur un DSP et semble fournir des résultats satisfaisants. Cette étude présente également la mise en place d'une dead zone sur l'algorithme de zero crossing, qui permet de réduire, voir de supprimer l'influence du bruit sur le ZCR. Le ZCR est une méthode simple à implémenter et très peu couteuse en ressources processeur pour la raison qu'elle ne nécéssite qu'un nombre d'addition égal N-1, N étant le nombre d'échantillons dans la fenêtre analysée.

Peleg *et al.* (2002) optent pour une méthode consistant à appliquer un seuil directement sur l'enveloppe du signal. Le signal est d'abord passé dans un filtre passe-haut à 30Hz, puis la valeur absolue du signal est faite, enfin, le signal est passé dans un filtre passe-bas à 2.5Hz. Cette méthode est une analogie de la méthode Mean Absolute Value (MAV), le filtre passe bas ayant pour but d'effectuer une moyenne des valeurs absolues du signal. Suite à cela, le seuil appliqué sur l'enveloppe est fixé comme étant 10 pourcents plus haut que la valeur de l'enveloppe de l'EMG au repos.

Tenore *et al.* (2007) utilisent également la méthode MAV sans toutefois préciser la manière dont ils ont choisi leur seuil de détection.

Une autre méthode simple à implémenter et peu couteuse en temps de calcul est la méthode de la Teager-Kaiser Energy (TKE) qui correspond à un taux de variation du signal. Elle a été

utilisée par Maheu (2011) comme une caractéristique du signal utilisée dans la classification, mais peut aussi être utilisée dans la détection d'onset comme l'ont montré Li *et al.* (2007). Elle ne constitue qu'une soustraction et deux multiplications par échantillon dans la fenêtre d'analyse.

2.1.3 Caractéristiques du signal EMG utilisées

Une fois l'ctivité musculaire du signal EMG détectée, on cherche à déterminer quel mouvement a été effectué. Ce travail est effectué par un classificateur. Ce classificateur doit donc prendre des éléments de comparaison en compte pour déterminer l'action faite. Ces éléments doivent être significatifs du signal capté, et sont appelés des features, ou des caractéristiques. Les caractéristiques extraites des signaux EMG sont nombreuses, aussi nous nous concentrerons sur celles ayant les meilleurs résultats dans la littérature. Ainsi la caractéristique ayant le plus de succès semble être l'auto-régression (Chang *et al.*, 1996; Peleg *et al.*, 2002; Maheu, 2011).

L'auto-régression consiste à trouver les n+1 coefficients d'un polynôme d'ordre n qui, appliqué sur un signal aléatoire, permettra de retrouver une approche du signal réel. Plusieurs algorithmes permettent de retrouver ces coefficients autorégressifs (AR).

Peleg *et al.* (2002) utilisent une auto-régression d'ordre 11. Cet ordre « a été déterminé de manière heuristique » (Peleg *et al.*, 2002).

Chang *et al.* (1996) ont choisi l'utilisation des coefficients AR mais également les coefficients cepstraux obtenus à partir des AR. Ils préconisent également de fixer l'ordre de l'AR à 4 en se basant sur la littérature et en justifiant par un compromis entre une bonne représentation du signal et un temps de calcul court. L'obtention des coefficients cepstraux à partir des coefficients AR se fait grâce à une relation mathématique récursive simple dépendant de l'ordre de l'auto-régression réalisée.

2.1.4 Classificateurs utilisés dans la détection de mouvements

Une fois les caractéristiques du signal calculées, celles-ci sont entrées dans un classificateur qui, grâce à un apprentissage, permet de déterminer quel mouvement a été effectué par le sujet. Encore une fois, beaucoup d'algorithmes existent pour réaliser cette opération. Toutefois, beaucoup de ceux-ci sont très consommateurs en temps de calculs, comme les réseaux de neurones (RN), où les analyses linéaires discriminantes (LDA).

Toute fois Englehart *et al.* (1999) ont montré que l'importance du choix du classificateur pour la reconnaissance de mouvements à partir de signaux EMG est limitée et que les gains en termes de performances de classificateurs complexes tels que les RN sont négligeables par rapport aux performances d'un classificateur plus simple comme le LDA.

Ceci étant dit nous nous intéresserons donc plus particulièrement aux deux études principales mettant en œuvre des dispositifs temps réels embarqués.

Chang *et al.* (1996) utilisent dans leur application embarquée un algorithme utilisant les distances des maximums de vraisemblances (MMLD), tandis que Tenore *et al.* (2007) utilisent quant à eux un classificateur de type LDA.

2.2 Systèmes Embarqués temps réels et EMG

La littérature comprend très peu d'applications embarquées de traitement temps réel de signaux EMG. Ainsi un premier article de référence a été publié par Chang *et al.* (1996) et rapporte la réalisation du traitement de 4 électrodes EMG afin de contrôler un pointeur informatique dans le but de remplacer une souris. Cette étude fait référence dans notre projet car elle a été développée dans le but de venir en aide aux blessés médullaires de haut niveau.

Deux études de la même époque traitent ensuite également de traitement embarqué des EMG. Martinez *et al.* (1999) utilisent un dsp, sans toutefois préciser lequel, pour traiter un canal EMG tandis Prasad *et al.* (1996) ont conçu une carte à base de DSP implantée dans un ordinateur pour acquérir et traiter cinq canaux EMG.

Enfin l'étude la plus récente est celle de Tenore *et al.* (2007) qui présentent dans leur article avoir réalisé l'une des premières applications embarquées classifiant le signal EMG.

2.2.1 Le traitement temps réel

Le temps réel au sens strict du terme en informatique, signifie que l'on doit savoir à tout instant combien de temps prendront les différentes actions réalisées par un programme. Ce temps peut s'exprimer de plusieurs manières, les deux principales étant le temps en secondes et le nombre d'instructions élémentaires d'une unité de calcul (microprocesseur, microcontrôleur).

Dans notre cas, et comme fixé dans la littérature du contrôle par EMG, la contrainte principale est le temps de réaction et de calcul maximal à laquelle le système doit satisfaire. La littérature traitant de l'implémentation temps réelle de tels systèmes parle ainsi d'un délai de calcul maximal de 300 millisecondes afin que l'utilisateur humain ne ressente pas la latence du système (Chu *et al.*, 2007; Englehart et Hudgins, 2003).

2.2.2 Méthodes d'acquisition du signal

La plupart des études utilisées pour cette revue utilisent des électrodes de surface pour faire leur reconnaissance de mouvements. Cependant, les manières de filtrer et d'échantillonner sont différentes à travers les études. Cette partie a pour but de les regrouper et de les comparer.

La plupart des études observent une fréquence d'échantillonnage de 1KHz permettant d'obtenir une bande utile théorique (respectant le théorême de Shannon) pour les signaux EMG allant jusqu'à 500Hz (Chang *et al.*, 1996; Chu *et al.*, 2007; Englehart et Hudgins, 2003; Tenore *et al.*, 2007). La plupart des signaux EMG bruts sont amplifiés avec des gains de l'ordre de 2000, sont passé dans des filtres passe bande entre 10 et 450 ou 500 Hz.

Peleg *et al.* (2002) ont toutefois opté pour une fréquence d'échantillonnage à 500Hz, avec un filtre anti repliement à 250Hz, alors que Crawford *et al.* (2005) ont opté pour une fréquence d'échantillonnage à 2048 Hz. Lorsqu'elles sont mentionnées, la précision des ADC utilisés

varie suivant les études, allant de 16 bits (Englehart et Hudgins, 2003) à 12 bits (Tenore *et al.*, 2007; Fukuda *et al.*, 2003; Prasad *et al.*, 1996).

2.2.3 Matériel informatique utilisé dans la littérature

La plupart des application trouvées dans la littérature n'ont pas une définition toujours précise de la notion de temps réel. La plupart des articles présentent un système exécuté sur des machines puissantes, sur des systèmes d'exploitation souvent non temps réels.

Englehart et Hudgins (2003) ont réalisé une implémentation temps réelle d'algorithmes de reconnaissance de mouvement à partir de signaux EMG. Cette réalisation a été réalisée sous matlab et son extension de programmation temps réel sur un PC distant.

Si l'on s'interesse plus particulièrement aux applications embarquées, Chang *et al.* (1996) ont réalisé un montage électronique basé sur un DSP TMS320C31 de chez Texas Instruments. C'est un processeur 32bits comprenant un cœur à virgule flottante tournant à une fréquence de 20MHz, avec quatre niveaux de pipe-line permettant d'exécuter une instruction à chaque cycle d'horloge. Cette architecture leur a permis d'obtenir un temps total de traitement du signal entre 140 et 160 ms ce qui satisfait à la condition évoquée au début de l'étude concernant le respect d'un temps de réponse maximum de 300 ms. Toutefois, cette étude ayant été réalisée en 1996, il existe aujourd'hui quantité de DSP plus rapide et plus efficaces.

Toute fois cette étude, ainsi que celles de Martinez *et al.* (1999) et de **?** ont aujourd'hui autour d'une quinzaine d'années et le matériel informatique utilisé alors a aujourd'hui beaucoup évolué.

Tenore *et al.* (2007) ont utilisé également un DSP de chez Texas Instruments : un TMS320VC5509A couplé à un ADC dédié de chez National Instruments. Ce DSP est un processor comprenant un cœur à virgule fixe exécutant des actions relativement simples, comme le calcul du ZCR du signal, le calcul de la longueur d'onde du signal et le MAV. Ce DSP leur a permis d'implémenter un classificateur LDA.

CHAPITRE 3

PROBLÉMATIQUE ET OBJECTIFS DU PROJET

CHAPITRE 4

MÉTHODOLOGIE

4.1 Choix des Muslces

Le premier choix important est de choisir quels muscles seront utilisés pour réaliser la commande. Dans un premier temps, le critère principal est que les muscles choisis doivent toujours être fonctionnels chez les blessés médullaires. Ainsi, les muscles choisis par Chang *et al.* (1996), les trapèzes et les sterno-cléïdo mastoïdiens sont considérer comme un choix judicieux dans un premier temps. En effet, ces deux muscles sont innervés en partie par des nerfs passant par la seconde vertèbre cervicale, ainsi, la plupart des blessés médullaires conserveront au moins une activité résiduelle dans ces muscles.

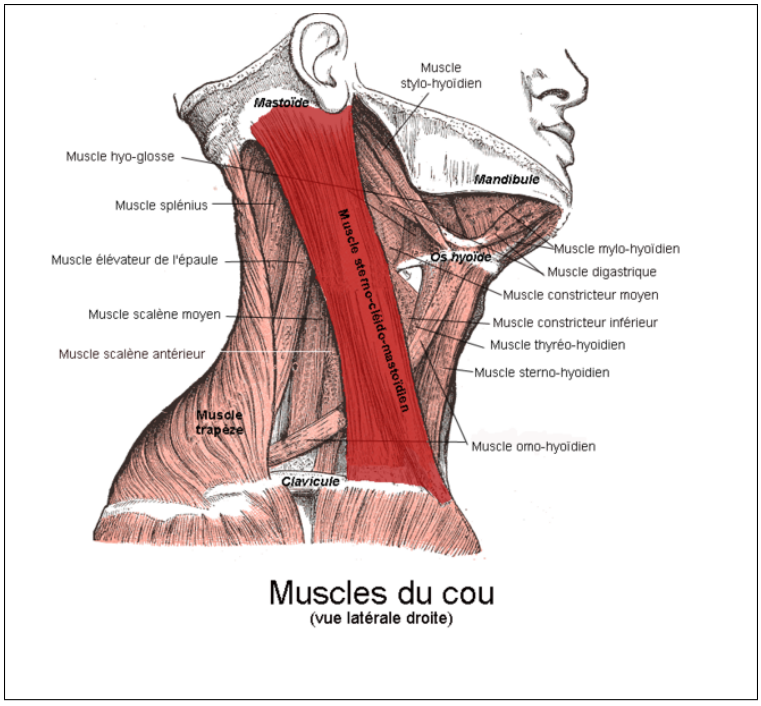

Figure 4.1 Muscles sterno-cléïdo mastoïdien et muscles du cou. par Berichard (travail personnel d'après Gray's Anatomy) [GFDL (http ://www.gnu.org/copyleft/fdl.html) ou CC-BY-SA-3.0-2.5-2.0-1.0 (http ://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Un autre muscle à considérer serait également le muscle occipito-frontal qui opère l'élévation des sourcils. Toute fois, il est peu probable que les personnes blessées médullaires apprécient le fait d'avoir des électrodes collées sur le front en permanence. Ce muscle ne représente donc pas une alternative viable pour le projet, de même que pour tous les autres muscles faciaux.

Une autre solution serait d'utiliser les muscles fléchisseurs et extenseurs du poignet. Toute fois, ces muscles sont innervés respectivement par les nerfs médian et radial dont les faisceaux passent par les dernières vertèbres cervicales ainsi que la première vertèbre thoracique. Ainsi, les blessés médullaire de haut niveau n'auront plus suffisamment, voir plus du tout d'activité résiduelle dans ces muscles.

Un point vient également appuyer la sélection des muscles trapèzes et sterno-cléïdo mastoïdiens et qui va influencer la manière de développer le projet : l'acquisition de signaux EMG sur ces muscles n'implique pas de "cross-talk" c'est à dire qu'on ne voit pas l'activité d'un muscle sur l'enregistrement des autres. Cette observation a été réalisée au début de la réalisation du projet, après avoir effectué des tests préliminaires. Ainsi, en faisant des mouvements simples de rotation de la tête et d'élévation des épaules, on ne voit qu'un seul muscle s'activer. Cette considération permet de faire un choix drastique sur la complexité du projet : on choisit de ne monitorer que des mouvements unitaires simples n'activant qu'un seul muscle.

Ainsi, on est capable d'établir un lien direct entre l'activité musculaire d'un muscle particulier et le mouvement réalisé. Lorsqu'une activité est détectée sur un muscle sterno-cléïdo mastoïdien, c'est qu'une rotation de la tête a été effectuée, et si une activité est détectée sur un trapèze, c'est une élévation de l'épaule qui a été réalisée.

Ce postulat permet de réduire le temps de calcul nécessaire au traitement des signaux en enlevant complètement l'étape de classification. Ce choix comporte des points négatifs et des points positifs. Un atout majeur est de retirer toute la charge de calculs nécessaire pour le classificateur, et de concentrer ces calculs sur la détection d'activité musculaire ainsi que sur le filtrage numérique du signal. En contrepartie, le nombre de mouvements reconnus par le systèmes est réduit à quatre mouvements unitaires : un pour chaque muscle.

Cependant cette perte en diversité de mouvements pourra être en partie comblée par des stratégies de communication différentes pour ces mouvements. En effet, on pourra effectuer des mouvements prolongés ou bien des mouvements composés de plusieurs clics, ou activations courtes répétées. Ces stratégies permettront d'étendre la diversité de commandes tout en ne reconnaissant que quatre mouvements.

4.2 Architecture Matérielle

Une fois le choix des muscles arrêté, les premiers choix a effectuer seront les choix architecturaux, et dans un premier temps, ceux concernant l'architecture matérielle. Ce chapitre présente les choix matériels effectué pour la conception du système tout en argumentant pourquoi chaque choix a été effectué et dans quel cadre.

4.2.1 Électrodes

Le choix des électrodes est une phase importante du projet car leur choix détermine la charge de travail qui va en résulter. En effet, des électrodes donnant un signal brut, très faible (la tension d'un signal EMG étant de l'ordre du millivolt) est impossible a exploité tel quel et nécessite des étape de filtrage et d'amplification avoir de pouvoir être échantillonné. Ainsi choisir ce type d'électrode ajoute une étape de conception électronique supplémentaire afin que le signal soit exploitable.

Le LIO travaille depuis maintenant plusieurs années avec des électrodes construites par la compagnie Delsys : les électrodes DE-2.3 et se présentent comme sur la figure 4.2.

Ces électrodes, contrairement au modèle moins cher DE-2.1 sont pré-amplifiées avec un gain de $1000V/V \pm 1\%$ et pré-filtrées sur une bande passante de $20 - 450Hz \pm 10\%$. Ceci nous permet de limiter les étapes de filtrage analogique avant de faire entrer le signal de l'électrode dans le microcontrôleur. En effet, le filtrage passe-bande déjà présent permet de garder la bande utile du signal EMG tout en limitant le bruit d'échantillonnage.

Figure 4.2 Électrode Delsys DE-2.3.

Les électrodes sont câblées avec du fil blindé, protégeant des interférences électromagnétiques, le blindage étant relié à la masse du circuit. Pour que ce blindage soit efficace, le connecteur utilisé doit être également de bonne qualité. Pour ce faire, des connecteur de marque LEMO sont utilisés. Ces connecteurs sont utilisés dans beaucoup de domaines critiques demandant des connecteurs résistants au temps et aux interférences.

Ainsi les connecteurs LEMO femelle correspondants à ceux présents sur les câbles des électrodes ont été acquis afin de relier les électrodes au système.

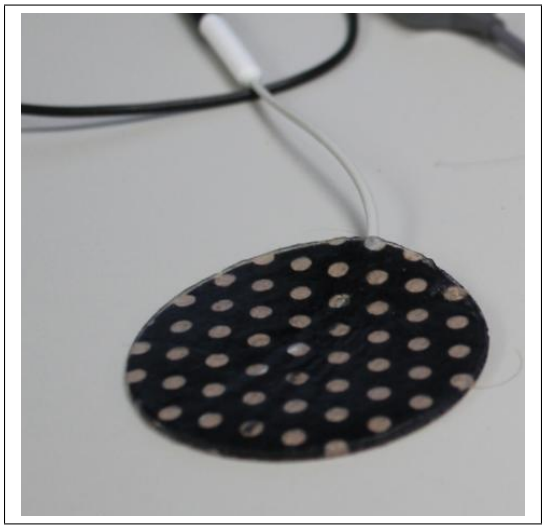

Figure 4.3 Électrode Delsys neutre.

Plusieurs électrodes neutres, comme présentée par la figure 4.3, ont également été acquises. Ces électrodes sont nécessaire car c'est la référence utilisée pour mesurer la tension acquise sur les muscles. Elles seront positionnées sur une partie la moins charnue possible, comme par exemple la pointe du coude de l'utilisateur. Elles seront connectées directement à la référence du convertisseur analogique-numérique du microcontrôleur.

4.2.2 Microcontrôleur

Le deuxième choix matériel important concerne celui du microcontrôleur ou du contrôleur de signaux numériques (DSP). En effet, ce choix est déterminant car c'est lui qui opèrera tous les calculs nécessaires au bon fonctionnement du système. Son choix est donc à faire de manière réfléchie.

Avant de présenter le microcontrôleur choisi pour le projet, les différents critères de choix seront décrits dans un premier temps.

Un premier critère est le nombre d'opérations élémentaires par seconde effectué par le calculateur. En effet, le calculateur doit être capable d'effectuer les calculs et les algorithmes choisis pour la stratégie de reconnaissance de mouvement. Ce critère peut se mesurer de deux manières. La première est la fréquence de l'horloge principale du calculateur. Toute fois, même si celle-ci est un bon indicateur, suivant l'architecture du microcontrôleur, un processeur avec une horloge plus rapide qu'un autre peut parfois effectuer moins d'opérations par seconde que le plus lent. Ceci vient du fait que certains calculateur ont besoin de deux cycles d'horloge pour effectuer une opération, l'opération devant d'abord être chargée, puis exécutée. Certains microcontrôleurs parallélisent ces actions à l'aide d'un "pipeline" qui pendant qu'il exécute une opération, "précharge" l'opération suivante. Une autre unité permettant d'exprimer la rapidité de calcul d'un processeur est le nombre de millions d'instructions par seconde (MIPS). Cette valeur fait abstraction de la manière dont fonctionne le processeur, et donne directement le nombre de calculs par unité de temps que peut soutenir le composant.

Une autre caractéristique concerne l'unité arithmétique et logique (ALU) du calculateur. Celle-ci peut être dite soit "à virgule fixe", soit "à virgule flottante". Une variable à virgule flottante

est une variable permettant de stocker un nombre réel dans lequel on peut déplacer la place de la virgule, tout en ne changeant pas le nombre grâce à un exposant. C'est un système similaire à l'écriture scientifique mais appliqué à l'informatique. Ceci permet pour une taille de variable donnée en octets d'optimiser la précision d'un nombre flottant.

La mémoire vive disponible dans le microcontrôleur est également un critère de sélection. En effet, dans le cas d'un fenêtrage de signal de $300ms$ échantillonné à $2KHz$, les données d'une fenêtre représentent 600 échantillons, qui, stockés dans des entiers de 16 bits représentent une quantité de mémoire de 1200 octets, soit $1.2Ko$. Sachant qu'il faut stocker une fenêtre pour la traiter et en acquérir une autre en même temps, le double de cet espace est nécessaire pour le traitement des fenêtres.

$2.4Ko$, mis à l'échelle des processeurs d'aujourd'hui ne représentent que très peu de données. Toute fois, pour un microcontrôleur, cet espace représente une contrainte forte. Contrairement à un microprocesseur qui utilise de la mémoire vive externe, la mémoire vive utilisée par un microcontrôleur est interne, et la taille de la mémoire disponible est donc très réduite.

Un élément supplémentaire de choix du contrôleur est le nombre et la diversité des entrées et sorties qu'il propose. En effet, le microcontrôleur doit avoir au moins quatre entrées analogiques auxquelles se connecteront les électrodes. Le système final devant également être intégrable au sein même de Jaco, et celui-ci utilisant un bus CAN (controler area network) dans ses communications interne, le microcontrôleur doit avoir une interface CAN. D'autres périphériques de communications comme des ports série seront également nécessaire.

Comme dit précédemment le système a pour but d'être intégré au bras Jaco, de ce fait, l'encombrement et la consommation énergétiques sont également un critère à prendre en compte.

Enfin le dernier élément à prendre en compte est la dépense nécessaire pour mettre en place la solution choisie. Dans un premier temps, l'existence d'une carte de développement préconçue permet de commencer directement le développement du programme sans avoir à faire de carte électronique. Les logiciels et outils de compilation entrent également en ligne de compte, car ceux-ci sont souvent très coûteux. Le fait que ces outils soient également disponibles sur plu-

sieurs systèmes d'exploitation est également un argument de choix, bien que moins important que tous ceux cités précédemment.

On étudie donc la classe de microcontrôleurs souvent utilisée dans la littérature : la classe TMS320 de Texas Instruments. Ces microcontrôleurs sont divisés en plusieurs gamme présentées dans le tableau 4.1.

Tableau 4.1 Gammes de microcontrôleurs/DSP TMS320 de Texas Instrument

	C5000	C6000	Keystone Multicore
Horloge	jusqu'à 300MHz	jusqu'à 1GHz	jusqu'à 16GHz
Mémoire vive	320Ko	450Ko	DDR3 externe
Prix du composant seul	$2 - $10	$5 - $25	$30 - $160

Les calculateur de type TMS320 sont des composants extrêmement gros et coûteux qui permettent d'effectuer des calculs très poussés à grande vitesse. Toute fois, les prix des cartes d'évaluation varient entre 200 et 500 dollars, et les suites de développement logiciel ont un prix allant d'environ 800 à 8000 dollars en fonction des options.

Il en va de même pour les calculateurs de chez Texas Instruments moins performants, les composants ont de bonnes caractéristiques mais nécessitent l'utilisation d'une suite logicielle coûteuse.

Le LIO possédait depuis quelques mois une exemplaire d'un microcontrôleur PIC32 produit par la societé Microchip ainsi que les outils hardware de programmation et de debug fournis avec celui-ci. Ce microcontrôleur répond aux critères de choix pour le projet : il possède un ADC 10 bits qui sera suffisant pour le projet et pouvant aller jusqu'à 16 canaux. Il tourne à une cadence de 80 MIPS et dispose de beaucoup de périphériques de communication dont plusieurs liaisons séries ainsi que qu'un bus CAN.

Ce microcontrôleur possède un cœur à virgule fixe, et intègre également dans son jeu d'instruction des instructions DSP flottante incluant divers outils dont les FFT. La version utilisée

possède également 128Ko de mémoire vive afin de pouvoir stocker une grande quantité de donnée.

Un autre avantage du PIC32 est que la suite logicielle permettant son développement peut être utilisée dans un premier temps en version gratuite. Cette version n'inclut pas les options d'optimisation de code pendant la compilation, mais ces options ne sont pas indispensable à la réalisation du projet.

Pour des raisons économiques et pratique, le PIC32, comme présenté sur la figure **??** a donc été choisi pour réaliser le système.

Le modèle choisi pour le projet est le PIC32MX795F512L. C'est le modèle le plus rapide et le plus gros de la gamme. C'est également celui possédant le plus d'entrées et sorties différentes, ainsi que la quantité de mémoire vive la plus grande des microcontrôleurs produits par Microchip.

Ce modèle a été choisi également car il est commercialisé par Microchip sur une carte de développement peu coûteuse. De plus cette carte de développement se présente sous la forme d'une petite carte pouvant être montée sur une carte fille permettant de connecter simplement d'autres composants aux entrées/sortie du PIC32. Le système complet est présenté sur la figure **??**.

4.2.3 Adaptation des Tensions

Les électrodes étant alimentées en $+5V/-5V$, et le PIC32 fonctionnant à $3.3V$, il est nécessaire d'adapter les signaux des électrodes de $+5V/-5V$ à $0V/3.3V$ afin de ne pas créer de sur-tension dans le microcontrôleur. Pour ce faire, un montage composé de deux étages d'amplificateurs opérationnels est nécessaire. Les amplificateurs utilisés pour cette réalisation sont des composants intégrés de type LM741CN. Ces composants, bien que peu récents ont l'avantage d'avoir un comportement aujourd'hui bien maîtrisé et connu.

Le premier étage est un montage inverseur ayant un gain de $1/3$. Ce montage est présenté par la figure 4.4.

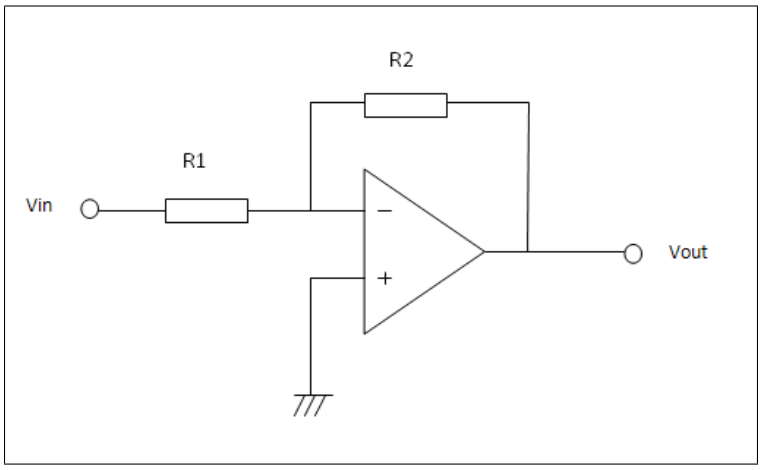

Figure 4.4 Étage inverseur utilisé dans l'adaptation de tension.

La fonction de transfert de cet étage d'amplification peut être calculée par l'équation 4.1. Ainsi, on fixe respectivement les valeurs des résistances $R1$ et $R2$ à $11Kohm$ et $3.3Kohm$.

$$Vout = -Vin \times (\frac{R2}{R1}) \tag{4.1}$$

Cet étage permet théoriquement d'amener la tension entre $-5V$ et $+5V$ à une tension comprise entre $-1.5V$ et $+1.5V$. En pratique, à cause des variation de valeurs des composants, et des caractéristiques de l'amplificateur opérationnel utilisé (imperfections statiques, tension d'offset), la tension est ramenée entre $-1.75V$ et $1.5V$.

Le deuxième étage est un offset réalisé par un montage sommateur inverseur permettant de repasser la plage de tension dans un intervalle positif. Le montage est décrit par la figure 4.5.

La fonction de transfert de cet étage d'amplification peut être calculée par l'équation 4.2.

$$Vout = -R3 \times (\frac{V1}{R1} + \frac{V2}{R2}) \tag{4.2}$$

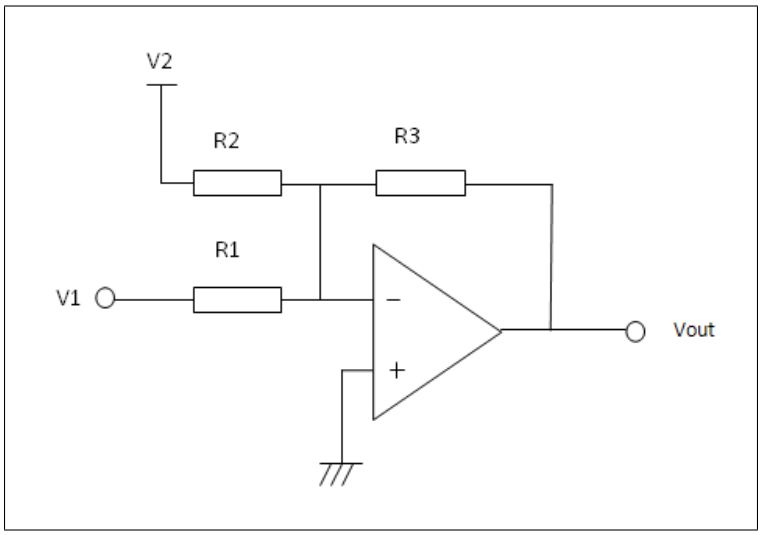

Figure 4.5 Étage sommateur utilisé dans l'adaptation de tension.

Soit, si $R1 = R2 = R3$:

$$Vout = -(V1 + V2) \tag{4.3}$$

Ainsi, V1 représente la sortie du premier étage d'amplification et V2 la tension d'offset à ajouter à V1 pour faire passer toute la plage de tension dans les tension négatives (en théorie donc -1.5V). Enfin le facteur -1 généré par la partie inverseuse du montage permet de repasser l'intervalle de tension dans les tensions positives uniquement.

Le plus important pour le signal final est de ne pas dépasser $3.3V$ pour les crêtes hautes et de ne pas passer sous les $0V$ pour les crêtes basses. Le signal doit également être centré par rapport à cette plage de tension, et donc être centré aux alentours de $1.75V$.

La figure 4.6 présente le schéma électronique utilisé sur chaque canal EMG avant de le connecter au PIC32.

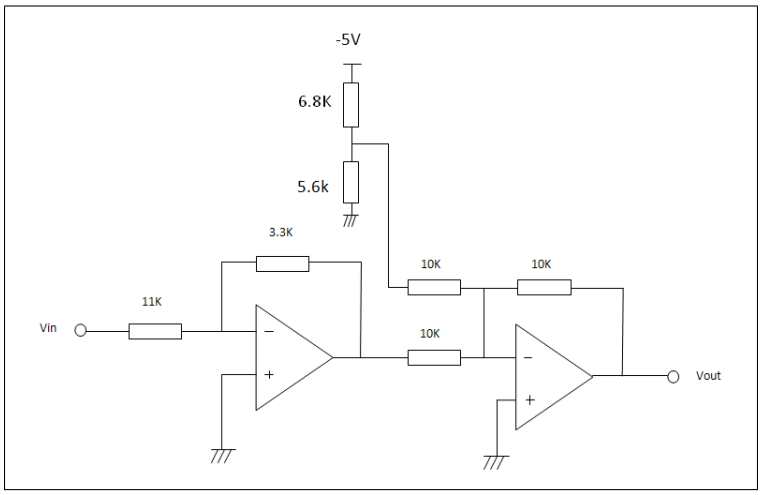

Figure 4.6 Circuit électronique d'adaptation de la tension de sortie des électrodes.

Cependant il est à souligner que la valeur exacte des composants électronique varie suivant leur tolérance (5% pour les composants utilisés ici). Ainsi le voltage correspondant au 0 peut ainsi varier (1.72V, 1.68V, ...) et un offset sera donc présent sur les valeurs enregistrées par le convertisseur du PIC.

Le schéma 4.6 présente les valeurs de composants pratiques pour la réalisation du circuit. Celles-ci sont définies en tenant compte des imperfections statiques des amplificateurs et des variations des composants.

4.2.4 Liaison à l'Ordinateur

L'architecture matérielle présentée au chapitre 4.2a présentée la manière dont sont reliées les différentes composantes physiques du système. Cette partie a pour but de présenter la liaison série réalisée entre la carte électronique du PIC32 et l'ordinateur utilisé pour visualiser les signaux temps réels.

Dans un premier temps, les niveaux de tensions du signal série délivré par le PIC32 ne sont pas compatibles avec ceux utilisés et transmis par l'ordinateur. En effet, le PIC fournit un signal entre 0V et +3.3V alors que les signaux du port série d'un ordinateur sont compris entre +12V et -12V.

Afin de faire la transition entre ces deux niveaux de tension, un composants électronique intégré est utilisé : un MAX3232, présenté sur la figure 4.7.

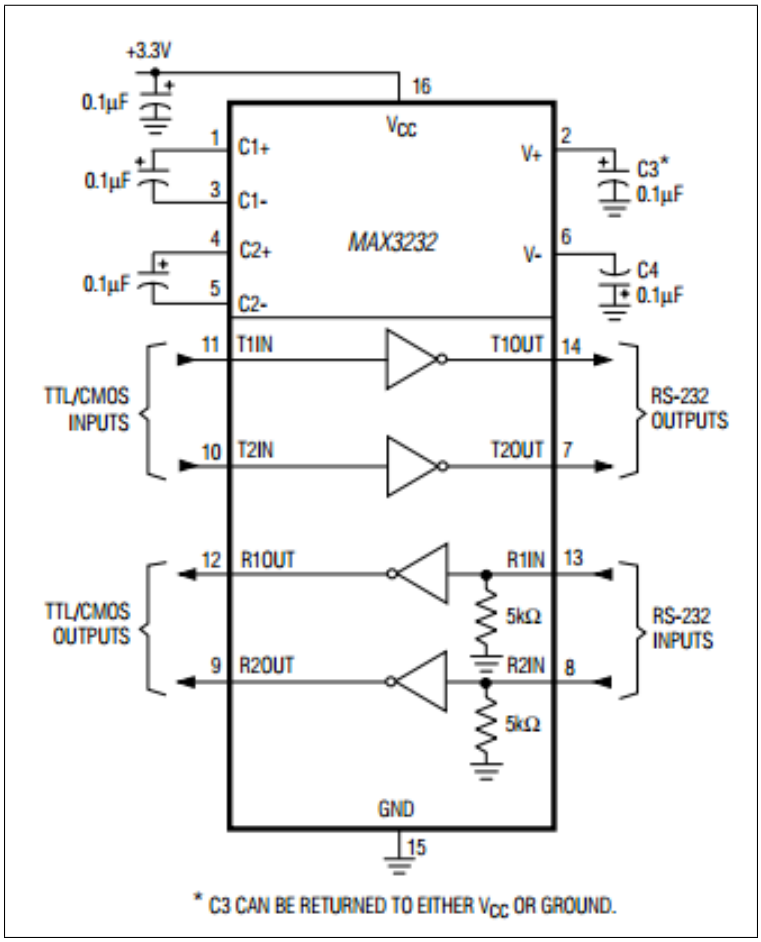

Figure 4.7 Circuit de câblage type du composant max3232, extrait de la documentation officielle du composant produit par Maxim Integrated Products Inc (2007).

Une fois la liaison physique réalisée, un protocole de transmission de données a été mis en place. En effet, les données arrivant sur le port série en RS-232 étant traitées octet par octet, il a fallu mettre en place une trame de données afin que celles-ci soit réceptionnées correctement, et permettant de vérifier que toutes les données reçues sont bien intègres. Le début de trame (SOF, Start of Frame) est constitué de 4 octets fixes, ne changeant jamais de valeur. Ceci constitue une protection contre une mauvaise détection de début de trame. En effet, la même séquence de quatre octets est très improbable de se présenter en tant que données, et de ce fait on aura peu

de chance de détecter un mauvais début de trame. Viennent ensuite les données. La longueur en octets des données à transmettre sur la liaison série est facilement configurable et modifiable à la fois dans le programme du PIC32 et dans le programme de l'application de monitorage sur l'ordinateur. Ensuite, les deux octets suivants concernent le contrôle de redondance cyclique (CRC) qui permet de vérifier que la trame reçue n'a pas perdu d'information, ou n'a pas vu un de ses octets corrompu lors du transfert. Le CRC est une valeur sur 2 octets obtenue grâce à un calcul récursif simple dont l'opération mathématique principale est une division modulo 2 dont le reste est le CRC.

Le CRC est calculé en prenant pour entrée uniquement la partie données de la trame. A la réception d'une nouvelle trame, le CRC des données est calculé, puis comparé au CRC transmis dans la trame. Si les deux CRC sont identiques, c'est que la trame est intègre, sinon, la trame est considérée comme non valide, et les données ne sont pas enregistrées, car corrompues.

Enfin, la fin de trame (EOF, End of Frame), est composé de 2 octets fixes qui ne changent pas non plus de valeur afin de détecter quand une fin de trame est réceptionnée. La figure 4.8 présente le schéma global de la trame utilisée dans le cadre de la liaison série.

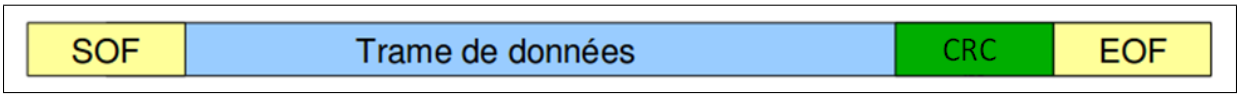

Figure 4.8 Format de la trame utilisée dans le cadre de la liaison série.

Cette trame est utilisée autant dans les données allant du PIC32 vers l'ordinateur que pour les commandes de calibration envoyée au PIC32 par l'ordinateur.

4.2.5 Liaison à Jaco

Le fonctionnement interne de Jaco utilise un bus de type CAN comme mentionné dans le paragraphe 4.2.2. Le bus CAN est un protocole de communication très répandu dans l'industrie et popularisé à l'origine par le secteur automobile.

C'est un bus série opérant en mode différentiel, c'est à dire que deux fils transportent le même signal de manière complémentaire, permettant un contrôle de l'intégrité des données. Ce bus comprend également une couche protocolaire définissant des types de messages à envoyer à une adresse sur le bus.

Dans un premier temps, il était impossible d'implanter directement le prototype du système directement dans le bras Jaco, ou même d'y faire rentrer des câbles pour s'y connecter.

Kinova livre avec le bras robotique des librairies de programmation permettant à un programme fait par un utilisateur de se connecter à Jaco et de lui envoyer des commandes ou des fichiers de configuration.

Le système au stade de prototype étant connecté un ordinateur par une liaison série, il est aisé de se servir de l'ordinateur comme intermédiaire entre le système de détection de mouvement et le bras Jaco lui-même.

De ce fait, le programme de monitorage des données du système doit également s'interfacer avec Jaco de manière à lui transférer des commandes en fonction des mouvements détectés. Cette connexion est effectuée grâce à la liaison USB disponible sur Jaco et utilise les librairies de programmation fournie par Kinova.

4.2.6 Schéma matériel Global

La partie 4.2 présente les différentes composantes nécessaire à la réalisation matérielle du projet. Cette partie a pour but de synthétiser tous les choix effectuer de manière à donner du recul et un point de vue global du prototype entier.

Détaillons dans un premier temps les entrées et sorties du PIC32.

Contrairement à beaucoup d'autres microcontrôleurs plus petits, le PIC32 ne possède pas d'entrées/sorties qui soit re-localisables. C'est à dire que l'on n'est pas en mesure de choisir quel périphérique prend ses entrées et sorties sur des pattes spécifiques du composants. Ceci implique donc des contraintes de câblage pour relier les électrodes et la liaison série sur celui-ci.

Ainsi on voit sur la figure 4.9 que les quatre électrodes sont reliées via l'adaptation de tension au entrées anaologiques AN1, AN2, AN3 et AN4 du PIC32.

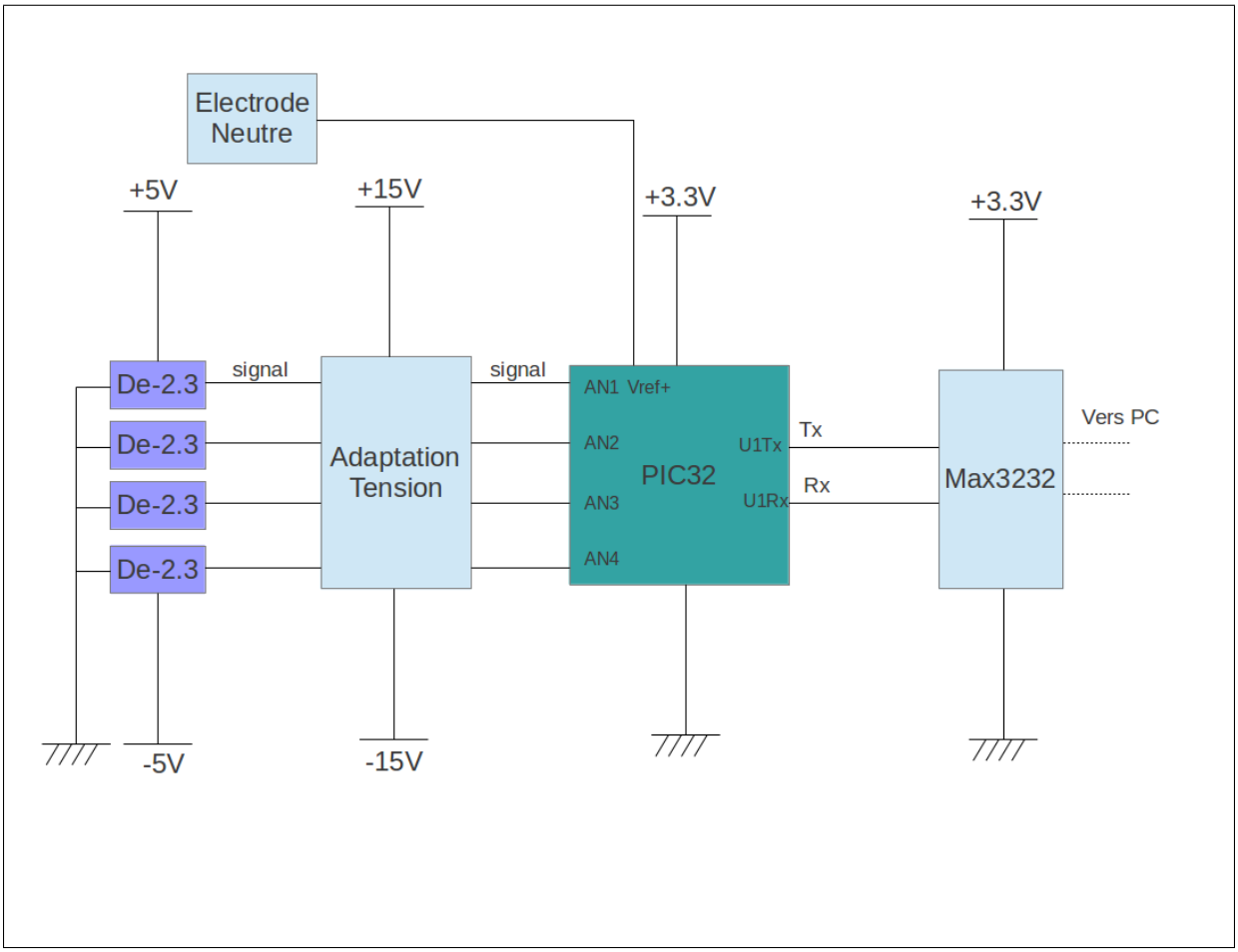

Figure 4.9 Schéma bloc présentant le câblage de la partie électronique du projet

L'électrode neutre est reliée au PIC par l'entrée VRef+ qui sert de référence de mesure pour le convertisseur analogique/numérique du microcontrôleur.

Une UART (Universal asynchronous receiver transmitter) du PIC32 est utilisée pour réaliser la liaison série à l'ordinateur. Les deux lignes de transmission et réception Tx et Rx sont reliées au Max3232 qui adapte les tensions entre le PIC et l'ordinateur par les entrées sorties U1Tx et U1Rx (UART 1 Tx et UART 1 Rx).

Si l'on prend maintenant un point de vue plus haut, nous sommes en mesure d'étudier le système dans son ensemble comme le présente la figure 4.10.

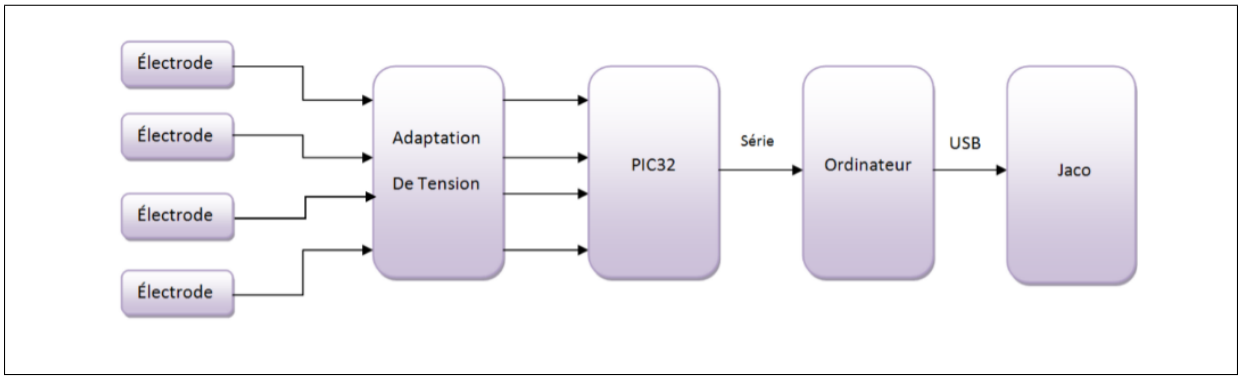

Figure 4.10 Schéma bloc présentant le système global

Il est nécessaire de préciser que dans cette architecture, l'ordinateur ne sert que d'intermédiaire entre le microcontrôleur et Jaco. En effet, le microcontrôleur effectue la totalité des calculs. Le mouvement effectué par l'utilisateur est envoyé sur la liaison série à l'ordinateur qui transmet la commande choisie à Jaco via la liaison USB.

Il est également à noter que cette architecture va induire un délai de traitement impossible à quantifier précisément. En effet, l'ordinateur n'étant pas temps réel au sens strict, le temps entre l'information d'un mouvement effectué et le nouvement de Jaco (ou toute autre action) commandé par USB est imprédictible précisément et sera beaucoup plus long que si le PIC32 était relié à Jaco directement par un bus CAN.

Ainsi, les temps de traitement mesurés caractériseront uniquement la partie calculatoire effectuée sur le PIC32, et non le temps entre le signal perçu par l'électrode et la commande appliquée sur le bras robotique.

4.3 Choix des Caractéristiques du Signal à utiliser et des Outils algorithmiques et Mathématiques

Une fois les choix matériel effectués et le calculateur choisi, l'étape de réalisation suivante est le choix des calculs a effectuer pour le bon fonctionnement de la détection d'activité musculaire.

Cette partie se décomposera en plusieurs sous partie à des fin de compréhension. La première partie concernera la manière de découper les fenêtre d'échantillons du signal EMG.

La seconde partie concernera le coeur du projet : la détection d'activité musculaire et les outils mathématiques utilisés pour la réaliser. Dans un premier temps, l'utilisation de l'énergie de Teager-Keiser sera détaillée puis l'utilisation d'un vote à la majorité pour le lissage de la détection sera expliquée et enfin la calibration de la détection.

Enfin une dernière partie détaillera le choix du filtre numérique appliqué sur le signal et la manière dont celui-ci est calibré pour chaque sujet et chaque canal EMG.

4.3.1 Découpage des fenêtres d'échantillons

Comme vu dans la revue de littérature, l'exploitation du signal EMG ne se fait pas échantillon par échantillon, le signal est découpé en fenêtre puis chaque fenêtre est utilisée pour calculer des caractéristiques. Ces caractéristiques sont utilisées dans un classificateur pour déterminer le mouvement effectué par l'utilisateur.

La taille de la fenêtre influe principalement sur les performance du classificateur, car une petite fenêtre induit un biais en raison de la variance du signal qui peut entraîner une classification fausse.

Le projet s'étant affranchi du classificateur, s'affranchit également du problème de fenêtrage du signal, dans le sens ou son choix ne joue que sur le temps de traitement de celui-ci, en effet, plus une fenêtre est longue plus le temps de calcul pour la traiter est long.

Toute fois, plus une fenêtre est courte, plus le délai avant de recevoir la prochaine fenêtre est également court.

Le choix du découpage en fenêtre du signal est donc ouvert et peut être fait en prenant tous les critères algorithmiques restant, à savoir le fonctionnement interne du PIC32, et des librairies qu'il met à disposition.

En effet, comme la partie 4.3.3 le présentera, le PIC32 met à disposition une librairie flottante incluant une fonction effectuant le calcul d'une transformée de Fourrier rapide (Fast Fourrier Transfor - FFT). Ces fonctions sont nécessaires au filtrage appliqué sur le signal. Les fonctions de cette librairie étant spécialement prévue pour l'architecture du PIC32, celles-ci ont une vitesse d'exécution optimale. La fonction FFT fournie par la librairie PIC32 est disponible sous 3 formes :

- fft8
- fft16
- fft32

Le nombre mentionné dans chaque fonction correspond au nombre d'échantillons dans le signal sur lequel la fft sera calculée. Ainsi fft8 prend en paramètre une fenêtre de signal de 8 échantillons et renvoie une FFT sur 8 points.

En conclusion, trois tailles de fenêtrage sont ici possibles de manière à faciliter le traitement du signal : 8, 16 et 32 échantillons. D'autres fenêtres seraient possible, mais il faudrait alors les découper en sous-fenêtres afin de pouvoir les filtrer.

4.3.2 Détection d'activité musculaire

La détection d'activité musculaire est le coeur du projet de contrôle de Jaco par EMG. En effet, le fait de s'être affranchi d'un classificateur implique que cette étape se déroule de manière optimale et toujours cohérente.

Une première partie présentera la caractéristique principale utilisée pour détecter les activité musculaire : l'énergie de Teager-Keiser (TKE).

Puis dans un second temps, le système de vote à la majorité sera détaillé. En effet, fixer un simple seuil sur le TKE du signal entraine une détection d'activité en changement d'état permanent, parfois à chaque nouvelle fenêtre. Le vote à la majorité permet de liser cette détection de manière à obtenir une détection d'activité continue sous la forme d'un échelon.

Enfin, le système de calibration pour chaque muscule et chaque sujet sera présenté.

4.3.2.1 Énergie de Teager-Keiser

L'étude de Li *et al.* (2007) a montré qu'une caractéristique du signal EMG est très efficace pour la détection d'activité musculaire automatique : l'énergie de Teager-Keiser (TKE). Cette caractéristique s'apparente à un taux de variation simple du signal en comparant un échantillon à son précédent et à son suivant. Le TKE se calcule comme présenté par l'équation 4.4.

$$TKE(k) = x^2(k) - x(k-1) * x(k+1) \tag{4.4}$$

Ainsi, une fois une fenêtre acquise complètement, le TKE instantané (sur chaque échantillon) est calculé. La figure 4.11 montre le TKE instantané obtenu sur un signal EMG donné.

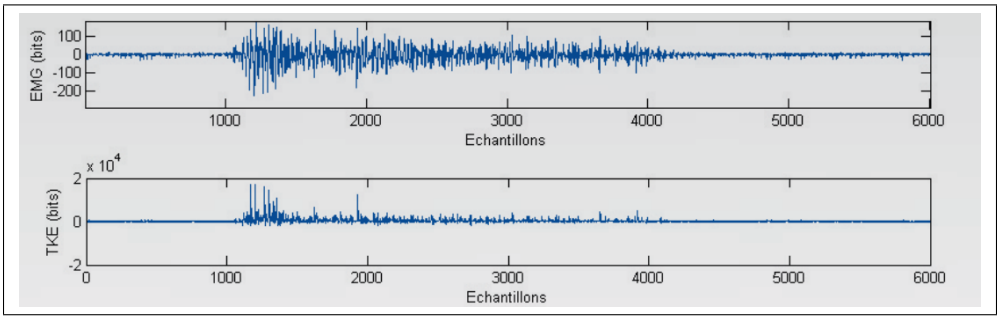

Figure 4.11 TKE instantané calculé sur un signal EMG

Cependant, ce TKE par échantillon est difficile à utiliser directement, car il est très variable et fixer un seuil efficace pour détecter les activités serait très compliqué. Pour faciliter son exploitation, on effectue la moyenne du TKE sur toute la fenêtre, afin d'en obtenir sont enveloppe. De cette manière, un seuil de détection est applicable plus facilement. La figure 4.12 présente le passage du TKE instantané au TKE moyen.

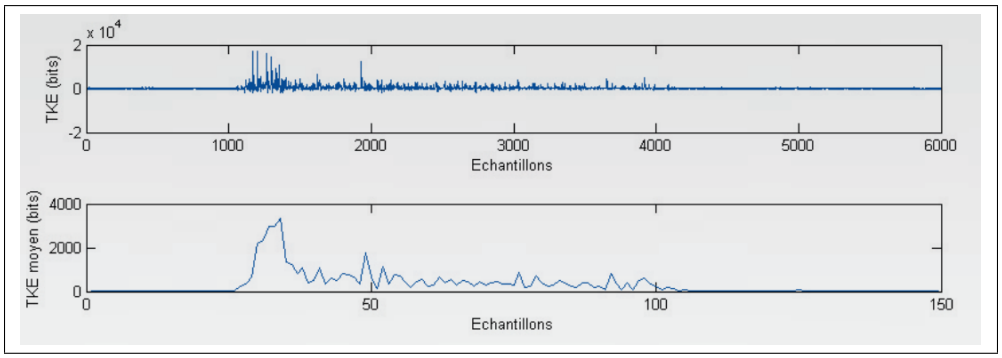

Figure 4.12 TKE moyen calculé sur un signal EMG

Une fois le TKE d'une fenêtre calculé, le but est de savoir si le muscle est activé ou non. Pour ce faire, un seuil doit être défini. Si le TKE de la fenêtre dépasse ce seuil, alors le muscle est considéré comme étant en activité.

4.3.2.2 Calibrage du seuil de détection

Une fois le système de détection d'activité musculaire défini, celui-ci doit ensuite être adapté à chaque utilisateur, et même à chaque muscle de cet utilisateur. En effet, chaque personne développe une force différente, de la même manière que chaque personne ne présente pas le même tonus musculaire lorsqu'il est au repos.

Par exemple une personne stressée sera beaucoup plus contractée au niveau des trapèzes qu'une personne complètement détendue, et ce même sans avoir l'impression d'être contracté.

De manière à calibrer efficacement le seuil de détection d'activité musculaire, il nous récolter deux information pour chaque muscle :

- le tonus musculaire au repos
- le tonus musculaire en contraction isométrique maximum

On acquiers ces données en se basant sur le TKE d'un muscule lorsqu'il est au repos, ainsi que lorsqu'il est en contraction isométrique maximum, c'est à dire que le muscle se contracte au maximum, tout en maintenant l'articulation fixe. Les équations 4.5 et 4.6 présentent les calculs permettant d'optenir les données.

$$Tonus_{repos} = \frac{\sum_{n=1}^{20} TKE(x_{repos}(n))}{20} \tag{4.5}$$

$$Tonus_{max} = \frac{\sum_{n=1}^{20} TKE_{max}(x_{max}(n))}{20} \tag{4.6}$$

où $x(n)$ représente une fenêtre de 16 échantillons du signal brut. On calcule donc la moyenne des TKE sur 20 fenêtre, ce qui correspond à une durée de $160ms$.

Ainsi, l'utilisateur reste au repos, ou bien rentre en contraction maximum, on actionne alors la calibration grâce à l'interface graphique sur l'ordinateur pour enregistrer les tonus de chaque muscle.

Une fois les tonus musculaires recueillis, on sait maintenant qu'à chaque mouvement, l'utilisateur produira un signal EMG dont le TKE moyen sera entre son tonus de repos et son tonus maximum.

Toutefois, il est peu probable que le tonus maximum soit atteint pendant un mouvement. En effet, une contraction isométrique produit un signal beaucoup plus gros qu'une contraction dans un mouvement lent. Le seuil de détection devra donc se trouver entre les deux tonus, le

plus proche possible du tonus de repos sans qu'un bruit occasionnellement plus élevé sur le canal ne puisse passer au dessus du seuil.

Le seuil a donc été fixé de manière heuristique à $1/4$ de l'intervalle entre le tonus de repos et le tonus maximum. La figure 4.13 met en image la manière dont le seuil est fixé pour chaque muscle.

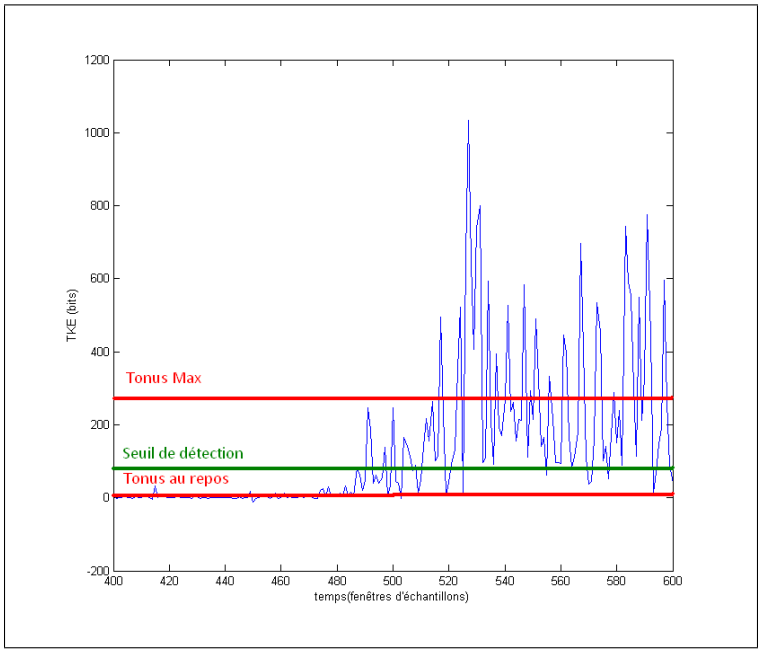

Figure 4.13 Définition du seuil de détection d'activité musculaire pour un canal EMG

4.3.2.3 Vote à la majorité

Le seuil fixé sur le TKE pour la détection d'activité permet donc de déterminer lorsqu'un muscle est en activité ou non, en observant lorsque ce seuil est dépassé. Cependant, le TKE variant au cours d'un mouvement, il se peut que celui-ci repasse par des moments proches de 0, même pendant un mouvement.

Ceci entraine alors une détection d'activité qui change alors souvent d'état comme le montre la figure 4.14.

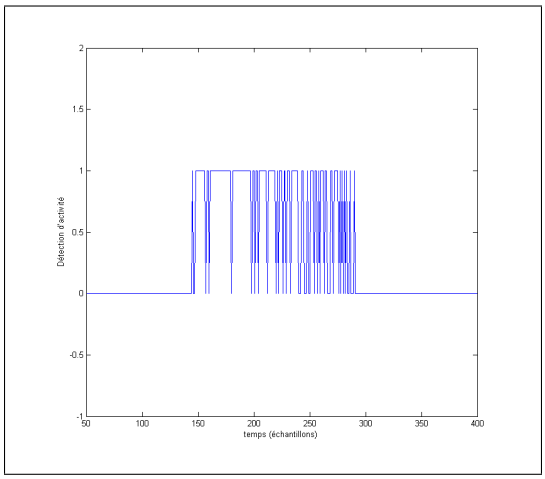

Figure 4.14 Détection d'activité musculaire brute, sans vote à la majorité.

Une conséquence supplémentaire est que si le seuil se veut d'être le plus bas possible afin d'augmenter la sensibilité du système, plus il est bas, plus le système est susceptible de détecter des faux positifs.

Le choix du seuil de détection est donc difficile, voir impossible à choisir idéalement. En effet, un seuil trop haut ne sera plus assez sensible pour détecter des mouvements faibles et en ratera une parte, tandis qu'un seuil trop bas sera sensible mais détectera des faux positifs si le bruit sur le TKE le dépasse.

Afin de remédier à ces deux problèmes, une méthode inspirée du vote à la majorité de Chang *et al.* (1996) a été mise en place. La méthode mise en place par Chang *et al.* (1996) consiste à comparer la caractéristique d'une fenêtre au seuil. Puis, pour déterminer la détection d'activité, on effectue un vote à la majorité sur les cinq dernières fenêtres. Si la majorité des fenêtre est au dessus du seuil, l'activité musculaire est mise à 1, sinon elle est mise à 0.

Ce système permet d'éliminer la plupart des faux positifs. Toute fois, un inconvénient persiste. En effet, une fenêtre d'activation suffit à faire la différence entre une activité ou un repos. Ainsi, la détection d'activité peut toujours osciller entre 0 et 1, si la comparaison au seuil est différente à chaque nouvelle fenêtre, résultant en une activité en dents de scie comme sur la figure 4.14.

La méthode mise au point pour le système consiste à mettre deux seuils de majorité et de minorité séparés. Ainsi, sur 32 fenêtres d'échantillons, si 16 ou plus sont au dessus du seuil de TKE, l'activité musculaire est mise à 1, puis, pour que celle-ci retombe à 0, il faut que le nombre de fenêtres au dessus du seuil retombe en dessous de 10. Cela permet de ne pas faire des allers-retours entre 15 et 16 fenêtres activées, qui entrainent des discontinuitées de détection.

Le résulstat de ce système de vote à la majorité est présenté par la figure 4.15.

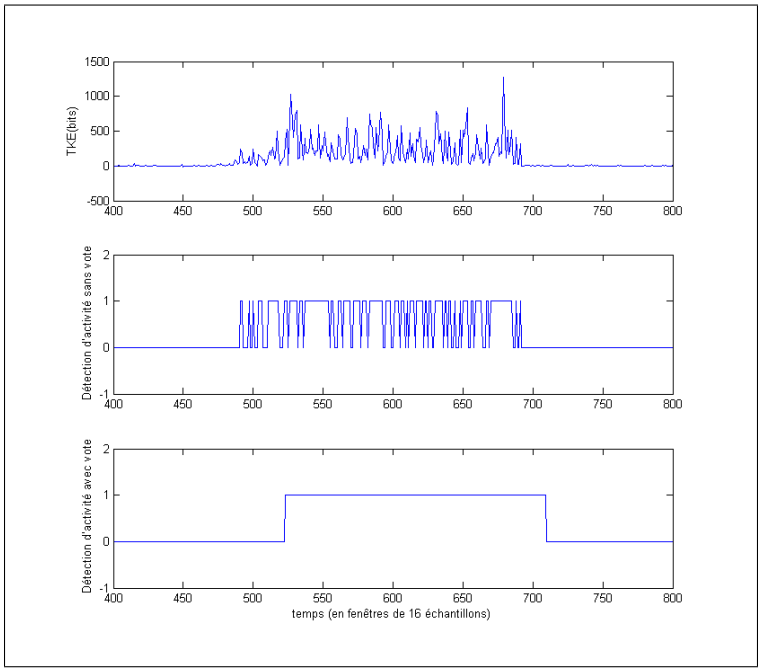

Figure 4.15 Détection d'activité musculaire avec vote à la majorité.

On observe qu'il n'y a plus de retour à 0 de la détection d'activité, et les faux positifs occasionnels ont également été retirés. Ce vote à la majorité provoque toute fois une légère latence de détection, en effet, il faut attendre que 16 fenêtres sur 32 soient au dessus du seuil de détection pour que l'activité soit détectée.

4.3.3 Filtrage

Dans un premier temps, le système a été réalisé sans filtrage numérique. Ainsi le système fonctionnait uniquement en se basant sur la détection d'activité musculaire décrite au chapitre 4.3.2.

Les tests ont été concluants concernant la détection d'activité seule. Cependant le système a présenté une faiblesse : pour être détecté, un mouvement doit être très ample et comporter une composante isométrique. Les mouvements simples et lents ne demandant pas beaucoup de force ne sont pas détectés par le système.

En résumé, le système manque de précision, et son utilisation chez des personnes blessées médullaires ayant potentiellement moins de tonus musculaire qu'une personne en bonne santé est alors compromise.

Une solution est de filtrer le signal EMG brut de manière à amplifier les phases d'activité musculaire et diminuer au maximum le bruit de mesure.

L'utilisation d'un filtre prédictif est donc préconisée pour effectuer ce travail.

Cette partie présente le filtre prédictif mis en place dans le système dans le but d'augmenter sa sensibilité tout en diminuant le bruit de mesure. Une brève description du filtre sera d'abord effectuée, puis le calibrage de celui-ci sera décrit dans une seconde partie, pour enfin terminer par la description détaillée du filtre et de son algorithme.

4.3.3.1 Concept

Le but du filtrage prédictif est de pouvoir prédire quelles partie du signal le filtre doit amplifier, et quelles parties de ce même signal il doit écraser.

Dans notre cas, la partie du signal à écraser est le bruit analogique mesuré par les ADC du PIC32, tandis que la partie du signal à amplifier est le signal EMG généré par le muscle sur lequel est collé l'électrode.

Le filtre ayant deux élément à reconnaitre (le bruit et le signal), il faut en caractériser un des deux, et définir que lorsque l'on ne le détecte pas, c'est que l'on est en présence de l'autre. Ainsi, on choisit de caractériser le bruit. Lorsque le système détecte la présence du bruit, alors le signal est écrasé, lorsqu'il ne le détecte pas, c'est qu'un EMG est présent dans le signal, il est alors amplifié.

Pour caractériser le bruit, on utilise les caractéristiques fréquentielles du signal. Le but est donc d'isoler le spectre d'amplitude du bruit afin d'en faire un modèle de comparaison. On enregistre un signal sans EMG et on calcule le spectre d'amplitude moyen de ce signal. C'est la bande de base.

Puis à chaque nouvelle fenêtre, on compare le spectre d'amplitude de cette fenêtre à celui de la bande de base pour calculer une probabilité de présence du bruit. Cette probabilité permet de fixer un gain qui amplifiera ou écrasera le spectre d'amplitude et donc par extension, le signal.

Il est également à noter que l'on ne veut pas créer de déphasage du signal. La phase du signal sera donc isolée de la fft pour être réutilisée sans modification pour reconstruire le signal après que le spectre d'amplitude a été transformé.

4.3.3.2 Calibrage du Filtre

Une fois le concept du filtre posé, le mode de fonctionnement précis du filtre va maintenant être détaillé. La première étape de la réalisation du filtre est son calibrage, ou l'enregistrement de la bande de base.

Le calibrage du filtre s'effectue sur une partie de signal où l'on est sur qu'aucun EMG ne sera présent. On demande donc à l'utilisateur de ne pas bouger pendant l'acquisition de la bande de base.

Le signal est alors acquis par fenêtre. Lorsqu'une fenêtre est acquise, on la normalise, c'est à dire qu'on retire sa moyenne à chaque échantillon (eq :normalCalib où $NFFT$ = 16), puis on calcule sa FFT (équation 4.8).

$$x_{k=1}^{k=NFFT}(k) = x(k) - \frac{\sum_{n=1}^{NFFT} x(n)}{NFFT} \tag{4.7}$$

$$fft = FFT_{16}(x(k)) \tag{4.8}$$

Le fft calculé par cette opération est donc un tableau de 16 nombres complexes représentant la transformée de fourrier de la fenêtre.

Une fois cette FFT calculée, on met la phase du signal de côté pour ne garder que le spectre d'amplitude M de la fenêtre (équation 4.9).

$$M_{k=1}^{k=NFFT}(k) = \|fft(k)\|^2 \tag{4.9}$$

On garde ensuite en mémoire ce spectre d'amplitude, puis on répète l'opération sur 20 fenêtres. Une fois les 20 spectres d'amplitude acquis, on calcule le spectre d'amplitude moyen de ces 20 fenêtres (équation 4.10).

$$BandeBase = \frac{\sum_{n=1}^{20} amplitude(n)}{20} \tag{4.10}$$

Ce spectre d'amplitude moyen constitue la bande de base qui sera comparée à chaque fenêtre du signal pour déterminer à chaque fenêtre la probabilité de présence du bruit dans le signal.

4.3.3.3 Description du Filtre

Une fois la bande de base du filtre calibrée, le filtre peut être mis en fonctionnement à chaque fenêtre de signal acquise. Cette partie détaille le fonctionnement algorithmique du filtre.

Une nouvelle fenêtre est acquise. Dans un premier temps, elle est sujette au même traitement que pour la calibration : elle est normalisée (équation 4.7) puis sa FFT est calculée (équation 4.8).

Une fois la FFT obtenue, on sépare le spectre d'amplitude (équation 4.9) de la phase ϕ qui cette fois-ci est aussi calculée (équation 4.11). En effet, celle-ci sera nécessaire pour reconstruire le signal une fois que l'amplitude aura été modifiée.

$$\phi(k) = atan2(\Im(fft(k)), \Re(fft(k))) \tag{4.11}$$

On calcule ensuite la probabilité a posteriori $postsnr$ de présence du filtre en faisant le rapport entre le spectre d'amplitude de la fenêtre et le spectre d'amplitude moyen (équation 4.12).

$$postsnr(k) = \frac{M(k)}{M_{Bandedebase}} - 1 \tag{4.12}$$

Cette probabilité est ensuite limitée pour ne pas créer de distorsion provoquée par une valeur trop petite (équation 4.13).

$$postsnr(k) = \max(postsnr(k), 0.01) \tag{4.13}$$

On compare ensuite la probabilité obtenue avec la probabilité obtenue avec le spectre d'amplitude obtenue à la fenêtre précédente. Cette méthode est appelée méthode de la "décision adaptative" (équation 4.14). Une nouvelle amplitude est alors calculée en appliquant cette probabilité sur l'amplitude d'origine (équation 4.15).

$$eta(k) = \alpha * \frac{M(k-1)}{M_{Bandedebase}} + (1-\alpha) * postsnr(k) \tag{4.14}$$

$$M_{new}(k) = \frac{eta}{eta+1} * M(k) \tag{4.15}$$

On calcule ensuite le gain à appliquer sur le spectre d'amplitude en comparant l'amplitude sur laquelle on a appliqué les probabilités a priori avec le spectre d'amplitude moyen (équation 4.16, 4.17, 4.18 et 4.19).

$$tsnr(k) = \frac{M_{new}(k)}{M_{Bandedebase}} \tag{4.16}$$

$$Gtsnr(k) = \frac{tsnr(k)}{tsnr(k)+1} \tag{4.17}$$

$$Gtsnr(k) = \max(Gtsnr(k), 0.01) \tag{4.18}$$

$$M_{new}(k) = Gtsnr * M(k) \tag{4.19}$$

Une fois la nouvelle amplitude calculée, il ne reste plus qu'à reconstruire la fft du signal, en réunissant l'amplitude et la phase (équation4.20 et 4.21).

$$\Re(fft(k)) = M_{new}(k) * cos(\phi(k)) \tag{4.20}$$

$$\Im(fft(k)) = -M_{new}(k) * sin(\phi(k)) \tag{4.21}$$

Une fois fait, on effectue la transformée de fourrier inverse pour retrouver le signal EMG filtré.

4.4 Architecture Logicielle

Les parties précédentes ont présenté l'architecture matérielle ainsi que les outils mathématiques utilisés dans le projet. La derniere grande partie technique à présenter restante est donc l'architecture logicielle.

En effet, tout en prenant en compte les choix effectués dans la partie précédente, la manière de les réaliser et de les connecter ensembles doit être présentée.

Ainsi, ce chapitre présentera d'abord le fonctionnement du firmware, c'est à dire le programme exécuté sur le PIC32 puis présentera le fonctionnement de l'application de monitorage des signaux, qui permet aussi la connexion du système au bras Jaco et qui est exécutée sur un ordinateur.

4.4.1 Firmware

Le firmware est le nom donné à un programme tournant sur un microcontrôleur, ou sur une carte électronique. Ce firmware est un programme développé dans un langage dépendant du microcontrôleur choisi pour le projet.

Le PIC32 est programmable en langage C, ainsi qu'en assembleur bien qu'il soit peu utilisé, excepté pour les parties de code nécessitant une grande optimisation.

Cette partie décrira le fonctionnement du PIC32 ainsi que l'algorithme global du firmware.

4.4.1.1 Fonctionnement et Configuration Globale du PIC32

Pour présenter le firmware, il est important de présenter la manière de fonctionner du PIC32. La différence principale entre un microcontrôleur et un ordinateur moderne est qu'il est impossible d'exécuter plusieurs programmes en parallèle sur un microcontrôleur. En effet, c'est le système d'exploitation qui permet dans un ordinateur de paralléliser les opérations effectuées.

Ainsi, un microcontôleur n'effectue qu'une seule série d'opérations : celle de son programme. Un programme est une suite d'instructions élémentaires compilées dans un fichier binaire. Cette suite d'instruction est inscrite dans la mémoire de programme du microcontrôleur et chaque instruction possède une adresse dans cette mémoire. Le compteur de programme (PC - program counter) et un nombre contenant l'adresse mémoire de l'instruction en cours. Une fois que cette instruction est finie, le PC est incrémenté pour passer à l'instruction suivante et jusqu'à ce que le programme se termine.

Cependant le microcontrôleur possède des périphériques dédiés capables d'effectuer les opérations pour lesquelles ils sont prévus sans influencer le déroulement du programme principal. Par exemple, le convertisseur analogique-numérique (ADC) enregistre les échantillons du signal sans que cela ait à figurer dans le programme.

Ces périphériques possèdent des interruptions, qui permettent de signifier au programme lorsque des données sont accessibles. Ces interruptions sont déclenchées quand une condition est rem-

plie et sont une fonction spécifique qui n'est exécutée que lorsque la condition est remplie. Par exemple, il est possible de configurer l'ADC du PIC32 pour qu'il déclenche une interruption à chaque fois qu'il a fini d'acquérir 16 samples.

Lorsqu'une interruption est déclenchée par un périphérique, le déroulement du programme est arrêté. L'adresse de l'instruction où il s'est arrêté est mise en mémoire, puis le code de la fonction d'interruption est exécuté. Une fois que cette fonction est terminée, le programme reprend son cours normal à l'adresse qui avait préalablement été mise en mémoire.

Ce système d'interruption permet d'effectuer automatiquement des opérations comme l'acquisition d'un signal, ou l'envoi et la réception de messages sur le port série sans influencer le programme principal, et donc gagner du temps de calcul.

Cependant, les interruptions bloquent le cours normal du programme principal. Les interruptions doivent donc être le plus courtes possibles pour ne pas perdre trop de temps dedans.

Il est donc important de bien choisir quels calculs sont effectués dans les interruptions, et quels calculs sont faits dans le programme principal.

Une interruptions peut également être interrompue par une autre interruption. Un système de priorité détermine l'ordre dans lequel ces interruptions ont lieu. Une interruption ne pourra jamais interrompre une autre interruption ayant une priorité plus haute. Ainsi l'interruption encours se terminera avant que l'autre puisse s'effectuer.

Ainsi, les deux périphériques utilisés dans le système fonctionneront à base d'interruptions, à savoir l'ADC permettant l'acquisition des quatre canaux, et l'UARTT permettant la communication série.

L'ADC du PIC32 permet de scanner plusieurs entrées les unes après les autres. Ainsi il faut donc multiplier la fréquence d'échantillonnage par le nombre de canaux afin de pouvoir acquérir tous les canaux à la bonne fréquence. On fixe donc la fréquence d'échantillonnage à 8KHz pour que chaque canal soit échantillonné à 2KHz.

L'interruption de l'ADC est configurée pour être déclenchée après avoir acquis le plus d'échantillons possible, soit 8 échantillons. Ainsi, tous les 8 échantillons, correspondant à 2 échantillons par canal, l'interruption de l'ADC est déclenchée.

L'UARTT du PIC32 quant à elle possède deux interruptions, une pour l'envoi de données, et l'autre pour la réception. Ces interruptions sont déclenchées respectivement à l'envoi et à la réception d'un octet complet. Ainsi, à chaque octet reçu, celui-ci est mis en mémoire dans l'interruption, et à chaque octet envoyé, on met l'octet suivant dans le "buffer" d'envoi.

4.4.1.2 Alogorithme global

Une fois le fonctionnement global du PIC32 et de ses interruption décrit, l'algorithme du programme principal ainsi que ceux des interruptions peuvent maintenant être présentés en détail.

Nous nous intéressons dans un premier temps au fonctionnement du programme principal, puis les algorithmes des deux interruptions principales seront détaillés.

Le programme principal du firmware commence par initialiser tous les périphériques du PIC32 qui seront utilisés, ainsi que l'horloge principale et les configurations des interruptions. C'est à ce moment que l'ADC et le port série sont configurés.

Une fois les initialisations effectuée, le programme entre dans une boucle infinie qui constitue la colonne vertébrale du programme. Les actions réalisée par cette boucle sont présentée par la figure 4.16.

La boucle commence ainsi par vérifier si une commande à été reçue sur le port série. Les commandes reçue sur le port série sont les commandes de calibration du système. Ainsi chaque commande est constituée du type de calibration a effectuer, c'est à dire une calibration du tonus maximum, une calibration du tonus au repos ou bien la calibration de la bande de base du filtre, et le numéro du canal sur lequel effectuer cette calibration.

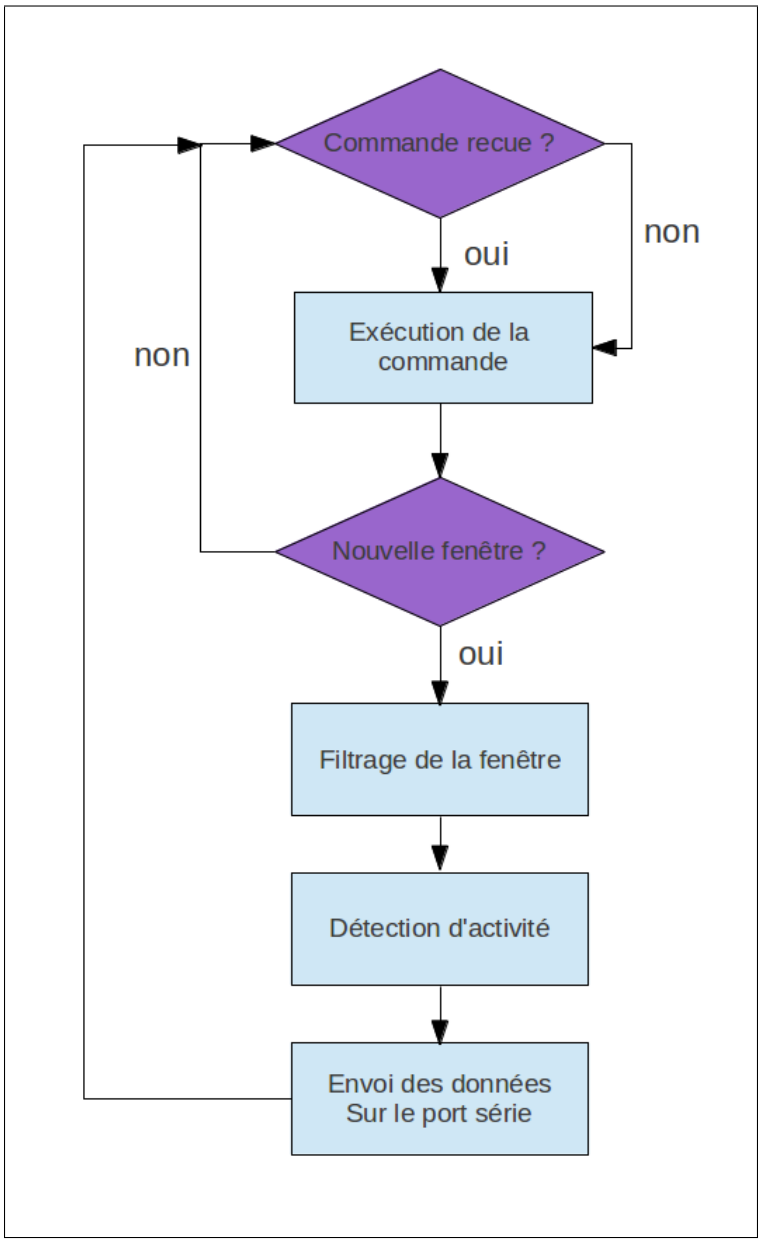

Figure 4.16 Boucle inifinie du programme principal.

On vérifie ensuite si une fenêtre de signal a été acquise en entier. Si c'est le cas, cette fenêtre est filtrée, puis passée dans la fonction de détection d'activités. Enfin les données calculées sont envoyée sur le port série pour être monitorées.

Il est à noter que le filtrage n'est effectué que si la calibration de la bande de base a bien eu lieu précédemment, mais ce test n'est pas affiché sur le diagramme à des fins de clarté.

Cette partie montre comment le signal est traité, présentons maintenant comment celui-ci est acquis. La figure 4.17 présente la routine d'interruption de l'ADC du PIC32. Celle-ci est déclenchée à chaque fois que 8 échantillons ont été acquis en tout, soit 2 échantillons par canal.

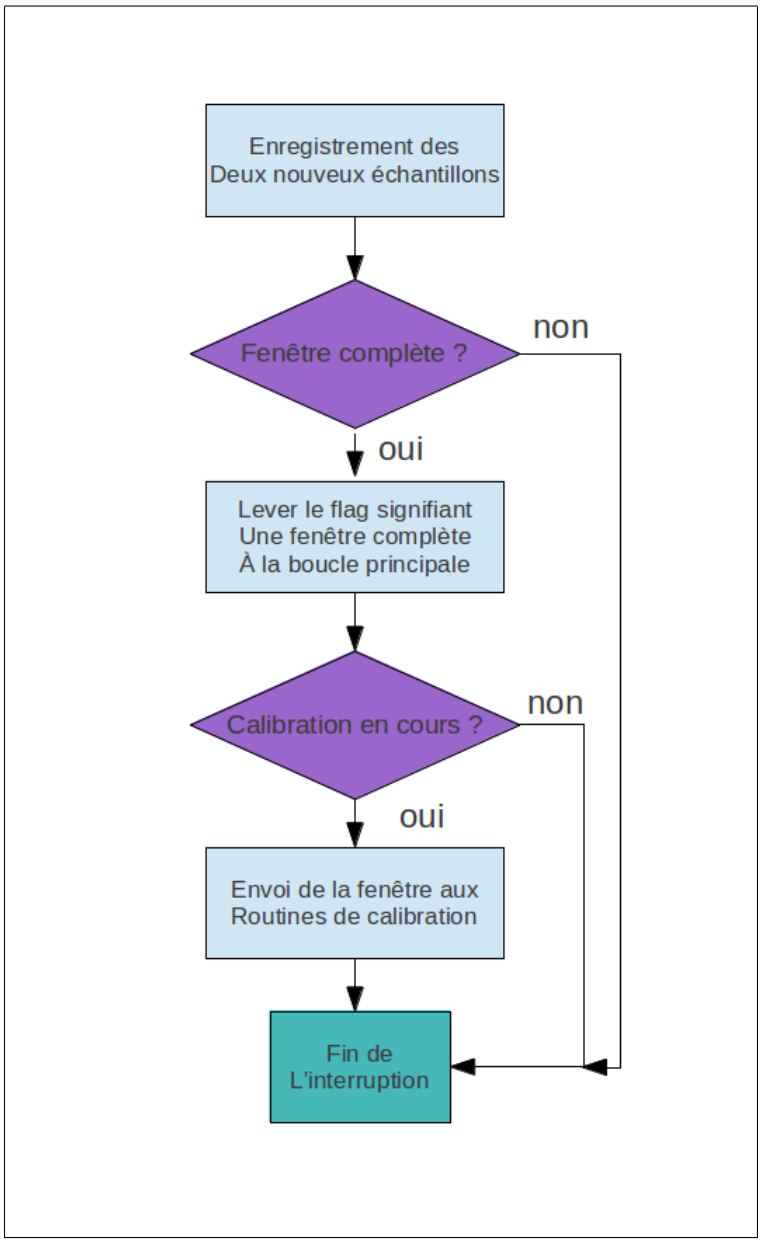

Figure 4.17 Routine d'interruption du firmware.

Il est à noter que deux tableaux de données sont utilisés pour chaque canal. L'un des tableau est rempli au fur et à mesure par l'ADC pendant que l'autre est traité, c'est à dire filtré puis

passé à la détection d'activité. Une fois qu'un tableau est rempli par l'ADC, celui-ci est passé en phase de traitement, et l'autre est rempli à nouveau par l'ADC. Il est donc nécessaire que le tableau à traiter le soit avant qu'une nouvelle fenêtre ne soit acquise, sinon des données seront perdues.

4.4.2 Application de monitorage des signaux

La partie firmware ayant été présentée, le dernier élément technique du projet est la réalisation du programme de monitorage des signaux. Cette application est un programme tournant sur un ordinateur. L'ordinateur, grâce aux données reçues à travers la liaison série, est en mesure d'afficher les données acquises ou calculées par le PIC32.

Ceci a été utile lors de la réalisation dans un premier temps à des fins de débogage, puis après pour l'utilisation même du système. En effet, l'interface graphique permet de lancer les phases de calibration du système, pour calibrer les seuils de détection d'activité musculaire, et pour calibrer la bande de base du filtre.

Ce programme permet également de faire le lien entre le PIC32 et le bras robotique JACO. Cette partie a nécessité l'ajout d'une surcouche logicielle supplémentaire pour cause d'incompatibilité de langages comme le présentera la partie 4.4.2.2.

Enfin la conception complète, à savoir le diagramme UML du programme ainsi que son algorithme principal seront détaillés, ainsi que toute les fonctionnalités finales du programme.

4.4.2.1 Choix des Librairies et des langages

Le choix des langages de programmation ont été effectués au début du projet. En lisant la documentation fournie par Kinova, il s'est avéré possible d'interfacer Jaco avec un programme C++. De plus, l'état de l'art de la robotique moderne utilise largement le langage C++, qui en est devenu un des standards.

En effet, le framework open source ROS (Robotic Operating System) est intégralement développé en C++ et Python, deux langages facilement interfaçables.

La librairie Qt est une librairie graphique open source qui permet la création facile et multi plateforme d'interface graphique robuste. Cette librairie est l'une des plus complète existant aujourd'hui et possède des modules supplémentaires multi plateforme en grand nombre, comme un module réseau, des bases de données, la gestion des fichiers ou encore l'utilisations d'architectures modèle-vue déjà implémentés. Ceci permet d'ouvrir beaucoup d'option pour la suite du développement du projet, dans le cas où de nouveaux éléments viendraient à y être ajoutés.

La combinaisons de ces arguments a donc logiquement tourné le choix du langage informatique principal du programme vers le langage C++.

Cependant, après plusieurs mois de développement, il s'est avéré que l'utilisation des librairies de Kinova doive se faire en passant par une couche logicielle supplémentaire : Mono. Cette partie est détaillée dans la partie suivante.

En effet, les librairies de Kinova ont été nativement réalisée en langage C#. Le C# est un langage développé par Microsoft et qui est de ce fait, conçu pour fonctionner de manière optimale sous Windows. Afin d'augmenter les possibilités de développement en C#, une librairie a été conçue par la communauté open source afin de rendre le C# utilisable sur d'autres systèmes d'exploitation comme linux ou MacOS.

4.4.2.2 Lien avec Jaco et Mono

Le but du projet étant de contrôler le bras JACO, il est nécessaire d'utiliser l'interface de programmation (API) de Kinova pour le contrôler. Cette librairie permet la connexion du bras JACO au programme via USB, et implémente toutes les méthodes permettant d'envoyer des ordres précis à JACO. Ces ordres peuvent être des ordres de configuration du bras, oubien des commandes pour lire des valeurs d'état du bras (courant dans les articulation, température des moteurs, angles des articulations, ...), des commandes de déplacements des articulations, ou mêmes des trajectoires de mouvement.

L'API Kinova a été réalisée intégralement en C#, et est donc inutilisable directement par un programme codé en C++ pur. Dans sa documentation, Kinova préconise l'utilisation d'un framework appelé Mono pour l'utiliser depuis un programme C++.

Mono est une librairie permettant de mettre a disposition des structures de données (classes, fonctions...) du monde C++ accessibles au monde .NET (C#) et inversement. De ce fait, un programme C++ peut déclarer des objets définis en C# et utiliser ses méthodes.

Le monde .NET permet de créer, à la compilation un type de fichiers appelés des Assembly. Ce sont des fichiers de type .exe ou .dll directement générés par du C#. Dans le cas du système, le but est d'avoir accès au code contenu dans les fichiers dll de Kinova.

L'API kinova est constituée de sept DLL regroupant diverses classes et fonctions utiles à l'utilisation de JACO. Plutôt que d'importer les 7 dll de kinova dans mono, et de devoir aller chercher chaque classe ou fonction dans le bon Assembly, il a été conseillé par un ingénieur Kinova d'encapsuler toutes les classes et fonctions requises et de générer une nouvelle DLL unique regroupant toutes ces fonctions. Cela permet de n'appeler qu'un seul fichier .dll ne contenant uniquement que les fonctions et classes dont on a besoin pour la réalisation du système. Cela permet un code beaucoup plus propre et beaucoup moins lourd à réaliser tant qu'à assimiler.

On crée ensuite une classe JACO en C++. Cette classe propose toutes les méthodes de la classe C# de l'API Kinova. Toute fois, les méthodes de cette classe contiennent les appels aux fonctions C# grâce à Mono. De cette manière, JACO est interfacé comme une classe normale, et aucun code concernant Mono ne sort de cette classe.

Tous ces étages réalisés permettent de faire abstraction de toute la partie gérée par Mono et d'utiliser l'API de Kinova comme si celle-ci était accessible directement en C++. Le schéma 4.18 résume l'utilisation des librairie Kinova dans le système.

En synthétisant :

- On écrit un programme C# regroupant toutes les fonctions de la librairie Kinova dont ont a besoin. Ce programme fait appel directement aux fichiers .dll fournis par Kinova.

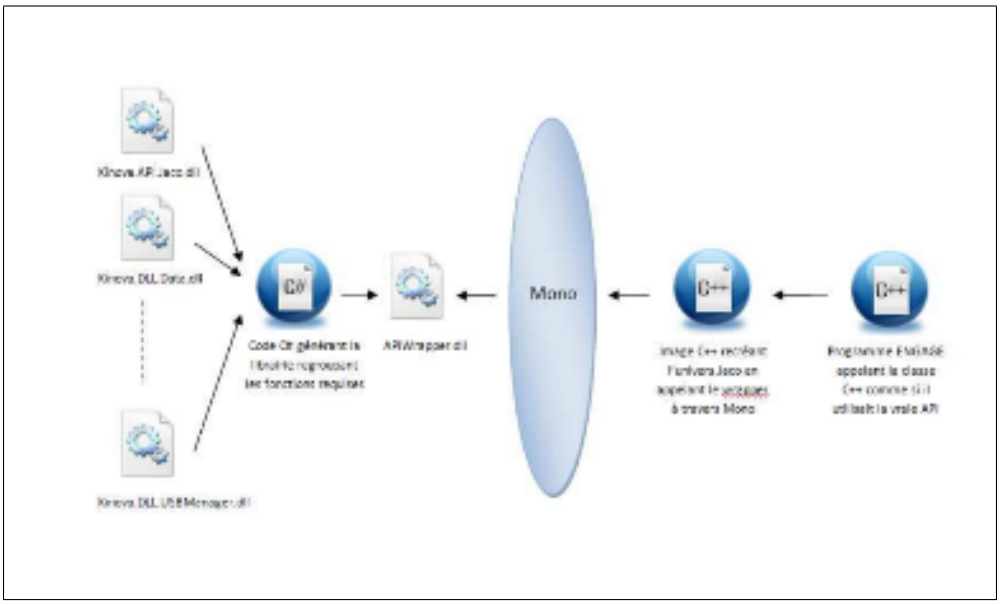

Figure 4.18 Interfaçage de l'API Kinova à travers Mono.

- On construit un nouvel assembly : APIWrapper.dll qui sera appelé dans le programme de monitorage
- Grâce à Mono, on charge les classes et méthode de APIWrapper dans le programme C++ grâce à une classe Jaco
- Le programme se connecte à Jaco en appelant les méthodes de la classe Jaco

4.4.2.3 Fonctionnement Global et Schéma UML du Projet

Le logiciel de monitorage des signaux est un programme basé autour de l'interface graphique réalisée grace à la librairie Qt. Le diagramme UML du programme est présenté par la figure 4.19.

MainWindow est la classe principale de l'interface graphique, c'est elle qui contient tous les éléments graphiques de l'interface. Elle dialogue avec les autres classe afin d'en afficher les données entrantes et sortantes.

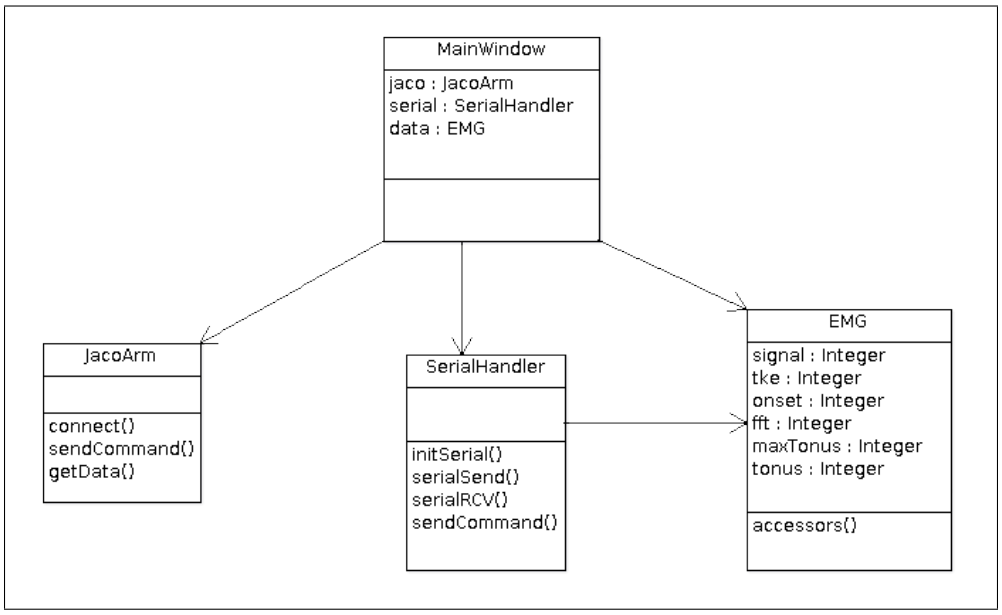

Figure 4.19 Diagramme UML du programme de monitoring.

La classe SerialHandler est l'objet qui gère tout ce qui touche à la communication série. Il possède une méthode de réception des données qui tourne dans un processus séparé du reste du programme.

La classe EMG est la classe contenant toutes les données EMG reçues du PIC32. Lorsqu'une donnée est reçue, elle est réceptionnée par l'objet SerialHandler qui met à jour les données dans la classe EMG. MainWindow va ensuite chercher les données dans la classe EMG afin de les afficher dans l'interface.

La classe JacoArm est la classe décrite dans la partie précédente qui fait le lien avec les méthodes des librairies de Kinova à travers Mono. C'est dans cette classe que Mono est utilisé.

L'interface graphique réalisée possède plusieurs onglet permettant chacun de visualiser un signal pour les quatre canaux. La figure 4.20 montre la fenêtre de visualisation des signaux EMG pour les quatre canaux. Un autre onglet permet de visualiser le TKE, un autre la détection d'activité musculaire et un dernier permet de monitorer une fft.

Deux onglet supplémentaires viennent s'ajouter aux quatre onglets de visualisation des signaux. Le premier, présenté par la figure **??** permet le contrôle de Jaco. En effet, Jaco étant normalement contrôlé par son joystick, il faut donc activer ou désactiver son contrôle par l'API. Cette fenêtre permet également de visualiser le statut de cette connexion et de visualiser les information renvoyées par Jaco.

Enfin le dernier onglet présenté sur la figure **??** permet d'envoyer les commandes de calibration au PIC32. Ces calibrations sont la calibration du tonus au repos, du tonus maximum et de la bande de base pour le filtre, et ce pour chaque canal EMG.

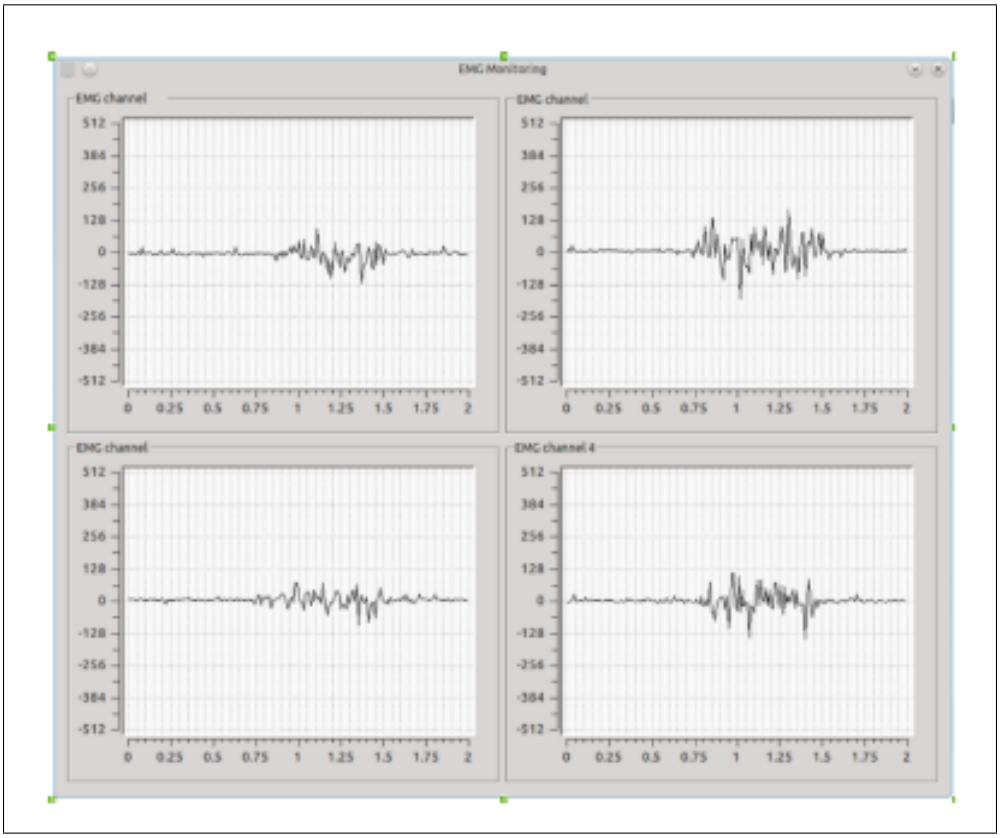

Figure 4.20 Capture d'écran de l'interface graphique du programme de monitorage des données EMG.

4.5 Validation du Système

Le système ayant maintenant été décrit dans son ensemble, il est maintenant important de présenté la manière utilisée pour valider ses performance sur un utilisateur. La partie suivante présente le protocole de validation de performances du système tel qu'il a été présenté en commité d'éthique en mai 2013, et présente les expérimentation prévues.

4.5.1 Protocole de validation

4.5.1.1 Mode de recrutement

Les participants seront recrutés par affichage au sein du laboratoire de recherche de recherche en imagerie et orthopédie ainsi que dans le département de génie de la production automatisée. Ils devront répondre aux critères suivants :

- Critères d'inclusion
  - Les sujets seront des adultes volontaires de plus de 18 ans.
- Critères d'exclusion, le sujet de devra pas être sujet à :
  - des douleurs cervicales
  - des torticolis fréquents
  - tout problème orthopédique de la ceinture scapulaire et cervicale
  - des maux de tête réguliers

4.5.1.2 Principe de fonctionnement

La détection de l'activité musculaire se fera par un seuil appliqué sur l'énergie du signal (TKE) du signal EMG échantillonné à 2KHz. Le but de ce protocole est de déterminer la sensibilité du système en le testant sur 10 sujets. L'activité musculaire générée lors d'un mouvement dépend de l'amplitude du mouvement effectué, ainsi que de la vitesse à laquelle le mouvement est effectué. Ainsi la validation s'effectuera en faisant effectuer au sujet plusieurs mouvements d'amplitudes différentes à des vitesses différentes.

4.5.1.3 Protocole expérimental

Un capteur inertiel (Interlink, Microstrain) sera utilisé afin d'enregistrer l'angle de rotation de la tête ainsi que la vitesse angulaire du mouvement. Celui-ci sera disposé sur un casque de type casque d'écoute audio avec les oreilles comme point de repère de placement.

Un pointeur laser sera également disposé sur la tête du sujet afin que celui-ci puisse indiquer au sujet l'orientation précise de sa tête.

L'angle visé sera indiqué au sujet par des marqueurs disposés devant lui et vers lesquels il devra tourner la tête en se repérant à l'aide du pointeur laser. La vitesse visée sera indiquée au sujet par le biais d'un métronome électronique. Le sujet devra réaliser le mouvement vers le marqueur indiqué en un temps de métronome.

La figure 4.21 décrit l'angle de rotation de la tête à effectuer par le sujet.

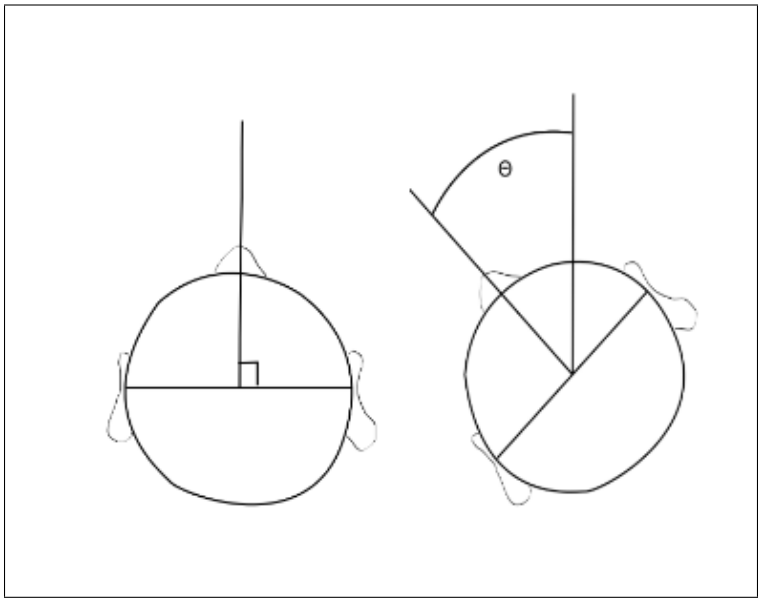

Figure 4.21 Description de l'angle de rotation de la tête.

4.5.1.4 Déroulement des acquisitions

Le sujet sera assis devant une surface comportant les cibles qu'il devra viser avec le pointeur laser. Le pointeur laser ainsi que le capteur angulaire seront disposés sur un casque d'écoute audio qui sera placé sur les oreilles du sujet. Une étape de familiarisation au métronome et d'entrainement à la visée permettra au sujet de s'entrainer à effectuer le mouvement aux angles prévus à la bonne vitesse.

Voici une liste des mesures qui seront prises pendant l'expérimentation :

- Angle de rotation de la tête
- Vitesse angulaire de rotation de la tête
- Signaux électromyographique du muscle sterno-cléïdo mastoïdien
- Signal de détection d'activité musculaire

Et ce pour chaque combinaison d'angle et de vitesse de métronome ciblés.

Le sujet devra réaliser des mouvements de rotation de la tête à des angles de 10 degrés, 20 degrés, 30 degrés, 40 degrés, 50 degrés, 60 degrés, 70 degrés et 80 degrés. Chaque déplacement angulaire sera effectué à des vitesses de métronomes de $60bpm, 90bpm, 120bpm, 160bpm$ et $200bpm$ (battement par minute) .

L'ordre des vitesses et des angles sera tiré au hasard avant l'expérimentation pour chaque sujet.

4.5.1.5 Notes importantes

Après chaque mouvement un fichier devra être sauvé sous la forme aXXvYYY.csv où XX correspond a la valeur de l'angle visé lors de l'enregistrement et YYY correspond à la valeur du métronome utilisé pour l'enregistrement.

A chaque acquisition, un monitoring sera effectué sur le programme Monitor afin de vérifier si le système a détecté une activité musculaire ou non. Le fichier de sauvegarde devra être

sauvegardé dans le même dossier que les données du capteur angulaire sous la forme emgaXX-vYYY.txt

4.5.1.6 Exemple de données récoltées

La figure **??** montre un exemple de la forme que prendront les données récoltées lors d'une acquisition sur une personne.

Ces données représentent les valeurs brutes enregistrées grâce aux capteurs inertiel et EMG.

4.5.1.7 Interprétation des données

Une fois les données acquises, 4 paramètres seront estimés pour évaluer la sensibilité du système correspondant au sujet testé : l'efficacité, le coefficient de l'angle maximum, le coefficient de la vitesse angulaire maximale ainsi que l'efficacité globale.

Le paramètre de l'efficacité est définit comme le rapport entre le nombre de détections sur le nombre d'essais total comme montré par l'équation 4.22.

$$Efficacite = \frac{Nb\ de\ dtections}{Nb\ d'essais\ total} \tag{4.22}$$

On est en mesure de donner une valeur entre 0 et 1 à l'efficacité du système pour une personne donnée. La valeur 1 indique que le système a détecté toutes les activités réalisées par la personne et la valeur 0 indique au contraire que le système n'en a détecté aucune.

Cependant ce paramètre ne prend pas en compte les angles et vitesses angulaires atteintes lors des expérimentations. On détermine donc un coefficient d'angle ainsi qu'un coefficient de vitesse angulaire, qui viendront pondérer le coefficient d'efficacité de chaque sujet. Ces coefficients seront basés sur l'angle maximum atteint par un sujet et la vitesse angulaire atteinte par un sujet. Ainsi les résultats seront pondérés relativement au sujet ayant atteint l'angle et la vitesse angulaire les plus grands.

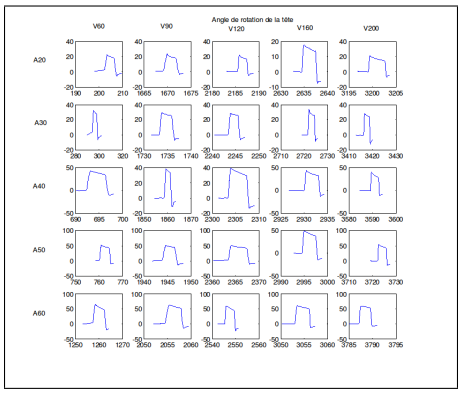

Figure 4.22 Exemple de données d'angles de rotation de la tête

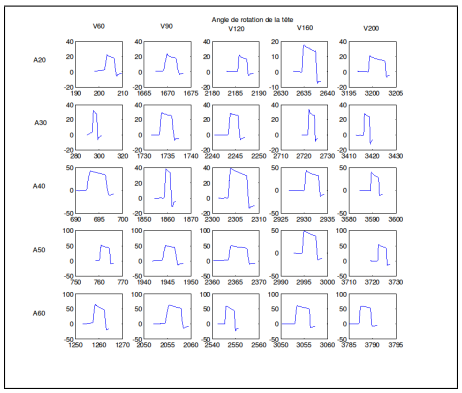

Figure 4.23 Exemple de données d'angles de rotation de la tête

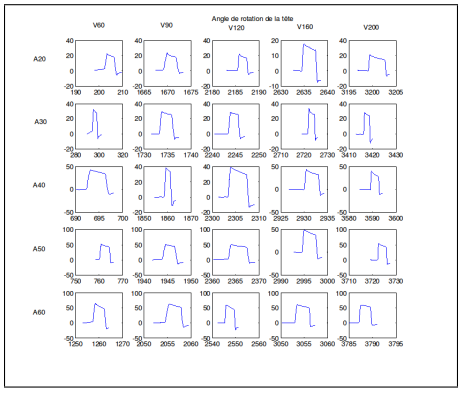

Figure 4.24 Exemple de données d'angles de rotation de la tête

Ainsi on fixe un angle maximum égal au plus grand atteint par un sujet, ce sujet aura un coefficient d'angle de 1. Les autres auront un coefficient défini comme le définit l'équation 4.23.

$$Coeff_{angle} = \frac{Angle\ max\ du\ sujet}{Angle\ max\ atteint\ par\ un\ sujet} \tag{4.23}$$

De la même manière un coefficient de vitesse angulaire sera appliqué. On fixe une vitesse angulaire maximum atteinte par un sujet, celui-ci aura un coefficient de vitesse angulaire de 1. Les autres auront un coefficient défini comme le définit l'équation 4.24.

$$Coeff_{vitesse} = \frac{Vitesse\ angulaire\ max\ du\ sujet}{Vitesse\ angulaire\ max\ atteint\ par\ un\ sujet} \tag{4.24}$$

Cela nous permet donc de fixer une efficacité globale pour chaque sujet qui sera calculée à partir de l'équation suivante :

4.5.1.8 Comparaison des données

Les valeurs d'efficacité globale de chaque sujet seront mises en commun et comparées les unes aux autres afin de savoir si le système fonctionne de manière similaire sur plusieurs personnes ou si ses performances sont variables d'une personne à l'autre. Une personne ayant une détection d'activité de 100% aura une efficacité de 1, une personne ayant une détection d'activité de 0% aura une efficacité de 0. Ainsi, plus un score d'efficacité est bon, plus il est proche de 1. Une efficacité moyenne, ainsi qu'une efficacité globale moyenne pourront être déterminées ainsi que leurs écarts types. Un seuil supérieur ou égal à la valeur de 0.75 pour l'efficacité globale permettra d'indiquer que le système est acceptable. En comparant les performances des sujets, si un sujet obtient un score plus faible que les autres, les données nous permettront

d’en étudier les causes, et permettront potentiellement de trouver des pistes d’amélioration du système.

CHAPITRE 5

RÉSULTATS

5.1 Détection d'activité

5.2 Temps de calcul et chronogrammes

5.3 Validation du système

CHAPITRE 6

DISCUSSION

CONCLUSION

Texte de conclusion

ANNEXE I

TITRE DE L'ANNEXE

S'il y lieu

1 Première Section de l'Annexe

<Texte à inserer>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque justo justo, porta sagittis feugiat eget, ornare rhoncus ligula. Nunc non odio sed lacus rutrum rhoncus. Mauris non congue arcu. Cras quis quam tortor. In ultrices tincidunt magna sed suscipit. Curabitur vel tellus sapien, ut tincidunt arcu. Maecenas dapibus ullamcorper urna, ut mollis mi tincidunt a. Nam eu orci nec lacus consectetur commodo. Donec purus tellus, consectetur at feugiat quis, scelerisque congue nibh. Aliquam urna dolor, congue nec euismod eget, convallis vitae libero. Sed vel magna suscipit leo suscipit porta quis et nunc. Nullam ante tellus, tincidunt a fringilla vel, rutrum non tellus. In volutpat consectetur purus, in euismod lorem feugiat vel. Aliquam sodales nisl eget sapien ullamcorper posuere consectetur orci bibendum. Vestibulum pulvinar viverra auctor. Vivamus ac sem et enim sodales dictum. Test citation **?**

Tests de figure en annexe.

$$2 * x = 4 \tag{A I-1}$$

Figure-A I-1 Logo de l'ÉTS dans l'annexe. Ici on va mettre un peu plus de texte pour voir comment va être la présentation de la légende dans ce cas.

Tableau-A I-1 Un autre tableau. Ici on va rédiger un peu plus de texte pour vérifier si la légende sera bien placé.

titre	**titre**	**titre**	**titre**	**titre**	**titre**	**titre**	**titre**
blá	blá	blá	blá	blá	blá	blá	blá
blá	blá	blá	blá	blá	blá	blá	blá
blá	blá	blá	blá	blá	blá	blá	blá
blá	blá	blá	blá	blá	blá	blá	blá
blá	blá	blá	blá	blá	blá	blá	blá
blá	blá	blá	blá	blá	blá	blá	blá

Test citation I-1.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque justo justo, porta sagittis feugiat eget, ornare rhoncus ligula. Nunc non odio sed lacus rutrum rhoncus. Mauris non congue arcu. Cras quis quam tortor. In ultrices tincidunt magna sed suscipit. Curabitur vel tellus sapien, ut tincidunt arcu. Maecenas dapibus ullamcorper urna, ut mollis mi tincidunt a. Nam eu orci nec lacus consectetur commodo. Donec purus tellus, consectetur at feugiat quis, scelerisque congue nibh. Aliquam urna dolor, congue nec euismod eget, convallis vitae libero. Sed vel magna suscipit leo suscipit porta quis et nunc. Nullam ante tellus, tincidunt a fringilla vel, rutrum non tellus. In volutpat consectetur purus, in euismod lorem feugiat vel. Aliquam sodales nisl eget sapien ullamcorper posuere consectetur orci bibendum. Vestibulum pulvinar viverra auctor. Vivamus ac sem et enim sodales dictum.

$$x = 42 \tag{A I-2}$$

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque justo justo, porta sagittis feugiat eget, ornare rhoncus ligula. Nunc non odio sed lacus rutrum rhoncus. Mauris non congue arcu. Cras quis quam tortor. In ultrices tincidunt magna sed suscipit. Curabitur vel tellus sapien, ut tincidunt arcu. Maecenas dapibus ullamcorper urna, ut mollis mi tincidunt a. Nam eu orci nec lacus consectetur commodo. Donec purus tellus, consectetur at feugiat quis, scelerisque congue nibh. Aliquam urna dolor, congue nec euismod eget, convallis vitae libero. Sed vel magna suscipit leo suscipit porta quis et nunc.

1.1 Test

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque justo justo, porta sagittis feugiat eget, ornare rhoncus ligula. Nunc non odio sed lacus rutrum rhoncus. Mauris non congue arcu. Cras quis quam tortor. In ultrices tincidunt magna sed suscipit. Curabitur vel tellus sapien, ut tincidunt arcu. Maecenas dapibus ullamcorper urna, ut mollis mi tincidunt a. Nam eu orci nec lacus consectetur commodo. Donec purus tellus, consectetur at feugiat quis, scelerisque congue nibh. Aliquam urna dolor, congue nec euismod eget, convallis vitae libero. Sed vel magna suscipit leo suscipit porta quis et nunc. Nullam ante tellus, tincidunt a fringilla vel, rutrum non tellus. In volutpat consectetur purus, in euismod lorem feugiat vel. Aliquam sodales nisl eget sapien ullamcorper posuere consectetur orci bibendum. Vestibulum pulvinar viverra auctor. Vivamus ac sem et enim sodales dictum.

mAuth1 (2001) mAuth2 (2002)

LISTE DE RÉFÉRENCES

mAuth1. 2001. « mTit1 ». *mJour1*, vol. 1, n° 1, p. 42-43.

mAuth2. 2002. « mTit2 ». *mJour2*, vol. 2, n° 2, p. 42-43.

BIBLIOGRAPHIE

Chang, Gwo-Ching, Wen-Juh Kang, Jer-Junn Luh, Cheng-Kung Cheng, Jin-Shin Lai, Jia-Jin J Chen, et Te-Son Kuo. 1996. « Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface ». *Medical engineering & physics*, vol. 18, n° 7, p. 529–537.

Chu, Jun-Uk, Inhyuk Moon, Yun-Jung Lee, Shin-Ki Kim, et Mu-Seong Mun. 2007. « A supervised feature-projection-based real-time EMG pattern recognition for multifunction myoelectric hand control ». *Mechatronics, IEEE/ASME Transactions on*, vol. 12, n° 3, p. 282–290.

Crawford, Beau, Kai Miller, Pradeep Shenoy, et Rajesh Rao. 2005. « Real-time classification of electromyographic signals for robotic control ». In *Proceedings of the National Conference on Artificial Intelligence*. p. 523. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999.

Englehart, Kevin et Bernard Hudgins. 2003. « A robust, real-time control scheme for multifunction myoelectric control ». *Biomedical Engineering, IEEE Transactions on*, vol. 50, n° 7, p. 848–854.

Englehart, Kevin, Bernard Hudgins, Philip A Parker, et Maryhelen Stevenson. 1999. « Classification of the myoelectric signal using time-frequency based representations ». *Medical engineering & physics*, vol. 21, n° 6, p. 431–438.

Fukuda, Osamu, Toshio Tsuji, Makoto Kaneko, et Akira Otsuka. 2003. « A human-assisting manipulator teleoperated by EMG signals and arm motions ». *Robotics and Automation, IEEE Transactions on*, vol. 19, n° 2, p. 210–222.

Li, Xiaoyan, Ping Zhou, et Alexander S Aruin. 2007. « Teager–Kaiser energy operation of surface EMG improves muscle activity onset detection ». *Annals of biomedical engineering*, vol. 35, n° 9, p. 1532–1538.

Maheu, V. 2011. « Développement des critères d'apprentissage pour le contrôle d'un bras robot manipulateur à 7DDL par le traitement des signaux EMG chez les blessés médullaires. ». Maîtrise en génie des technologies de la santé, Montréal, École de Technologie Supérieure, 133 p.

Martinez, R, R Munoz, L Leija, PR Hernandez, et Ja Alvarez. 1999. « A control unit for myoelectrically controlled prosthesis ». In *[Engineering in Medicine and Biology, 1999. 21st Annual Conf. and the 1999 Annual Fall Meeting of the Biomedical Engineering Soc.] BMES/EMBS Conference, 1999. Proceedings of the First Joint*. p. 641–vol. IEEE.

Maxim Integrated Products Inc, a. 2007. « Documentation des composants Max2222/Max3232/Max3237/Max3241 ». <http://datasheets.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf>.

Peleg, Dori, Eyal Braiman, Elad Yom-Tov, et Gideon F Inbar. 2002. « Classification of finger activation for use in a robotic prosthesis arm ». *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 10, n° 4, p. 290–293.

Prasad, G, S Srinivasan, et KM Patil. 1996. « A new DSP-based multichannel EMG acquisition and analysis system ». *Computers and biomedical research*, vol. 29, n° 5, p. 395–406.

Tenore, Francesco, Ander Ramos, Amir Fahmy, Soumyadipta Acharya, Ralph Etienne-Cummings, et Nitish V Thakor. 2007. « Towards the control of individual fingers of a prosthetic hand using surface EMG signals ». In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. p. 6145–6148. IEEE.