**Gaëlle THIBAUDAT**

**Informatique et Electronique des Systèmes Embarqués**
**POLYTECH GRENOBLE**
**Rapport de stage IESE4**

# Développement d'une prothèse de main myo-électrique

**Tome principal**

**2022/2023**
**Stage du 8 mai au 28 juillet 2023**

## REMERCIEMENTS

Je tiens à remercier chaleureusement M. Christophe BRAILLON de m'avoir acceptée comme stagiaire à l'INRIA, pour sa confiance, ses conseils tout au long de mon stage, le temps qu'il a pris pour m'expliquer des techniques et des théories, ainsi que pour son accompagnement patient et pédagogique dans le secteur de la Recherche et Dévéloppement.

Je tiens à exprimer ma gratitude à Guillaume Thomann, chargé de mission handicap de Grenoble INP et Elodie Tohdeim, du service des ressources humaines handicap d'Institut national de recherche en sciences et technologies du numérique (INRIA) Grenoble, pour m'avoir aidée dans mes recherches de stage et pour leur écoute attentive à mes besoins d'aménagements.

Je remercie également M. David EON d'avoir accepté d'être mon tuteur à l'école, ainsi que tous les professeurs qui ont pris le temps d'expliquer mes doutes pendant les cours, ce qui a contribué à mon meilleur développement durant mon stage.

Je tiens surtout à témoigner ma reconnaissance envers toute l'équipe SED et ses stagiaires qui ont fait preuve de patience face à ma surdité lorsque la communication verbale devenait difficile voire impossible. Leur bienveillance et les activités sympathiques proposées durant les pauses café et déjeuners du midi ont rendu l'expérience encore plus agréable

Je suis très reconnaissante de la présence de mon codeur LfPC, Pascal Geanty, qui m'a permis de suivre le séminaire à Lorient.

Enfin, je tiens à remercier ma famille et mes amis pour tout le soutien et les encouragements qu'ils m'ont apportés tout au long de ce parcours.

## ABRÉVIATIONS

**IDE** Integrated Development Environment. 23

**INRIA** Institut national de recherche en sciences et technologies du numérique. 1, 6, 37

**MHK** My Human Kit. 6, 37

**SED** Service Expérimentation Développement. 6, 37

## TABLE DES FIGURES


| | | |
|---|---|---|
| 1 | Schéma de cablâge avec les fiches bananes | 7 |
| 2 | Logo de My Human Kit | 8 |
| 3 | Nicolas Huchet, porteur du projet | 9 |
| 4 | Groupes de travail pour le projet Bionico | 10 |
| 5 | Diagramme de Gantt | 11 |
| 6 | Schéma de câbâge de l'architecture | 12 |
| 7 | Carte uC doté de RP2040 | 12 |
| 8 | Schéma du circuit de type pont H avec les paramètres | 13 |
| 9 | Schéma du fonctionnement de PWM | 15 |
| 10 | Extrait du code du fichier motor.h | 15 |
| 11 | Schéma de câblâge de PWM et ses connecteurs | 15 |
| 12 | Schéma de câblâge avec les connecteurs CM1 et CM2 | 16 |
| 13 | Schéma de cablâge avec les fiches bananes et le multimètre | 17 |
| 14 | Schéma de moteur bloqué, équivalent au montage de résistance | 18 |
| 15 | Prototype avec la pièce 3D imprimée, déplacée sur le rotor du moteur | 18 |
| 16 | Tracé de la courbe mesurant les courants A et B, du multimètre | 19 |
| 17 | Tracé de la courbe mesurant les courants A et B, du multimètre du moteur sans frottement (figure 2) | 20 |
| 18 | Electrodes | 21 |
| 19 | Algorigramme qui décrit le fonctionnement de l'ouverture et de la fermeture de la main | 21 |
| 20 | Prototype de la prothèse main-myoéléctrique avec les electrodes | 22 |
| 21 | Intégration générale de modules | 24 |
| 22 | Prototype final et testé par le porteur du projet | 27 |
| 23 | Tableau d'entrées du drive moteur MP6551 | 29 |
| 24 | Schéma du fonctionnement PWM | 29 |
| 25 | Fonctions permettant la configuration PWM | 30 |
| 26 | Configuration de sorties EN1 et EN2 | 30 |
| 27 | Configuration de sorties IN1 et IN2 | 31 |
| 28 | Configuration du contrôle de moteur | 31 |
| 29 | Extrait du fichier Makefile | 32 |
| 30 | Mode d'emploi du logiciel Pusa Slicer ode d'emploi du logiciel Pusa Slicer | 32 |
| 31 | Schéma du bras avant avec la contraction musculaire et les nerfs | 33 |
| 32 | Schéma de la pièce 3D pour le prototype sur FreeCAD | 33 |
| 33 | Extrait du code de la bibliothèque electrode.h | 34 |
| 34 | Extrat du code de la fonction ADC | 34 |
| 35 | Liste de contraintes pour la convention de code | 34 |

## TABLE DES MATIÈRES


**1 INTRODUCTION** **6**

**2 Contexte du stage** **7**

2.1 Présentation de l'entreprise 7

2.2 Service SED 7

2.3 HumanLab-INRIA 8

2.4 My Human Kit 8

2.5 Contexte du projet 9

2.5.1 Origine du projet 9

2.5.2 Problématique du projet 9

2.5.3 Objectifs 10

2.5.4 Cahier des charges 11

**3 Architecture matérielle** **12**

3.1 Commande moteur 12

3.1.1 Le drive moteur 13

3.1.2 La génération PWM 13

3.2 Mesure de courant moteur 16

3.3 Validation de mesures de courant avec le multimètre 17

3.4 Test de blocage du moteur 17

3.4.1 Etude sur le blocage de moteur et ses facteurs 18

3.4.2 Conception mécanique 18

3.4.3 Expérimentation de mesures de courant du moteur bloqué en mode PWM 19

3.5 Detection de mouvement de main 20

3.5.1 Etude sur les capteurs electrodes 20

3.5.2 Configuration d'éléctrodes 21

**4 Developpement logiciel** **23**

4.1 Outils et méthodes de développement 23

4.2 Linux 23

4.2.1 Visual Studio 23

4.2.2 Git 23

4.2.3 Convention codage 24

4.3 Intégration avec l'architecture matérielle 24

4.3.1 Organisation de code source et ses fonctionnalités 24

4.3.2 Processus de Déploiement et de Test 25

4.4 Résultats 26

4.4.1 Test de l'interface éléctronique avec le porteur du projet 26

**5 CONCLUSION** **28**

**6 Annexes** **29**

6.1 Configuration du drive moteur 29

6.2 Calculs de PWM 29

6.3 Mode d'emploi de la configuration PWM 29

6.4 Liste de fonctions PWM 30

6.4.1 Configuration de l'horloge PWM 30

6.4.2 Configuration des sorties EN1 et EN2 30

6.4.3 Configuration des sorties IN1 et IN2 31

6.4.4 Fonction permettant la commande de la vitesse et de la direction du moteur 31

6.4.5 Modification du fichier Makelist 32

6.5 Mode d'emploi du logiciel Pusa Slicer 32

6.6 Etude sur le mouvement de main 33

6.7 Conception 3D finale pour le prototype 33

6.8 Fonctions permettant la configuration d'éléctrodes 34

6.8.1 Ajout de la bibliothèque pour le fichier electrode.c 34

6.8.2 Lecture de valeurs ADC 34

6.8.3 Convention code 34

**RÉFÉRENCES** **35**

# 1 INTRODUCTION

Dans le cadre de mon diplome d'ingénieur en Informatique et Electronique des systèmes embarqués, j'effectue un stage de quatrième année, sous la tutelle de Christophe Braillon. Le stage a eu lieu au sein du Service Expérimentation Développement (SED) de l'INRIA sur le site de Montbonnot-Saint-Martin (38), précisement dans Inovallée, une technopole de l'innovation située à Meylan, pour douze semaines du 08 Mai 2023 au 30 Juillet 2023.

Ce rapport porte sur le sujet du développement d'une prothèse de main myo-électrique en collaboration avec HumanLab de Rennes et My Human Kit (MHK) et son porteur de projet utilisateur de ce type de prothèse lors de mon stage au sein du Service d'Expérimentation et Développement d'INRIA Grenoble. Le dévéloppement du matériel et du logiciel se fait en plusieurs parties dans ce rapport et il est réalisé en open-source pour que les autres personnes puissent fabriquer et adapter ces dispostifs à leurs besoins

Mon tuteur de stage, Christophe BRAILLON, m'a confié le développement de la prothèse de main myo-électrique, en se concentrant sur le travail de programmation et la préparation du code pour qu'il soit prêt à être utilisé avec les bibliothèques nécessaires. L'objectif du projet est de réaliser une prothèse fonctionnelle et adaptable, en fournissant une solution open-source.

Dans le cadre des missions attribuées par mon tuteur de stage, je suis chargé de réaliser une architecture matérielle en réponse au cahier des charge et en mettant l'accent sur les capteurs utilisés. Par la suite, nous détaillerons l'architecture logicielle mise en place pour assurer le bon fonctionnement du dispositif. Finalement, nous conclurons sur les résultats obtenus en soulignant les réussites et les difficultés que nous avons rencontrés, au cours de ce stage. Nous évoquerons les connaissances et les compétences acquises de cette expérience professionnelle.

Ce rapport est le fruit des travaux que j'ai menés lors de mon stage, ce qui nous permettra d'observer l'ensemble du projet et ses réalisations à travers les différentes étapes du développement.

# 2 Contexte du stage

## 2.1 Présentation de l'entreprise


Figure 1 – Bâtiment de l'INRIA de l'Université Grenoble-Alpes

L'INRIA est l'Institut national de recherche en sciences et technologies du numérique. Il a été créé en 1967 sous le nom d'IRIA (Institut de Recherche en Informatique et en Automatique) puis a été renommé INRIA (Institut national de recherche en informatique et en automatique) en 1979. Sa création s'inscrivait dans le cadre du plan de développement de l'industrie française pour répondre aux besoins croissants dans le domaine du numérique. Il a acquis une reconnaissance internationale en raison de la qualité de ses recherches et de sa contribution à l'innovation technologique. Il regroupe environ 215 équipes-projets et compte plus de 3 900 chercheurs et ingénieurs de 105 nationalités différentes. Ces équipes sont réparties dans 8 centres de recherche situés en France, dans des villes telles que Paris, Lyon, Rennes, Bordeaux, Lille, Nice, Nancy, Saclay. Ces centres sont implantés à proximité de grandes universités de recherche, dont l'Université Grenoble Alpes, en collaboration avec des partenaires industriels.

L'objectif principal de l'INRIA est de relever les défis futurs dans le domaine des sciences et technologies numériques.

## 2.2 Service SED

SED-GRA (Service Expérimentation Développement Inria Grenoble Rhône-Alpes) est un service de soutien aux équipes de recherche de l'INRIA, au niveau de l'expérimentation et du dévéloppement logiciel. Il est constitué d'une quinzaine d'ingénieurs de recherches.

## 2.3 HumanLab-INRIA

HumanLab-Inria est un espace collaboratif de fabrication numérique ou de réparation d'objets pour les personnes présentant un handicap. Il regroupe plusieurs groupes d'ingénieurs et chercheurs du réseau INRIA, qui a pour objectif de répondre aux besoins exprimés par des individus porteurs d'handicap dans le cadre du réseau des Humanlabs et des partenaires cliniques, en collaboration avec l'association My Human Kit. Ils participent à plusieurs projets dont Bionicohand, Fit&Fun, etc en dévéloppant des outils technologiques. Chaque année Human-Lab-INRIA contribue à un ou deux événements nommés Fabrikarium dont l'objectif est de dévélopper plusieurs projets avec des groupes d'une dizaine de personnes avec des profils et des compétences variées. Parallèlement, une activité de médiation scientifique et de sensibilisation au handicap est aussi proposée afin d'accompagner et sensibiliser les enseignants de collèges et écoles primaires dans la mise en place de projets de robotiques et de programmation informatique.

## 2.4 My Human Kit

My Human Kit (MHK) est une association de logiciel open-source du numérique, crée en 2014 à Rennes, suite à la concrétisation du projet Bionicohand en 2013. Il est actuellement présidé par Charlie Dréano.


Figure 2 – Logo de My Human Kit

L'objectif de My Human Kit est de créer un espace collaboratif de partage d'inventions, de savoirs et d'aides techniques pour les personnes en situation de handicap, en misant sur la création de prototypes à moindre coût réalisés en DIY, afin de répondre aux besoins spécifiques de ces personnes. L'association vise la democratisation des savoirs en réunissant des équipes pluridisciplinaires issues de différents coins de la France et du monde entier, comprenant des ingénieurs, des chercheurs, des techniciens, des étudiants, et des bénévoles. L'objectif de My Humam Kit est de faciliter l'accès aux aides techniques pour les personnes en situation de handicap, en les rendant accessibles sur le plan financier et en facilitant leur utilisation. Depuis la création du premier HumanLab en région rennaise, d'autres ont émergé un peu partout en France et à l'étranger, comme HumanLab à Grenoble où je travaille.

## 2.5 Contexte du projet

Dans le cadre de mon stage chez INRIA, j'étudie la partie éléctronique du projet prothèse de main myoéléctrique, nommé Bionicohand v2, en collaboration avec l'association My Human Kit et plusieurs acteurs impliqués tels qu'ArianeGroup, Orthopus, Co Work'Hit. Le porteur du projet est Nicolas Huchet, lui-même utilisateur et co-fondateur de My Human Kit.

### 2.5.1 Origine du projet

Figure 3 – Nicolas Huchet, porteur du projet

Le projet Bionicohand trouve ses racines en 2012, lorsque Nicolas Huchet, amputé de l'avant-bras et équipé d'une prothèse à pince, découvre le fablab (laboratoire de fabrication numérique) à Rennes. C'est à ce moment où il fait connaissance avec les imprimantes 3D et le concept de "maker," où des mains imprimées en 3D sont conçues et diffusées. C'est ainsi qu'est né le projet Bionicohand, en 2013 avec l'idée de créer une prothèse de main myoélectrique à moindre coût en utilisant ces techniques et l'adaptant aux besoins particuliers d'utilisateurs en termes de poids, d'esthétisme et de maniabilité. Cette solution facilite la vie des utilisateurs amputés du membre supérieur, leur permettant de retrouver une autonomie au quotidien tout en leur offrant la possibilité de personnaliser leur bras pour changer son regard sur le handicap.

### 2.5.2 Problématique du projet

Le coût élevé des prothèses poly-digitales, qui peut atteindre entre 30 000€ et 70 000€ et ne sont pas remboursées par la sécurité sociale, représente un grand obstacle majeur à l'accès à ces dispositifs. Selon OMS [1], 9 personnes sur 10 ayant besoin de l'aide technique n'y ont pas accès, ce qui explique le coût elevé de prothèses. Les pays émergents sont particulièrement touchés car ils souffrent du manque de l'accessibilité à l'appareillage prothétique qui est limitée dû à l'absence de politique publique de santé, un manque de sensibilisation, de disponibilité, de personnel formé, de financement. Le taux élevé de rejet des

prothèses s'explique par des problèmes liés au poids, à l'esthétisme et à la maniabilité des prothèses ce qui démontrant que la majorité de prothèses fabriquées ne sont pas adaptés aux besoins d'utilisateurs. En outre, le marché de prothèses de membre supérieur étant de niche, le modèle économique actuel ne permet pas de proposer des prothèses à un prix raisonnable.

Face à ces défis, il est essentiel de savoir comment développer une prothèse de main myoélectrique abordable, ergonomique et durable, avec la possibilité de maintenance, capable de répondre aux besoins spécifiques des utilisateurs tout en leur favorisant une autonomie quotidienne et une plus grande accessibilité au large public.

### 2.5.3 Objectifs

L'objectif général du projet est de réaliser une prothèse de main myoélectrique accessible et adaptable, en proposant une solution open-source à moindre coût, qui puisse être accessible partout dans le monde.

Le projet comprend plusieurs parties du travail, comme la partie mécanique, qui sont réparties entre différents acteurs impliqués dont INRIA, comme montre la figure 4

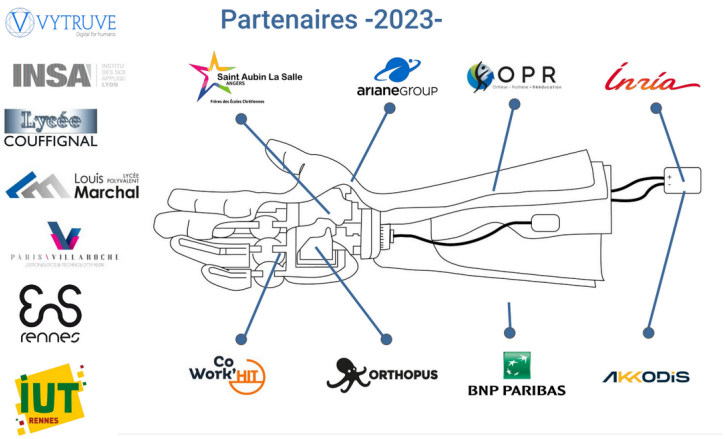


Figure 4 – Groupes de travail pour le projet Bionico

Dans le cadre de stage, on se concentre spécifiquement sur la partie électronique du projet, en particulier sur le dévéloppement du matériel et du logiciel, réalisé en open-source. Ce type de prothèse est utilisé par des personnes amputés au niveau du poignet et utilise les signaux éléctriques de contraction des muscles de l'avant bras.

Notre objectif principal est de concevoir un système logiciel capable de commander l'ouverture et la fermerture de la main à l'aide de capteurs éléctrodes dont on dispose déjà. La

carte éléctronique de traitement du signal des électrodes et de commande moteur ont déjà conçus par Christophe Braillon, en janvier dernier. Cette carte µC est basée sur la puce RP2040 de Raspberry.

Mon rôle est de réaliser un code embarqué sur cette carte en prenant en compte du cahier de charges.

### 2.5.4 Cahier des charges

Le cahier de charges a été établi, pour répartir les tâches que je devais faire au long du stage, en respectant les critères ci-dessous :

- Gestion de l'alimentation
- Acquisition et traitement du signal myo-éléctrique
- Détection des intentions d'ouverture/fermeture de la main
- Commande moteur de couple (courant) et détection de blocages
- Réaliser et tester par itération successives les différents prototypes en collaboration avec le porteur du projet
- Documentation du dispositif sur un wikilab

Le déroulement de stage se fait en plusieurs étapes comme décrit la figure 5

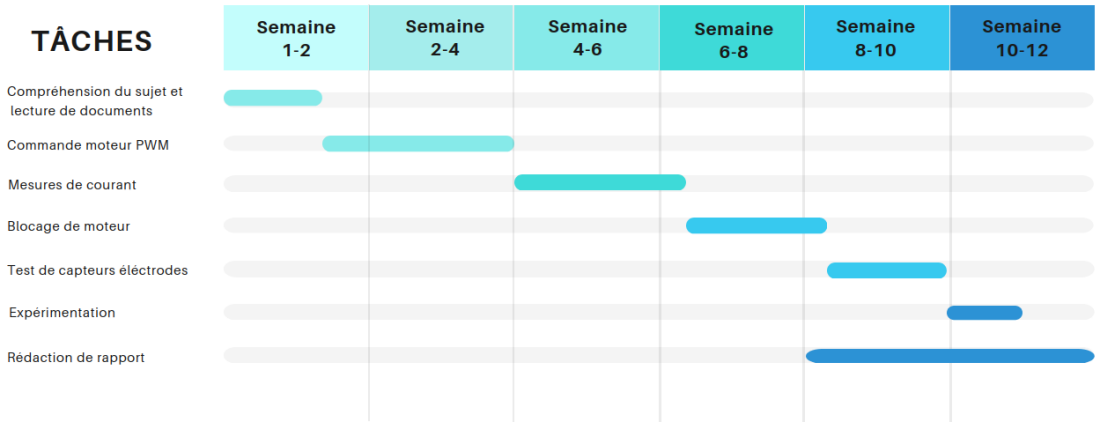


Figure 5 – Diagramme de Gantt

# 3 Architecture matérielle

Cette première partie se concentre sur l'architecture matérielle, qui consiste à créer un programme de commande pour le moteur, en développant une bibliothèque spécifique qui pourra être utilisée ultérieurement par d'autres utilisateurs pour des prototypes similaires. L'objectif principal est de garantir un contrôle précis du moteur, en prenant en compte la direction et d'autres paramètres importants comme les mesures de courant, le blocage de moteur, la lecture d'electrodes.

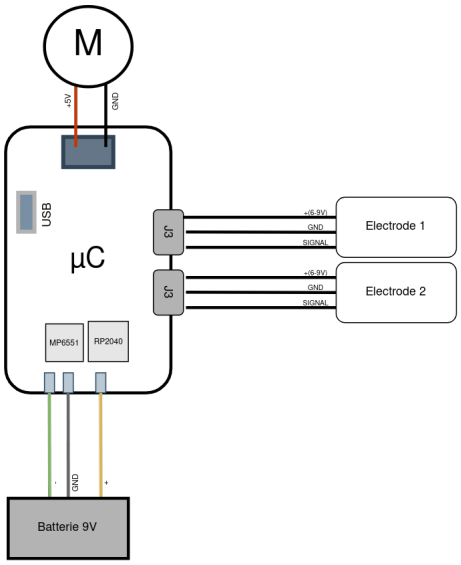


Figure 6 – Schéma de câbâge de l'architecture

## 3.1 Commande moteur

La commande du moteur dans ce projet est réalisée à l'aide d'un microcontrôleur RP2040, d'un drive moteur MP6551 et du moteur à courant permettant l'ouverture et la fermeture de la main. Le microcontrôleur RP2040 7est doté de la génération des signaux de commande et de leur envoi au drive moteur pour contrôler la vitesse et la direction de rotation du moteur.

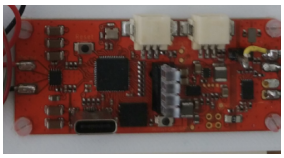

Figure 7 – Carte uC doté de RP2040

L'objectif de la commande moteur est d'assurer un contrôle précis du moteur, en prenant en compte la vitesse, la direction et la protection contre les blocages du moteur. Cette solution

permet de garantir le bon fonctionnement du moteur avec ses fonctionnalités pour le projet.

### 3.1.1 Le drive moteur

Nous utilisons un moteur à courant continu assisté par un drive moteur MP6551 afin d'obtenir un contrôle précis sur la vitesse et le sens de rotation du moteur en ajustant avec précision le courant et la tension. Le drive moteur MP6551, reçoit des signaux de commande du microcontroleur uC et les convertit en signaux analogiques pour alimenter et commander le moteur. Il permet aussi de protéger le moteur en surveillant les mesures de courant et en détectant les blocages du moteur.

Le drive moteur MP6551 s'appuie sur le circuit de type pont en H avec 4 entrées de commande IN1, IN2, EN1 et EN2, dont la plage varie entre 2.5V et 14V. Grâce à cette méthode de pont, les entrées IN1 et IN2 déterminent le fonctionnement des transistors de chaque branche du pont en H, en appuyant sur une table logique définie comme illustre la figure 8 et le tableau 6.1 d'après les informations sur la documentation MP6551 [2].

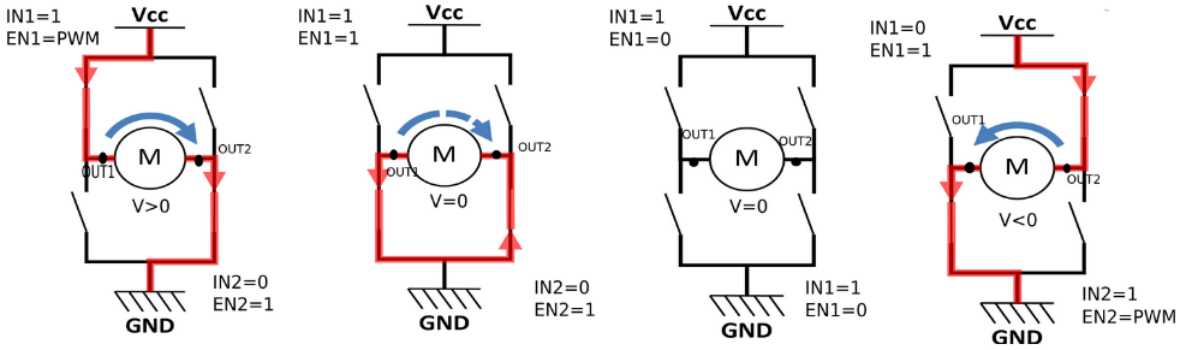


Figure 8 – Schéma du circuit de type pont H avec les paramètres (Voir Annexe 6.1)

Les entrées EN1 et EN2 servent à mettre les pin OUT1 et OUT2 en haute impédance. Si on met EN1 ou EN2 à 0, le moteur est en roue libre, ce qui signifie que le courant ne circule plus vers le moteur. Pour faire varier la vitesse, il faut donc hacher la tension moteur en appliquant une PWM sur EN1 ou EN2. Cette méthode permet l'ouverture et la fermeture de la main en changeant le sens du moteur

### 3.1.2 La génération PWM

Le microcontrôleur integré de la puce RP240 dispose d'un périphérique PWM intégré, qui permet de générer des signaux de commande analogiques à partir de signaux numériques. La génération PWM est nécessaire pour contrôler la vitesse du moteur en ajustant le rapport

cyclique des signaux PWM. Le fonctionnement du périphérique PWM est décrit dans la documentation de RP240 [3]. La configuration de la broche d'E/S en mode PWM sur le RP2040 est indiquée dans la documentation MP6551 [2]. Une fois configurée, la broche peut être utilisée pour générer des signaux PWM de sortie.

Le périphérique PWM est cadencé par une horloge qui est issue de la fréquence d'horloge système ($f_{sys} = 133\mathrm{MHz}$ par défaut si on ne la reconfigure pas). Cette horloge est ensuite divisée par un diviseur $D$. La fréquence du compteur est donc de $f_{count} = \frac{f_{sys}}{D}$.Le compteur est incrémenté à chaque pas d'horloge jusqu'à atteindre une valeur $TOP$, il est ensuite remis à $0$ et l'incrémentation continue.

La fréquence de la PWM est dans ce cas :

$$f_{PWM} = \frac{f_{count}}{TOP} \frac{f_{sys}}{D \times TOP} \tag{3.1}$$

D'après la documentation [2], on a $t_{timing}$<3µS et on le multiplie par 10, ce qui donne T»30µs.

La condition s'impose : f <33KHz

On convertit la fréquence PWM en période PWM :

$$T_{PWM} = \frac{Max * N}{F_{clock}}$$

L'application numérique à partir de valeurs a été réalisée pour obtenir la valeur Max et la valeur N (voir Annexe 6.2)

**Configuration de PWM**

Après avoir étudié théoriquement le fonctionnement de PWM et fait les calculs (cf. Annexe 6.2), il est nécessaire de configurer les fonctions de PWM avec ces données tel que le temps de génération PWM, l'horloge PWM, et la vitesse PWM. Les fonctions sont déjà crées et prêtes à utiliser à l'aide de la bibliothèque "pwm.h" et du site Raspberry [4] pour connaître ses caractéristiques de fonctions.

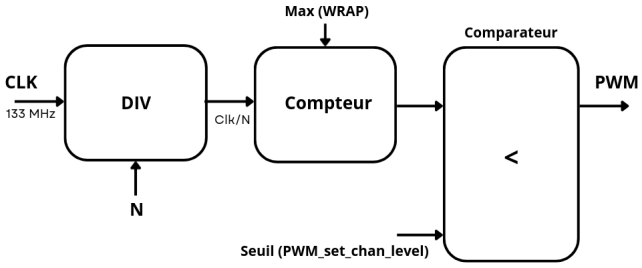


Figure 9 – Schéma du fonctionnement de PWM

La figure 9 décrit le fonctionnement de PWM avec l'horloge PWM, le diviseur, le compteur et les rôles de fonctions qu'on utilise pour coder. Prenons l'exemple de deux fonctions nécessaires permettant la configuration de PWM :

```
1  ...
2
3  #define MAX_PWM 100
4  pwm_config config = pwm_get_default_config();
5  pwm_config_set_clkdiv(&config, 40);        (1)
6  pwm_config_set_wrap(&config, MAX_PWM);     (2)
7
8  ...
```

Figure 10 – Extrait du code du fichier motor.h

(1) Cette fonction permet de definir le diviseur d'horloge PWM représenté par la variable N, qui vaut 40, dans la configuration PWM.

(2) Cette fonction permet de définir la valeur de bouclage du compteur PWM dans la configuration PWM indiquée ici par $MAX_PWM$, ayant une valeur de 100

Il est important de noter que les valeurs de N et $MAX_PWM$ peuvent être ajustées en cas de modification de la vitesse du moteur.

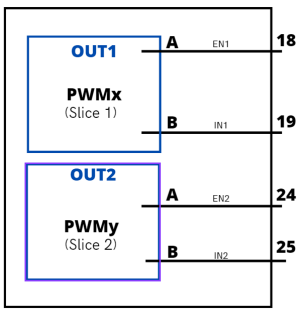


Figure 11 – Schéma de câblâge de PWM et ses connecteurs

Après avoir mené l'analyse sur l'utilisation des ports en appuyant sur la documentation RP240, nous pouvons observer sur la figure 11 les connecteurs nécessaires et leurs numéros de ports pour configurer les sorties OUT1 et OUT2. Ces connexions sont établies à l'aide des fonctions de la bibliothèque "pwm.h", soigneusement sélectionnées en fonction des besoins spécifiques du projet. Ces fonctions sont codées et décrites dans l'annexe 6.4

Les informations de l'ensemble de fonctions nécessaires pour la configuration PWM sont disponibles et accessibles dans l'onglet PWM du site Raspberry [5]

## 3.2 Mesure de courant moteur

La mesure du courant moteur se fait grâce aux sorties 'CM1' et 'CM2' du MP6551. La sortie 'CM1' mesure le courant dans la branche du pont en H correspondant à 'OUT1' et 'CM2' dans celle correspondant à 'OUT2' (voir Référence [2] p9) comme illustre la figure 12.

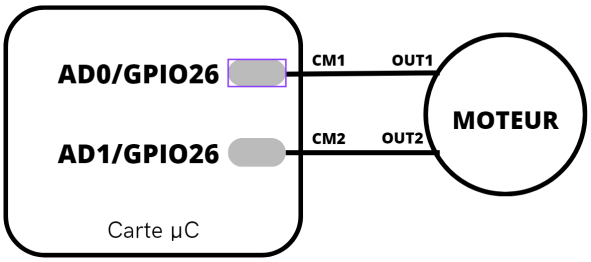


Figure 12 – Schéma de câblâge avec les connecteurs CM1 et CM2

La page 10 de la référence [2] indique que les sorties 'CMx' sont des sorties courant. Le courant de sortie est l'image directe du courant circulant dans les deux branches du pont en H. Le courant en sortie des 'CMx' est le courant circulant dans les branches du pont divisé par 10000. Le montage en sortie de 'CM1' et 'CM2' permet de passer d'un courant à une tension.

Avec le montage présent, la relation entre le courant dans le moteur et la tension en entrée de l'ADC est :

$$ADC = \frac{V_{CC}}{2} + \frac{I_{LOAD}}{10000} \times 5100$$

## 3.3 Validation de mesures de courant avec le multimètre

On vérifie les valeurs de courant A et B avec le multimètre. Dans ce cas, on modifie le montage de câblage du prototype. On désoude un des fils de moteur pour relier le fil positif à celui de fiche banane. Le moteur à courant continu, équipé d'un rotor, a été choisi pour effectuer des mesures expérimentales.

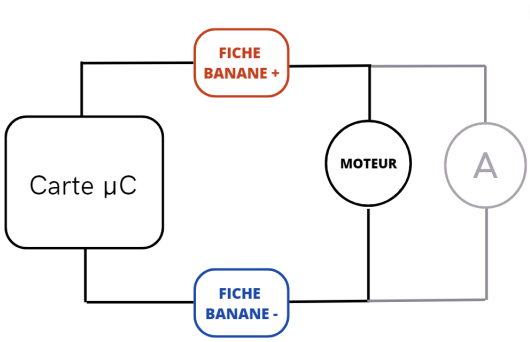


Figure 13 – Schéma de cablâge avec les fiches bananes et le multimètre

Ce montage permet de valider la mesure de courant au multimètre en faisant un relevé du courant qu'on lit sur la sortie ADC et en comparant ces valeurs avec le courant réel qui traverse le circuit. Cette méthode nous a permis de mieux comprendre les disparités de courant mesuré sur la sortie ADC. Nous avons identifié que ces disparités pourraient être réduites en ajoutant un condensateur d'une capacité de 100 à 1000 uF ou une résistance. Nous avons expérimenté cette hypothèse en ajoutant un condensateur de 1000 uF, mais les résultats ont montré peu de modification dans les mesures de courant. Il est important de rappeler qu'un multimètre possède deux modes de mesure : l'ampèremètre et le voltmètre. Étant donné que nous nous intéressons aux mesures de courant, nous choisirons le mode ampèremètre et nous sélectionnerons le calibre 10A de type DC sur le multimètre pour effectuer nos mesures.

## 3.4 Test de blocage du moteur

On réalise un test de blocage moteur qui détecte le dépassement du courant à la limite. Les mesures du courant sont realisés par l'µC. Ce dispostif permet de minimiser les risques de surtension, de surcourant et de décallage mécanique. Dans la première partie, on étudie sur les phénomènes du moteur bloqué. Ensuite, on observe les données mesurées et les compare avec les courbes tracées sur GNUPLOT.

### 3.4.1 Etude sur le blocage de moteur et ses facteurs

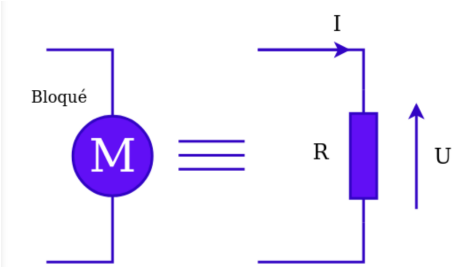


Figure 14 – Schéma de moteur bloqué, équivalent au montage de résistance

Il est essentiel de se rappeler que la puissance utilisée par un moteur est le produit de la tension et de courant, qui le traversent. Cela déduit l'ajout de résistance dans le schéma, comme indique la figure 14. Lorsque le courant ou la résistance augmentent, la tension du moteur sera elevée. Quand nous souhaitons freiner le moteur, une résistance ajoutée est introduite dans son mouvement. L'effet de résistance entraîne l'augmentation du courant nécessaire pour faire tourner le moteur car cela nécessite de la puissance exprimée par la formule U=R*I, pour maintenir la rotation. C'est pourquoi on observons une augmentation du courant lorsque le moteur est soumis à des frottements :

$$ratio = \frac{T_h}{T} = 50$$

### 3.4.2 Conception mécanique

Pour obtenir des mesures de courant bloqué, il est nécessaire de générer du frottement dans le moteur, ce qui entraîne l'effet de résistance et provoque une augmentation du courant. Dans ce cas, nous avons créé une pièce 3D rotative (Photo 15) à l'aide du logiciel FreeCAD, qui sera ensuite assemblée sur le rotor du moteur. L'impression de cette pièce est réalisée grâce à Prusa Slicer, en utilisant l'imprimante 3D située dans l'atelier de l'INRIA. Cela requiert une compréhension des techniques de manipulation (voir Annexe 6.5)

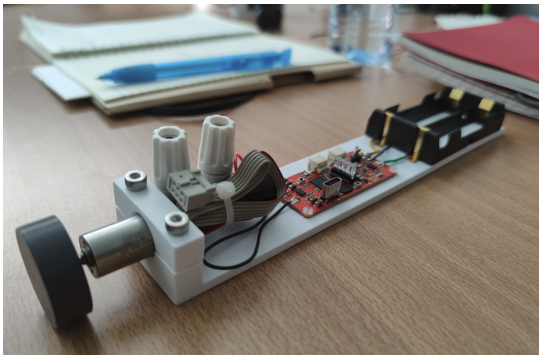

Figure 15 – Prototype avec la pièce 3D imprimée, déplacée sur le rotor du moteur

Après avoir réalisé la première impression et l'avoir testée, nous avons remarqué une usure du trou après seulement trois cycles, ce qui a néccesité une nouvelle impression avec de nouvelles dimensions ajustées. Cependant, ces dimensions sont sous-estimées, rendant impossible l'assemblage avec le moteur. Pour rémédier à ce problème, nous avons effectué un troisième impression en supprimant la forme polygone et en ajoutant un trait sur le haut du cercle. Cette modification a permis de mieux bloquer la rotation du moteur et d'assurer un ajustement correct.

### 3.4.3 Expérimentation de mesures de courant du moteur bloqué en mode PWM

On prend les mesures du courant du moteur bloqué mesuré par le multimètre et celles du courant A et B mesurés sur µC [6] afin d'observer les phénomènes de blocage de moteur. Pour bien vérifier le fonctionnement du capteur, il est nécessaire d'effectuer plusieurs essais de mesure en mode PWM.

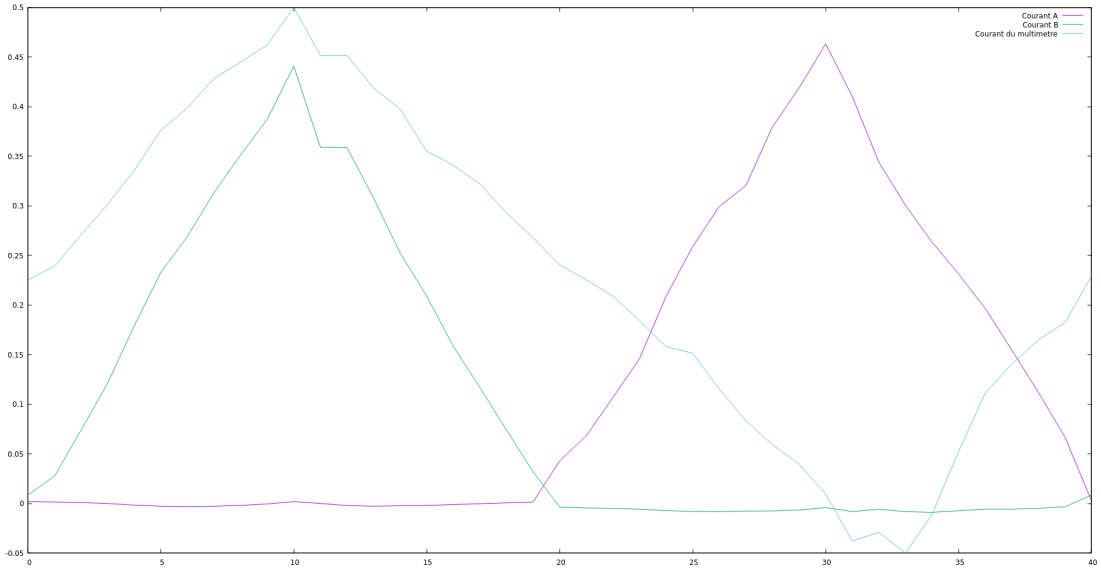


Figure 16 – Tracé de la courbe mesurant les courants A et B, du multimètre

D'après la figure 16, on remarque qu'en bloquant le moteur, le courant augmente plus vite. Le courant B s'approche de celui de courant mesuré par le multimètre. Le courant A n'est pas le meme que celui mesuré par le multimètre, ce qui signifie que les courants A et B ont les valeurs positifs alors que le courant mesuré par le multimètre connait les valeurs négatives. Par rapport à la courbe de mesures du courant du moteur non frotté comme illustre la figure 17, les pentes sont plus raides ce qui explique l'augmentation rapide du courant du moteur en cas de frottement.

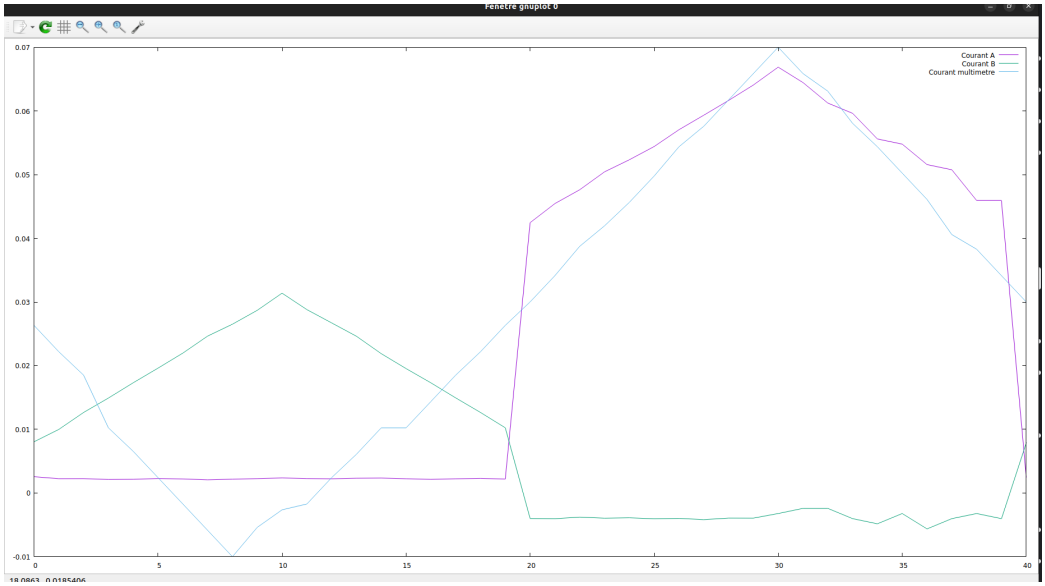


Figure 17 – Tracé de la courbe mesurant les courants A et B, du multimètre du moteur sans frottement

Nous choisissons 0.25A comme le courant de limite pour blocage de moteur pour µC alors que la valeur réelle de limite de courant mesurée par le multimètre est 1A.

## 3.5 Detection de mouvement de main

Nous menons une étude approfondie de la détection du mouvement de la main, dans le but de mieux comprendre son fonctionnement lorsqu'elle est en mouvement. Cette compréhension servira pour la configuration des électrodes. Cette étape impliquera des recherches documentaires concernant ses connecteurs, ainsi que la programmation requise pour la détection du mouvement de la main avec les paramètres nécessaires comme, par exemple, la valeur du courant de seuil.

### 3.5.1 Etude sur les capteurs electrodes

Nous allons utiliser des capteurs électrodes qui seront placées sur un bras afin de détecter l'activité musculaire lors du mouvement de la main (voir Annexe 6.6). Ils mesurent l'activité musculaire en mesurant les signaux électriques générés par les muscles lorsqu'ils se contractent. Le seuil de tension mesurée par les électrodes sera utilisé pour contrôler l'ouverture et la fermeture d'une main myoélectrique. Les électrodes sont fabriquées par l'entreprise Orthopus et sont des "clones" d'un capteur Ottoback.

On s'interesse aux caractéristiques d'éléctrodes pour pouvoir les configurer afin de détecter les contractions musculaires qui seront converties en courant. Comme indique la figure 18, les


Figure 18 – Electrodes

éléctrodes possèdent trois sorties : tension entre 6 et 9V, la masse et le signal. Il faudra faire attention aux sens de fils, pour les connecteurs, qui seront changés pour s'adapter à celui de la carte µC. D'après l'annexe 6.6, il y a deux types de nerfs dans le bras, ce qui nous interesse pour les mesures. En effet, concernant la position et l'orientation des électrodes du capteur musculaire, elles ont un impact significatif sur la qualité du signal mesuré.

### 3.5.2 Configuration d'éléctrodes

Selon le schéma éléctrique de la carte uC [7] , les entrées EL1/J2 et EL2/J3 sont requises pour brancher les connecteurs d'éléctrodes. Les valeurs de courant qu'émettent les éléctrodes sont analogiques. Le convertisseur ADC permet de les convertir en valeurs numériques afin de récupérer les valeurs réelles du courant. Dans ce cas, nous sélectionnons l'entrée J2 comme entrée principale du connecteur d'un électrode pour la phase de test. Pour activer les ports ADC, nous utilisons la bibliothèque *adc.h*. Pour le bon fonctionnement d'éléctrodes, j'ai crée un nouveau fichier "test_electrode.c" à l'aide de fichiers complémentaires electrode.h et electrode.c. J'ai également mis à jour le fichier CMakeList.txt pour compiler et éxécuter.

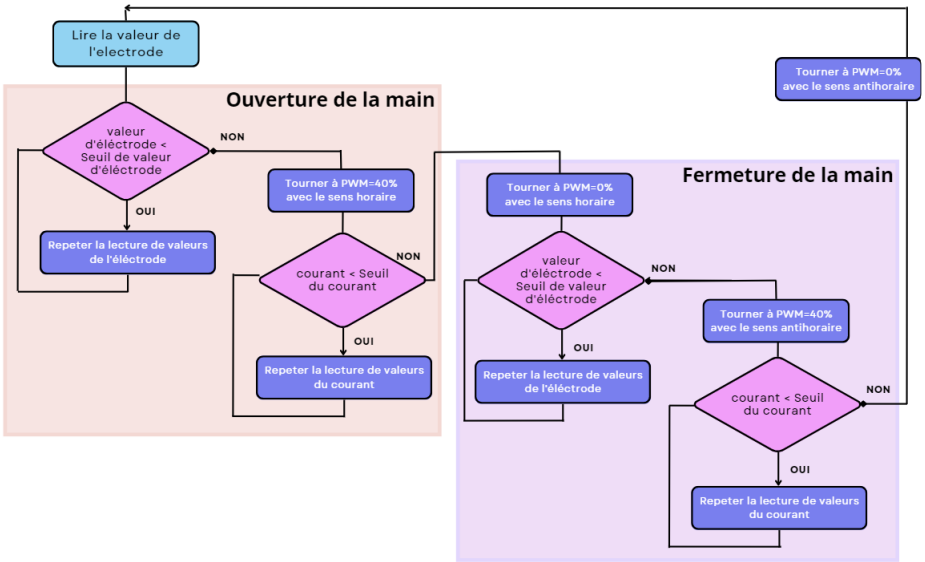


Figure 19 – Algorigramme qui décrit le fonctionnement de l'ouverture et de la fermeture de la main

Le programme comme décrit la figure 19 permet d'effectuer l'ouverture et la fermeture de la main en réponse à la réception d'un signal de l'impulsion, ce qui signifie que la main s'ouvre ou se ferme en cas de contraction musculaire. A chaque itération de la boucle, il est essentiel d'effectuer la lecture de valeurs d'éléctrodes afin de vérifier si les conditions requises (définies par la boucle while) sont satisfaites. Cette lecture des valeurs de courant joue le rôle de l'ouverture et de la fermeture de la main. Pour son bon fonctionnement, elle nécessite une valeur seuil de courant préalablement calibrée. Ce calibrage est nécessaire pour assurer le bon déroulement des opérations de la prothèse. Cependant, il est important de noter que la précision de la position des électrodes sur l'avant-bras peut présenter des défis lors de la procédure de calibrage, rendant ainsi cette étape plus complexe. Le changement de direction du moteur se produit lorsque le courant de seuil est excédé à chaque cycle itératif de la boucle. L'ensemble des itérations, avec des paramètres variables tels que la vitesse et la direction du moteur, est répété, reproduisant ainsi la même séquence après la dernière boucle.

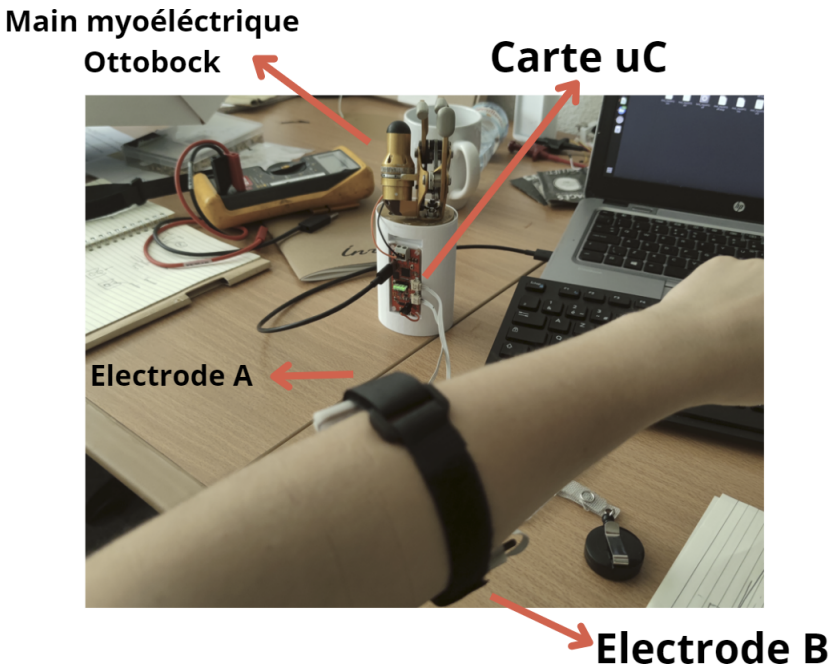


Figure 20 – Prototype de la prothèse main-myoéléctrique avec les electrodes

La figure 20 montre l'ensemble du prototype comprenant le moteur de main Ottobock, la carte éléctronique et les éléctrodes Orthopus, qui seront placés sur le bras avant. La conception 3D finale est réalisée pour créer un support pour la carte uC et le moteur, qui joue le role de l'ouverture et de la fermeture de la main, comme montre l'annexe 6.7. Ce dispositif est prêt à être soumis à la phase de test avec le porteur du projet, Nicolas Huchet.

# 4 Developpement logiciel

## 4.1 Outils et méthodes de développement

## 4.2 Linux

Nous utilisons Linux qui est un système d'exploitation open-source pour la création du code embarqué au coeur de notre projet. Linux est couramment adopté par les développeurs pour son accessibilité et sa grande flexibilité en termes de programmation et de compilation, notamment à travers l'utilisation d'une console. Il joue son rôle central dans la réalisation de l'architecture logicielle à l'aide de Visual Studio et de la commande console. Pour faciliter l'utilisation de la console et la compilation en lien avec l'architecture matérielle, nous avons élaboré une fiche d'astuces (voir Annexe ) répertoriant les commandes essentielles. Cette ressource s'est avérée utile pour interagir efficacement avec la console et compiler les programmes adaptés à notre architecture matérielle.

### 4.2.1 Visual Studio

Pour faciliter l'organisation de code embarqué, nous faisons usage de Visual Studio, environnement de devéloppement intégré (Integrated Development Environment (IDE)) reconnu pour son nombre de fonctionnalités et sa prise en charge de plusieurs langages de programmation. Visual Studio est notre principal outil pour la rédaction, la modification, l'organisation des fichiers de code et le débogage. Son utilisation simplifie considérablement l'avancement du développement en offrant un environnement complet et efficace.

### 4.2.2 Git

Pour partager les fichiers de code en ligne, nous utilisons Git, un système de contrôle de version, integré à notre projet pour une gestion efficace du code source. Nous avons crée un dépôt de code source ainsi que ses branches sur la plateforme Gitlab INRIA [8] ce qui nous permet de travailler sur les différentes fonctionnalités comme commande moteur et la lecture d'electrodes, par exemple. Cet outil s'est averé être un outil indispensable pour suivre l'historique des modifications, de récupérer des versions antérieures du code et préserver l'intégrité du projet.Le code source est déposé dans un dépôt git dédié (repository). Il est important de réaliser des commit réguliers à chaque étape du développement pour garder une trace du cheminement et pouvoir faire des retours en arrière, en cas de besoin.

Pour chaque développement de fonctionnalité, il est recommandé de créer une branche distincte. Cette branche est fusionnée avec la branche principale une fois que la fonctionnalité est jugée complète, testée et validée. Par exemple, c'est le cas du code de commande moteur, où cette méthode permet d'assurer une gestion organisée et contrôlée du développement des

fonctionnalités. En outre, nous avons réalisé une fiche d'astuces [9] pour maitriser mieux Git et ses fonctionnalités.

### 4.2.3 Convention codage

La convention codage (voir Annexe 6.8.3) a été mise en place par Christophe, mon tuteur de stage, pour maintenir la bonne lisibilité et la cohésion du code au long de stage, ce qui facilite la gestion du projet avec la structure du code, la maintenance du code et surtout la compréhension commune au sein de l'équipe.

## 4.3 Intégration avec l'architecture matérielle

### 4.3.1 Organisation de code source et ses fonctionnalités

L'organisation méthodique du code source est essentiel pour la réussite d'un projet logiciel. Dans cette partie, nous menons une approche modulaire pour structuré notre code source et intégrer les différentes fonctionnalités du projet.

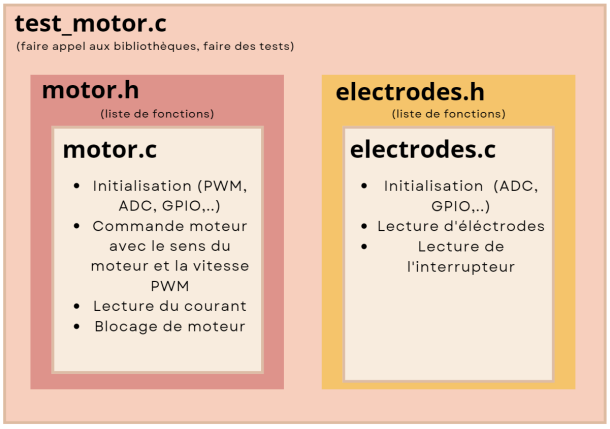


Figure 21 – Intégration générale de modules

D'abord, nous avons dévéloppé le module principal test motor, qui compose de deux bibliothèques comme illustre la figure 21. Cette méthode permet de mettre en oeuvre la commande moteur et la lecture d'éléctrodes de manière cohérente. Concernant la bibliothèque motor, joue le role de controle du moteur. Cela nous permet de gérer les paramètres de vitesse et de direction, en assurant une interaction fluide avec les electrodes. La bibliothèque lecture electrodes.h a été intégrée pour permettre la détection du mouvement de la main. Cette étape implique la lecture des valeurs de courant des électrodes, ce qui contribue à la commande du moteur en réponse aux signaux musculaires. Cette démarche d'intégration dans test moteur est une illustration de notre approche globale d'organisation du code source.

### 4.3.2 Processus de Déploiement et de Test

L'intégration du logiciel avec l'architecture matérielle est une étape importante. Nous avons mis en place un processus de déploiement basé sur le "flashing" à l'aide de trois commandes Linux (cmake, make, sudo pyserial-miniterm), pour transférer le code compilé sur le microcontrôleur RP240 via la connexion USB et effectuer des tests. Cette démarche permet de charger le programme sur la carte et de mettre en œuvre les fonctionnalités prévues, de mener plusieurs tests.

- **cmake** : C'est un système de build qui simplifie la création des makefiles, indispensables pour la compilation. Le fichier CMakeLists.txt contient les règles de compilation du code source. En cas d'ajout de bibliothèques ou de modifications du code, ce fichier doit être ajusté pour garantir une compilation correcte.

- **make** : Cette commande est utilisée pour compiler le code source dans le fichier CMakeLists.txt. Elle assure la génération des exécutables à partir des fichiers source.

- **sudo pyserial-miniterm** : Cette commande est utilisée pour afficher les résultats et les messages émis par le microcontrôleur. Elle facilite la surveillance en temps réel et permet de vérifier le bon fonctionnement du programme concerné sur le microcontrôleur.

Pour faciliter cette démarche, nous avons créé un guide détaillé expliquant les étapes à suivre pour la configuration et l'utilisation de CMake. Ce guide est inclus en référence [10], fournissant une référence utile pour assurer une compilation et un exécution du code source sur le microcontrôleur.

## 4.4 Résultats

### 4.4.1 Test de l'interface éléctronique avec le porteur du projet

Nous menons le test de l'interface électronique que nous avons mené en collaboration avec le porteur du projet. Ce test a eu lieu lors du séminaire à Lorient, les 12 et 13 juillet 2023. Ce séminaire a pour objectif de réunir les collaborateurs du projet et de faire le point après plusieurs années de travail. Avant le séminaire, nous avions achevé toutes les tâches nécessaires pour mettre en œuvre le prototype de la partie électronique. Toutes les parties du code ont été mises à jour sur Git pour qu'ils soient prêts à être utilisés pour faire un test avec le porteur du projet au séminaire. Cela nous a permis de nous préparer à réaliser un test approfondi avec le porteur du projet pendant le séminaire. Pendant le séminaire, nous avons procédé à l'assemblage de la partie électronique, en collaboration avec les autres parties, notamment la partie mécanique. Cette coopération étroite a été essentielle pour donner naissance au prototype final, comme illustre la photo 22.

Le prototype a été soumis à plusieurs tests avec le porteur du projet. Cette phase a permis d'identifier des améliorations indiquées en dessous-ci :

- **Utilisation d'électrodes Ottobock** : Nous avons opté pour l'utilisation d'électrodes Ottobock pour remplacer ceux d'Orthopus car elles sont plus fiables. Cela permet d'améliorer la qualité des signaux électriques.

- **Modification du Code** : Une modification du code a été effectuée afin de permettre un meilleur contrôle de la vitesse de la main en fonction des contractions musculaires, ainsi que l'arrêt automatique de la main lorsque la contraction musculaire se relâche.

- **Placement Optimal des Électrodes** : Nous avons également optimisé l'emplacement des électrodes pour garantir une meilleure précision et efficacité de la détection des signaux musculaires.

Malgré ces améliorations, il subsiste encore des défis à relever, en ce qui concerne la vitesse de la main, l'amplitude d'ouverture, et la course du pouce, qui doivent atteindre un minimum de 300 mm/sec.

Cette phase de test, lors du séminaire à Lorient a permis de valider une grande partie du travail et d'apporter des améliorations sur la prothèse de main-myoéléctrique. Elle a répondu aux besoins du porteur du projet, notamment en garantissant la discrétion de la carte électronique avec ses capteurs fiables et bien positionnés sur la peau. Ces avancées ont satisfait Nicolas Huchet, porteur du projet, qui peut désormais effectuer des actions telles que ramasser des

objets, prendre un café, ou serrer la main. Cette évolution lui permettra de réussir son objectif de participer à l'événement Cybathlon qui aura lieu en Suisse, en Octobre 2024.

Figure 22 – Prototype final et testé par le porteur du projet

En résumé, notre interface utilisateur a été soigneusement conçue pour offrir un contrôle intuitif et efficace du mouvement de la main. La conception ergonomique, la convivialité, l'intégration avec l'architecture logicielle et la prise en compte des retours utilisateur ont tous contribué à créer une interface utilisateur efficace et adaptée aux besoins du porteur du projet.

# 5 CONCLUSION

La partie éléctronique de la prothèse de main myoéléctrique est developpée avec succès au cours de stage. Le travail de programmation en langage C a été bien mené en élaborant sur la commande moteur incluant le blocage de courant et le controle de vitesse et de direction ainsi que la lecture d'éléctrodes à l'aide de bibliothèques. Les avancées réalisées de ce projet ont permis au porteur de projet, Nicolas Huchet, d'effectuer des tests concluants sur la nouvelle version de la prothèse. Ces tests ont été essentiels pour obtenir des retours pertinents en vue d'améliorer les fonctionnalités liées à l'ouverture et à la fermeture de la main. L'architecture matérielle a été élaborée de manière méthodique pour intégrer le dispositif d'ouverture et de fermeture de la main, ainsi que pour gérer le fonctionnement des électrodes et du moteur, tout en respectant le cahier des charges. Les ajustements nécessaires pour optimiser l'ouverture et la fermeture de la main ont été mis en place conformément aux recommandations émanant du porteur du projet, exprimées lors du séminaire à Lorient.

Toutefois, ce stage de quatrième année a pour objectif de consolider et de mettre en pratique mes connaissances acquises au cours de ma formation d'ingénieur en informatique et electronique des systèmes embarqués pour être formée assistante-ingénieure bureau d'études ou en électronique embarqué dans un cadre professionnel. En effet, j'ai amélioré mes connaissances en dévéloppement logiciel embarqué et en électronique. Ce stage m'a permis de me familiariser avec le milieu de la recherche en entreprise au sein de l'équipe compétente. J'ai eu l'opportunité d'approfondir mes compétences en développement logiciel embarqué au cours de ce stage. Cette expérience m'a permis de m'immerger dans le monde de la recherche en entreprise, au sein d'une équipe compétente. La liberté d'action et l'autonomie dont j'ai bénéficié ont renforcé ma responsabilité et ma capacité à résoudre des problèmes de manière autonome.

De plus, ce stage m'a offert la possibilité de réévaluer mon approche de l'organisation du travail, de perfectionner les connaissances en programmation et d'apprendre à m'adapter à l'environnement professionnel en tenant compte de ma surdité. D'un point de vue personnel, ces douze semaines passées à l'INRIA ont été une expérience enrichissante qui m'a permis de gagner en autonomie, de faire des découvertes passionnantes et de rencontrer des professionnels inspirants. En outre, ce stage a eu un impact positif à la fois sur mon développement professionnel et personnel, en m'apportant de nouvelles compétences dans le domaine de l'informatique et de la recherche.

# 6 Annexes

## 6.1 Configuration du drive moteur

| IN1 | IN2 | OUT1 | OUT2 | Sens de rotation |
|---|---|---|---|---|
| 0 | 0 | GND | GND | Pas de rotation (moteur bloqué) |
| 0 | 1 | GND | VBAT | Rotation dans le sens horaire |
| 1 | 0 | VBAT | GND | Rotation dans le sens anti-horaire |
| 1 | 1 | VBAT | VBAT | Pas de rotation (moteur bloqué) |

Figure 23 – Tableau d'entrées du drive moteur MP6551

## 6.2 Calculs de PWM

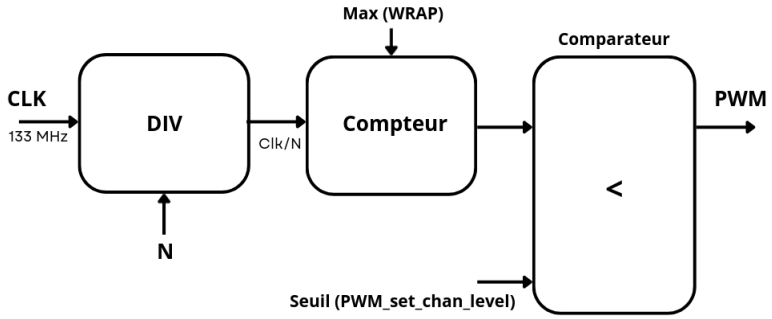


Figure 24 – Schéma du fonctionnement PWM

**Application numérique**

Avec $f_{\text{PWM}} = 30\,\text{kHz}$, $T_{\text{PWM}} = 30\,\mu\text{s}$, Clk $= 133\,\text{MHz}$ :

$$30\mu\text{s} = \frac{\text{Max} \times N}{\text{Clk}}$$

On en déduit : Max $\times\ N = 133 * 10^{-}3 * 30 * 10^{-}3 = 3990 = 4000$

Donc : Max $= 100$ et $N = 40$

## 6.3 Mode d'emploi de la configuration PWM

| PIN | Numéro de GPIO | Fonction | Slice | Channel |
|---|---|---|---|---|
| EN1 | 18 | PWM | 1 | A |
| EN2 | 24 | PWM | 2 | B |
| IN1 | 19 | WRAP | 1 | A |
| IN2 | 25 | WRAP | 2 | B |

Ces informations proviennent de la documentation MP6551 **??**(section 1.4.3 page 13)

## 6.4 Liste de fonctions PWM

### 6.4.1 Configuration de l'horloge PWM

```
...

#define MAX_PWM 100
pwm_config config = pwm_get_default_config();
pwm_config_set_clkdiv(&config, 40);
pwm_config_set_wrap(&config, MAX_PWM);

...
```

Figure 25 – Fonctions permettant la configuration PWM

### 6.4.2 Configuration des sorties EN1 et EN2

```
...

/*Configuration de sorties EN1 et EN2*/
gpio_set_function(EN1, GPIO_FUNC_PWM);
gpio_set_function(EN2, GPIO_FUNC_PWM);

//Configuration de la sortie OUT1
uint slice_1 =pwm_gpio_to_slice_num(EN1); //PWM slice connecté à EN1
pwm_init(slice_1,&config,true);

//Configuration de la sortie OUT2
uint slice_2=pwm_gpio_to_slice_num(EN2); //PWM slice connecté à EN2
pwm_init(slice_2,&config,true);
...
```

Figure 26 – Configuration de sorties EN1 et EN2

### 6.4.3 Configuration des sorties IN1 et IN2

```
...

/*INITIALISATION DE SORTIES*/
    gpio_init(IN1);
    gpio_init(IN2);


/*CONFIGURATION DE SORTIES  OUT1 ET OUT2*/
    gpio_set_dir(IN1, GPIO_OUT);
    gpio_set_dir(IN2, GPIO_OUT);
...
```

Figure 27 – Configuration de sorties IN1 et IN2

### 6.4.4 Fonction permettant la commande de la vitesse et de la direction du moteur

```
...
        printf("\nCommande de vitesse et du sens\n");

        if(direction == false)
        {
                //Changement de sens -> sens antihoraire
                gpio_put(IN1, 0);
                gpio_put(IN2, 1);
                pwm_set_gpio_level(EN1,MAX_PWM);
                pwm_set_gpio_level(EN2,speed);
                sleep_ms(2000);
        }
        else
        {

                //Changement de sens -> sens horaire
                gpio_put(IN1, 1);
                gpio_put(IN2, 0);
                pwm_set_gpio_level(EN2,MAX_PWM);
                pwm_set_gpio_level(EN1,speed);
                sleep_ms(2000);
        }
...
```

Figure 28 – Configuration du contrôle de moteur

### 6.4.5 Modification du fichier Makelist

```
...

add_library(motor motor.c)
target_link_libraries(motor hardware_pwm hardware_adc)

add_executable(test_motor test_motor.c)
target_link_libraries(test_motor motor pico_stdlib)
pico_enable_stdio_usb(test_motor 1)
pico_add_extra_outputs(test_motor)

```

Figure 29 – Extrait du fichier Makefile

## 6.5 Mode d'emploi du logiciel Pusa Slicer

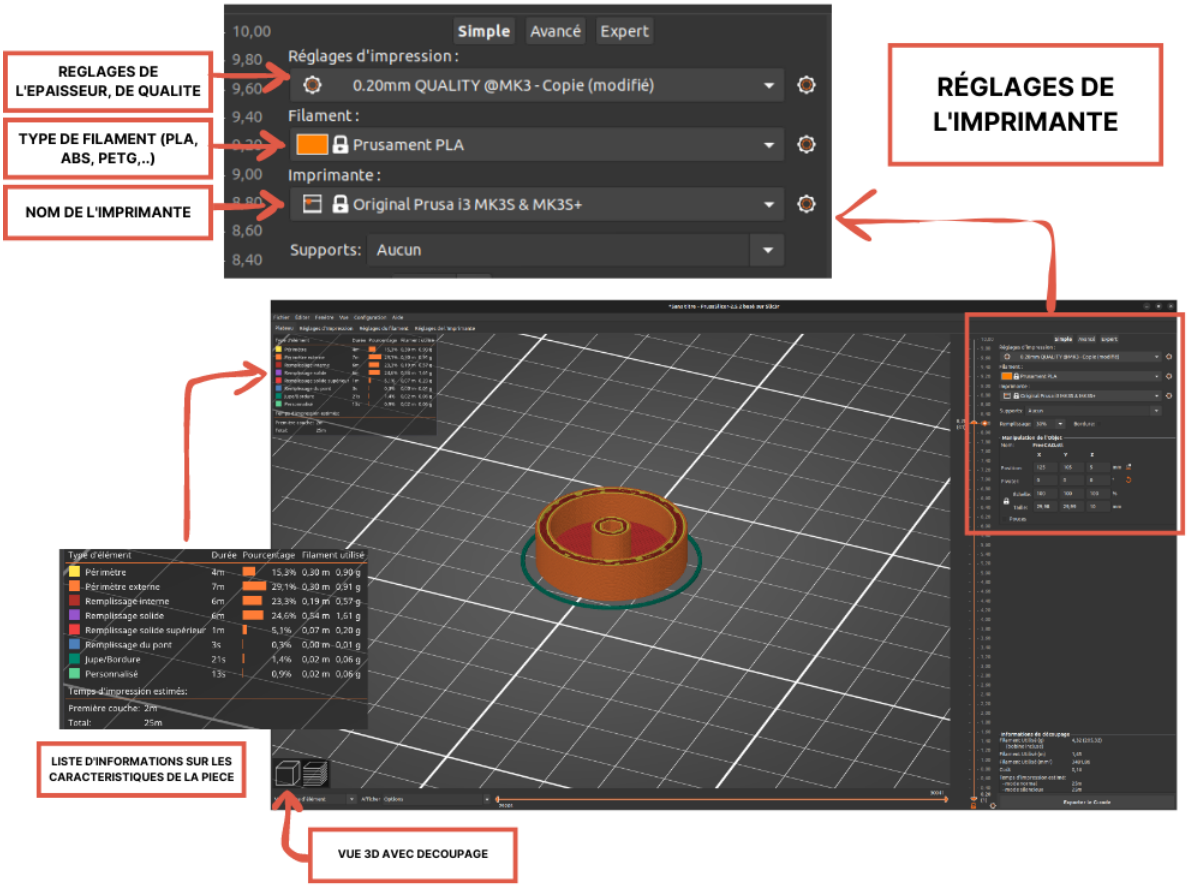


Figure 30 – Mode d'emploi du logiciel Pusa Slicer ode d'emploi du logiciel Pusa Slicer

1. Importer .STL sur Prusa Slicer
2. Faire les réglages de l'imprimante 3D (épaisseur, type de niveau de qualité, etc)
3. Séléctionner PLA pour le prototype du projet
4. Séléctionner Prusa i3 MK35MK35+
5. Vérifier le découpage en cliquant sur la surface de la pièce
6. Exporter le g-code
7. Importer le g-code dans l'interface de l'imprimante 3D

## 6.6 Etude sur le mouvement de main

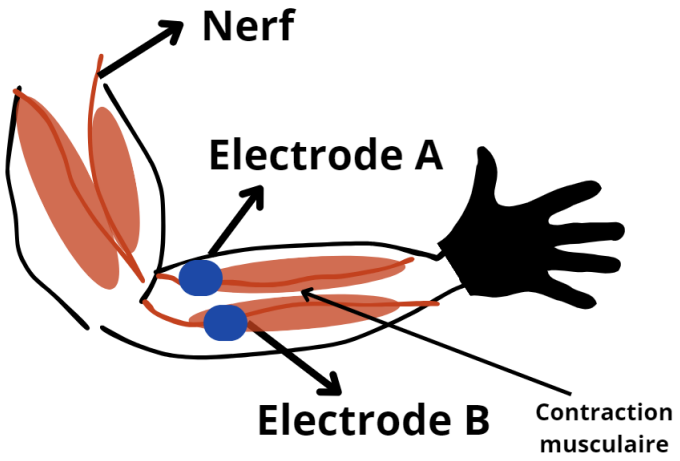


Figure 31 – Schéma du bras avant avec la contraction musculaire et les nerfs

## 6.7 Conception 3D finale pour le prototype

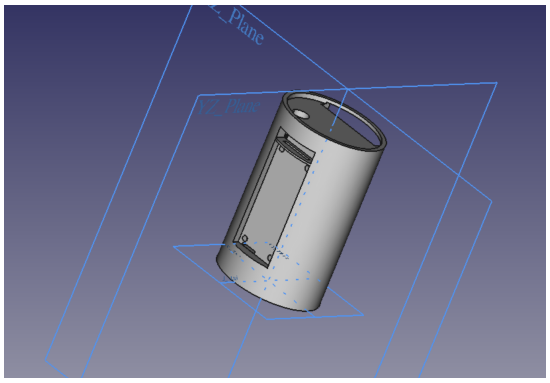

Figure 32 – Schéma de la pièce 3D pour le prototype sur FreeCAD

## 6.8 Fonctions permettant la configuration d'éléctrodes

### 6.8.1 Ajout de la bibliothèque pour le fichier electrode.c

```
add_executable(test_electrode test_electrode.c)
target_link_libraries(test_electrode electrode pico_stdlib hardware/adc.h)
pico_enable_stdio_usb(test_electrode 1)
pico_add_extra_outputs(test_electrode)

add_library(electrode electrode.c)
target_link_libraries(electrode hardware_adc)
```

Figure 33 – Extrait du code de la bibliothèque electrode.h

### 6.8.2 Lecture de valeurs ADC

```
//measure the output of ELECTRODE
void motor_read_electrode(float *_electrode)
{
        const float conv_val = 3.3f / (1 << 12);
        adc_select_input(2); //ADC input for switch
        *_electrode = adc_read();
        *_electrode=*_electrode * conv_val;
}
```

Figure 34 – Extrat du code de la fonction ADC

### 6.8.3 Convention code

- Documentation / commentaires en anglais
- Pas de franglais 😄 dans les noms de fonction
- Bien identer le code (4 espaces)
- Noms de fonction en minuscule avec _ pour séparer les mots, ex: `motor_init` et pas `MotorInit`
- Préfixer les noms de fonction par le nom de module, par exemple pour le module `mp2672`, le fonction d'initialisation s'appelle `mp2672_init`
- Dans les fichiers .c d'une bibliothèque, bien penser à mettre le mot clé `static` pour les fonctions qui sont internes à la bibliothèque

Figure 35 – Liste de contraintes pour la convention de code

## RÉFÉRENCES


[1] OMS, “Rapport mondial sur l’handicap.” https://www.unicef.fr/sites/default/files/userfiles/rapport_mondial_handicap_oms_2012.pdf. Accédé : 01-08-2023.

[2] “Fiche technique du drive moteur mp6551.” https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP6551GQB/document_id/9795/. Accédé : 01-08-2023.

[3] Raspberry, “Documentation rp2040.” https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Accédé : 01-08-2023.

[4] Raspberry, “Liste de fonctions raspberry pour l’embarqué.” https://www.raspberrypi.com/documentation/pico-sdk/hardware.html#hardware_gpio. Accédé : 01-08-2023.

[5] Raspberry, “Liste détaillée de fonctions pwm.” https://www.raspberrypi.com/documentation/pico-sdk/rp2__common_2hardware__pwm_2include_2hardware_2pwm_8h.html. Accédé : 02-08-2023.

[6] G. THIBAUDAT, “Mesures de courants.” https://docs.google.com/spreadsheets/d/1nvVwh06k2fwCLOKClJQD3I-qHsBNTEPclkCydlKT0go/edit?pli=1#gid=605308550. Accédé : 09-08-2023.

[7] C. BRAILLON, “Schéma éléctrique de la carte uc.” https://drive.google.com/file/d/1yh8xHI2pF5hInUXLceHT1uxT_GWPCtwD/view?usp=drive_link. Accédé : 09-08-2023.

[8] C. B. et Gaelle THIBAUDAT, “Dépôt de code source et ses fonctionnalités.” https://gitlab.inria.fr/humanlab-inria/bionico/. Accédé : 10-08-2023.

[9] C. B. et Gaelle THIBAUDAT, “Guide sur l’utilisation de git.” https://drive.google.com/file/d/1v4v6lpmSpprgjra_hiDs6kI2sx7lthRw/view?usp=sharing. Accédé : 10-08-2023.

[10] C. B. et Gaelle THIBAUDAT, “Guide sur l’utilisation de commande cmake.” https://drive.google.com/file/d/19LwVeXieJ54sp6-4zMZVYe-HAohvvs3Y/view?usp=drive_link. Accédé : 10-08-2023.

Étudiant : Gaëlle THIBAUDAT Année d'étude dans la spécialité : 4ème

Entreprise : INRIA
Adresse complète : 655 Av. de l'Europe, 38330 Montbonnot-Saint-Martin
Téléphone : +33 04 76 61 52 00

Responsable administratif :
Téléphone :
Courriel :

Tuteur de stage : Christophe BRAILLON
Téléphone :
Courriel : christophe.braillon@inria.fr

Enseignant-référent : EON David
Téléphone : +33 04 76 88 10 79
Courriel : david.eon@univ-grenoble-alpes.fr

Titre : ETUDE SUR LA PROTHÈSE DE MAIN MYO-ÉLÉCTRIQUE

**Résumé :**

Dans le cadre de la quatrième année IESE, j'effectue le stage de douze semaines dans l'équipe SED à INRIA Grenoble du 8 Mai au 28 juillet 2023.

Ce rapport présente le sujet de stage, qui m'a été proposé par Christophe BRAILLON, ingénieur en bureau d'études, qui vise à concevoir une prothèse de main myoéléctrique à faible coût et en opensource, en collaboration avec le porteur du projet, Nicolas Huchet, co-fondateur de MHK. L'objectif principal de ce stage est de se concentrer sur la partie électronique de cette prothèse myoélectrique, en élaborant son architecture matérielle et logicielle comprenant une commande moteur et la lecture des électrodes.

Par mes objectifs définis, j'ai plongé dans le domaine de développement logiciel et embarqué en mettant en oeuvre les missions nécessaires pour répondre au cahier de charges. Du point de vue pratique, j'ai découvert l'outil de l'imprimante 3D pour la conception 3D. J'ai conçu la commande moteur et le dispostif de lecture d'éléctrodes en programmant le microcontroleur uC à l'aide de la convention code et d'outils acquis pendant mon stage. Les tests que j'ai menés de manière automne sont été itératifs pour obtenir de résultats précis et fiables.

**Abstract :**

As part of my fourth year at IESE, I am doing a twelve-week internship at INRIA Grenoble from May 8 to July 28, 2023.

This report presents the subject of my internship, proposed to me by Christophe BRAILLON, a design office engineer, which aims to design a low-cost, opensource myoelectric hand prosthesis, in collaboration with the project leader, Nicolas Huchet, co-founder of MHK. The main objective of this internship is to focus on the electronic part of this myoelectric prosthesis, by developing its hardware and software architecture, including motor control and electrode reading.

Through my defined objectives, I discovered the field of software and embedded development, implementing the tasks required to meet the specifications. From a practical point of view, I discovered the tool of the 3D printer for 3D design. I designed the motor control and the electrode reading device by programming the uC microcontroller using the code convention and tools acquired during my internship. The tests I made were iterative to obtain accurate and reliable results.