

Au513 : Projet Prothèse de Main par Machine Learning



Projet de Réalisation de l'entraînement d'un réseau neuronal à destination d'une prothèse de main à quatre doigts, à partir de données simples d'un capteur. Utilisation d'Arduino et Python, avec la bibliothèque Tensorflow pour le Machine Learning

Table des matières

1-	Contexte	3
2-	Moyen pour la mise en œuvre.....	3
3-	Description du projet pas à pas	5
	Etape préliminaire : logiciels	5
	Etape 1 : Installation	6
	Etape 2 : Code Arduino	9
	Etape 3 : Code Python de récolte des données entraînement.....	9
	Etape 3.5 : Débruitage du signal sEMG obtenu	10
	Etape 4 : Le machine Learning et l'entraînement.....	12
	Etape 5 : Machine Learning embarqué, un sujet d'études	16
4-	Devis et cout de production	16
5-	Conclusion	18

1- Contexte

Après une étude¹ rapide des cas d'amputation en France, notre pays compte près de 37 400 français amputés. Parmi les amputés des membres supérieurs (accident de la route, accident du travail ou accident vasculaire), 70% sont amputés au niveau de l'avant-bras.

En quelques chiffres, 61% de ces opérations sont d'origine traumatique et sont un véritable handicap au quotidien, en plus d'être une discrimination certaine au travail, pour ces gens déjà vivement affectés par une opération lourde.

On place en France entre 12 000 et 15 000 prothèses par an, la plupart simples et légères, à destination des membres inférieurs. Pour ce qui est des membres supérieurs, malheureusement, recouvrir la motricité du membre perdu est une tâche bien plus difficile. En effet, les prothèses de main ou de bras sont bien souvent à but purement esthétique (membre en silicone imitant une peau organique), ou très simple (crochet ou pince).

Pour une prothèse motorisée, il faut en général compter dans les 4000€ pour une prothèse myoélectrique².

Lors de ce projet, nous allons étudier la faisabilité d'une prothèse de main avec des capteurs bas coût mais avec le Machine Learning appuyant le traitement de la donnée. Nous verrons le coût final estimé de notre prothèse si elle est probante.

Etude prothèse supérieure :

https://www.larenaissancesanitaire.fr/pdf/pub_scientifiques/protheses_membres_superieurs.pdf

2- Moyen pour la mise en œuvre

Pour réaliser ce projet, nous avons besoin de matériel et d'une base logicielle afin de pouvoir le mettre en œuvre correctement. Dans un premier temps, nous allons détailler les moyens physiques nécessaires.

Il est nécessaire tout d'abord de pouvoir récupérer les informations concernant l'électromagnétisme de la zone du corps étudiée. Dans notre projet, nous nous intéressons au bras. Il existe alors deux moyens de récupérer ces données : *Needle EMG* et *Surface EMG*

¹ <http://uf-mi.u-bordeaux.fr/ter-2016/mazouffre-figus/quelques-chiffres/statistiques/#:~:text=%20Il%20y%20a%2037%20400,%2C%2069%25%20sont%20des%20hommes>

² <http://uf-mi.u-bordeaux.fr/ter-2016/mazouffre-figus/quelques-chiffres/prix/#:~:text=%20Les%20gammes%20de%20prix%20des,la%20prothèse%20%3A%2026%20469€.&text=Emboîture%20sous%20coude%20de%20la%20prothèse%20myoélectrique%20%3A%203810€>

(sEMG). « EMG » signifie électromyographie. La première est une méthode invasive et coûteuse, et c'est pourquoi nous nous sommes intéressés principalement à la méthode « sEMG », qui utilise trois électrodes placées sur le bras pour obtenir les signaux électromagnétiques, notamment lorsqu'on effectue un mouvement tel que bouger un doigt.

Notre capteur envoie des données brutes que notre ordinateur ne peut traiter en continu. L'utilisation d'un microcontrôleur adapté permettant de trier et de sélectionner les données reçues est donc indispensable. Notre programme utilisant le *machine learning*, la puissance de calcul nécessaire ne peut être effectuée sur une carte mère de type *Arduino* ou *Raspberry* ; le microcontrôleur ne peut donc pas se suffire à lui-même avec les moyens dont nous disposons. Néanmoins, il existe un hardware qui permet de régler ce soucis, mais nous ne l'étudierons pas dans ce projet, faute de temps et de moyens. A notre échelle, nous utiliserons une carte Feather de la marque *Adafruit*. Un *adapter USB* spécifique pour relier la carte mère à l'ordinateur est nécessaire pour protéger celle-ci de la tension délivrée par l'ordinateur.

Pour résumer la partie physique de notre projet, il est donc nécessaire de posséder des électrodes, un capteur, un microcontrôleur ainsi que le câble pour relier ce microcontrôleur à un ordinateur. On considérera que l'ordinateur est déjà acquis. Le matériel proposé est listé en partie 4 : *Devis et coûts de production*.

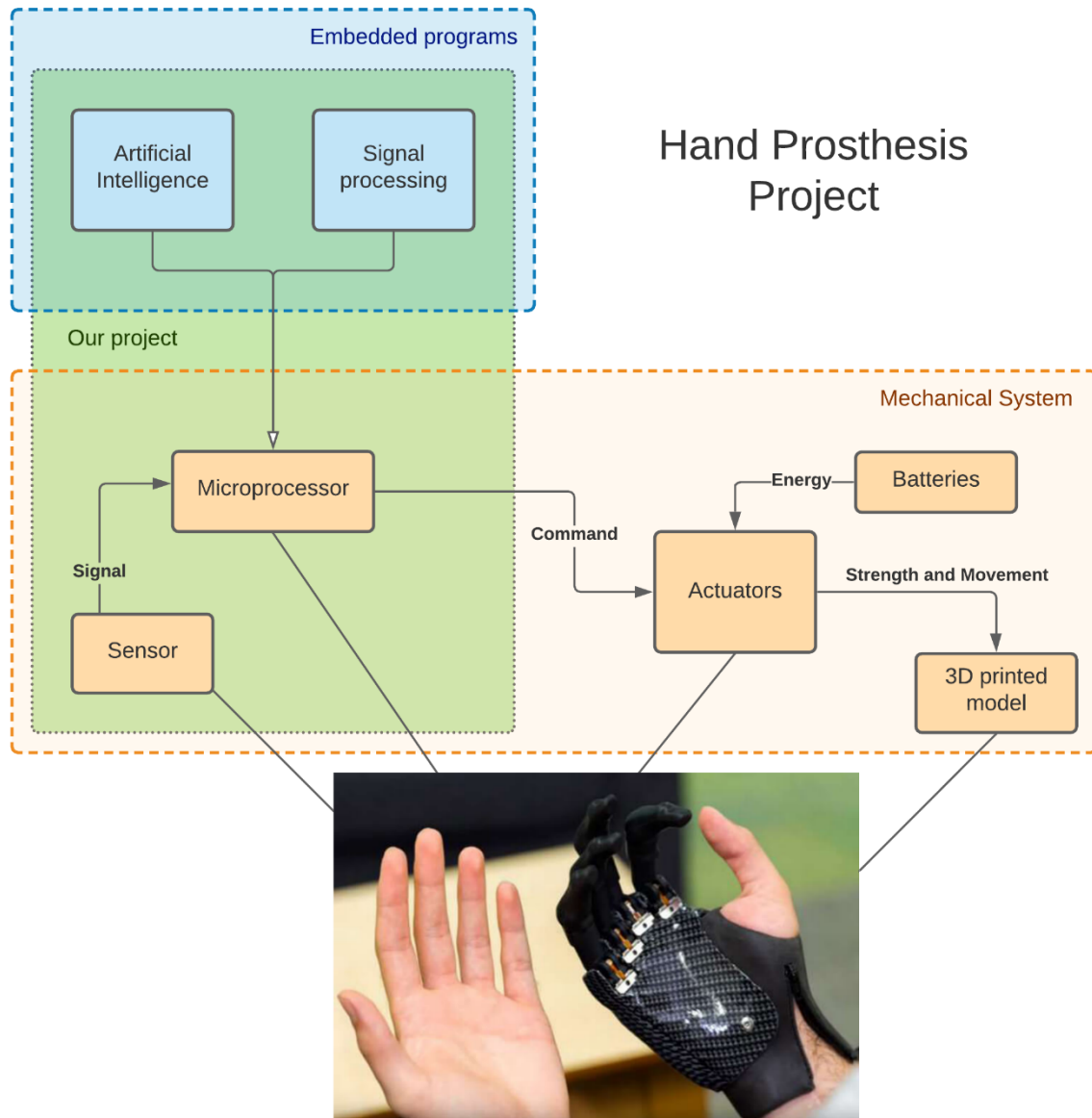
Dans un second temps, il est maintenant bon de s'intéresser à la partie informatique de notre projet, et de détailler ce que nous allons utiliser.

Il vient en premier la question de quel langage utiliser dans le cadre de notre projet. Du fait de notre recherche de matériel, nous avons préféré nous tourner vers une carte mère de type de programmation *Arduino*, de marque *Adafruit*.

Pour la partie sur le machine learning, deux langages principaux se distinguent pour pouvoir créer facilement un réseau neuronal : le *C++* et le *Python*. Du fait de nos connaissances personnelles et de l'accès à la librairie *Tensorflow* qui correspond exactement à nos souhaits, nous avons décidé d'opter pour le second langage de programmation.

3- Description du projet pas à pas

Lien GitHub : <https://github.com/Rodjohans/ManoRobo>



Etape préliminaire : logiciels

Sont à disposition sur GitHub les installateurs des différents logiciels. On y retrouve :

- Anaconda, contenant le logiciel Spyder permettant de manipuler les codes python. Il existe des méthodes pour automatiser ces codes (<https://datatofish.com/executable-pyinstaller/>) mais par manque de temps nous ne pouvons pas le proposer ici.
- Arduino, permettant d'ajouter un code sur le microcontrôleur. La version 1.6 est suffisamment stable.

- Le driver de la carte Adafruit dans le cas de notre projet, où nous travaillons sur ce type de cartes.

Cette partie du projet se compose de trois axes principaux :

- Installation
- Acquisition des données
- Construction du réseau neuronal

Etape 1 : Installation

1) Librairies

Pour que les codes puissent s'exécuter, il faut installer les librairies Serial et tensorflow.

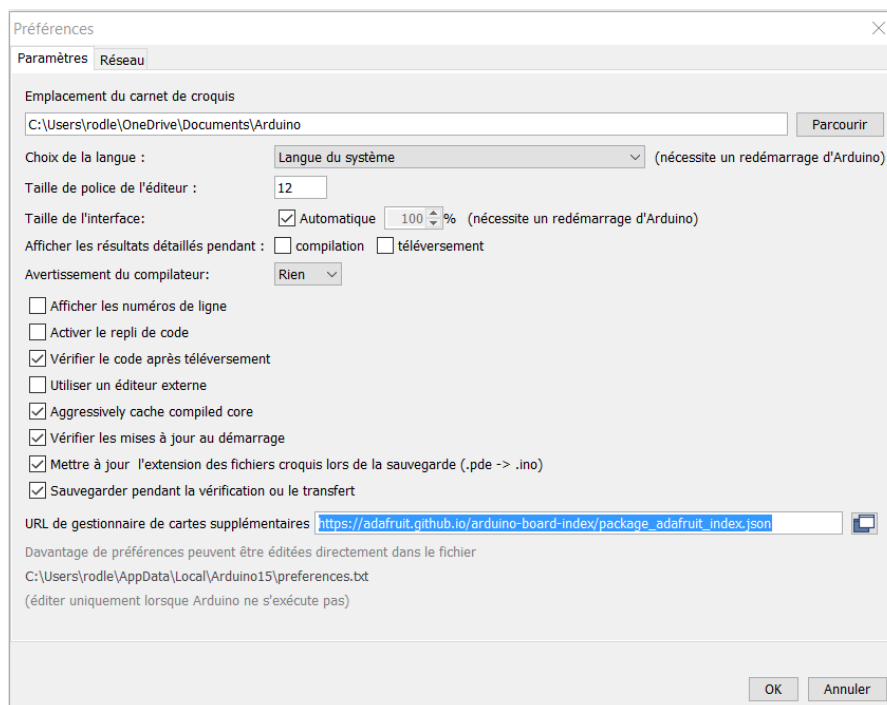
Serial : <https://pyserial.readthedocs.io/en/latest/pyserial.html>

Tensorflow : <https://www.tensorflow.org/install?hl=fr>


2) Drivers Adafruit

Après avoir installé Arduino et le driver Adafruit, ouvrez le logiciel Arduino et allez dans l'onglet Fichier/Préférences. Dans la section "URL de gestionnaire de cartes supplémentaires", insérez le lien :

https://adafruit.github.io/arduino-board-index/package_adafruit_index.json



Puis allez dans l'onglet Outils/Type de carte/Gestionnaire de carte. Tapez "Adafruit" dans le moteur de recherche et téléchargez les gestionnaires suivants :

 Gestionnaire de carte

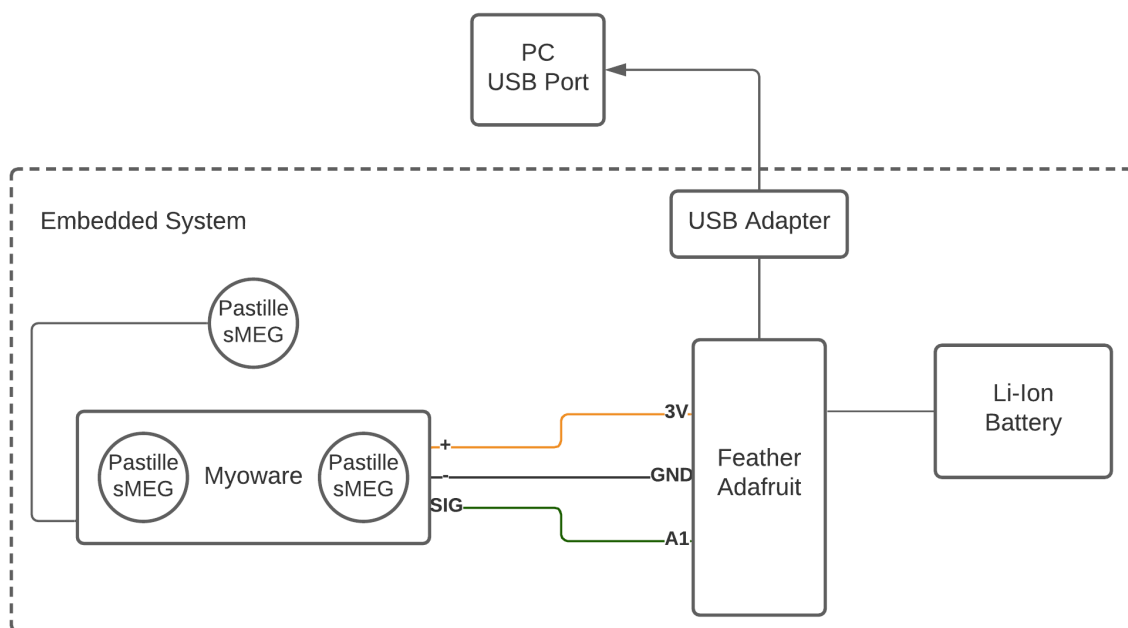
Type Tout ▼ Adafruit

Arduino AVR Boards by Arduino version 1.8.3 INSTALLED
 Cartes incluses dans ce paquet:
 Arduino Yún, Arduino Uno, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATmegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
[Online help](#)
[More info](#)

Arduino SAMD Boards (32-bits ARM Cortex-M0+) by Arduino version 1.8.9 INSTALLED
 Cartes incluses dans ce paquet:
 Arduino MKR WiFi 1010, Arduino Zero, Arduino MKR1000, Arduino MKRZERO, Arduino MKR FOX 1200, Arduino MKR WAN 1300, Arduino MKR WAN 1310, Arduino MKR GSM 1400, Arduino MKR NB 1500, Arduino MKR Vidor 4000, Arduino Nano 33 IoT, Arduino M0 Pro, Arduino M0, Arduino Tian, Adafruit Circuit Playground Express.
[Online help](#)
[More info](#)

Adafruit SAMD Boards by Adafruit version 1.6.4 INSTALLED
 Cartes incluses dans ce paquet:
 Adafruit Feather M0, Adafruit Feather M0 Express, Adafruit Metro M0 Express, Adafruit Circuit Playground Express, Adafruit Gemma M0, Adafruit Trinket M0, Adafruit ItsyBitsy M0, Adafruit pIRkey M0, Adafruit Metro M4, Adafruit Grand Central M4, Adafruit ItsyBitsy M4, Adafruit Feather M4 Express, Adafruit Hallowing M0, Adafruit NeoTrellis M4, Adafruit PyPortal M4, Adafruit PyBadge M4, Adafruit Metro M4 AirLift, Adafruit Matrix Portal M4, Adafruit BLM Badge, Adafruit QT Py.
[Online help](#)
[More info](#)

3) Branchement du hardware



Commencez par connecter la carte Adafruit à l'adaptateur USB, puis l'adapter à l'ordinateur. Ouvrez le logiciel Arduino et allez dans l'onglet Outils. Dans Types de cartes, choisissez la carte correspondant au microcontrôleur que vous utilisez. Il est possible que votre ordinateur détermine automatiquement ce choix. Assurez-vous qu'il s'agisse bien du bon type de carte (Adafruit et non Arduino). Dans l'onglet Port, sélectionnez votre carte et **retenez le numéro de port correspondant**.

Ouvrez le fichier *DataCollection.ino* dans Arduino. Vérifiez le code à l'aide des boutons en haut à gauche. Une barre de chargement apparaît en bas à droite de l'écran. Une fois le chargement complété et la barre disparue, s'il n'y a pas d'erreurs téléversez le code sur la carte Adafruit. Vous pouvez vérifier que le microcontrôleur fonctionne en ouvrant le moniteur série dans l'onglet Outils. Si une liste de nombres défile, alors la carte fonctionne.

Soudures :

Fermez Arduino puis débranchez la carte de l'ordinateur. Débranchez la batterie si vous l'aviez branchée à la Feather. Soudez des pins aux sorties A1, 3V et GND en vous assurant de ne pas déborder sur les autres branchements. Sur le capteur Myoware, soudez des pins aux ports "+", "-" et "SIG".

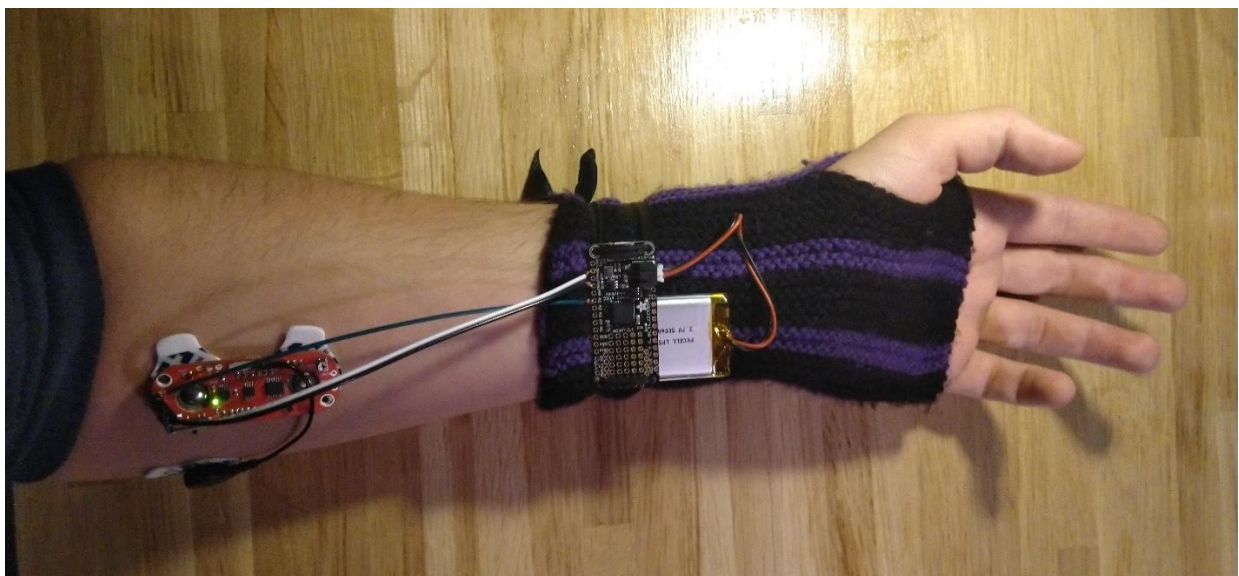
Branchez le pin "+" du Myoware au pin "3V" de l'Adafruit, le "-" au "GND", et le "SIG" au "A1".

Dans la version embarquée du dispositif, les différents actionneurs pourront être branchés aux autres ports de la carte Adafruit, et une batterie pourra être branchée directement à la carte. Certaines cartes sont livrées avec une batterie embarquée. Ce dispositif est léger (environ 1 g) et peu volumineux, ce qui le rend très confortable à porter au poignet par exemple.

Choisissez un lieu d'installation des pastilles sur l'avant-bras, de préférence à un endroit où les contractions se font sentir, et installez les pastilles du Myoware sur votre avant-bras. Différents sites expliquent quelles précautions peuvent être prises.

Pour l'installation des équipements, nous recommandons des isolants en tissu, comme un bracelet de force ou une mitaine par exemple.

Une fois le dispositif installé sur votre avant-bras, branchez l'adaptateur USB à l'ordinateur. Assurez-vous qu'aucun câble ou pin ne soit en contact avec votre peau. Vérifiez que le numéro de port dans le logiciel Arduino n'ait pas changé. Il est préférable de lancer Arduino après le branchement de la carte à l'ordinateur, même si ce n'est pas nécessaire.



Etape 2 : Code Arduino

A présent que notre branchement est effectué et que la partie physique est donc entièrement réalisée, le cheminement des données se réalise par l'algorithme téléversé plus tôt. En réalité, le microcontrôleur, une fois alimenté par une batterie ou par l'ordinateur, effectue le code en boucle. Nous utilisons un code python pour utiliser ces données. Pour des raisons pratiques, nous avons décidé de nous concentrer uniquement sur le mouvement des doigts, à l'exception du pouce, puisqu'il dispose d'une possibilité de rotation rendant son système plus complexe du fait de nombreux états possibles. Des systèmes mécaniques passifs peuvent être mis en place pour lier le mouvement latéral des doigts à la rotation du poignet :



Etape 3 : Code Python de récolte des données entraînement

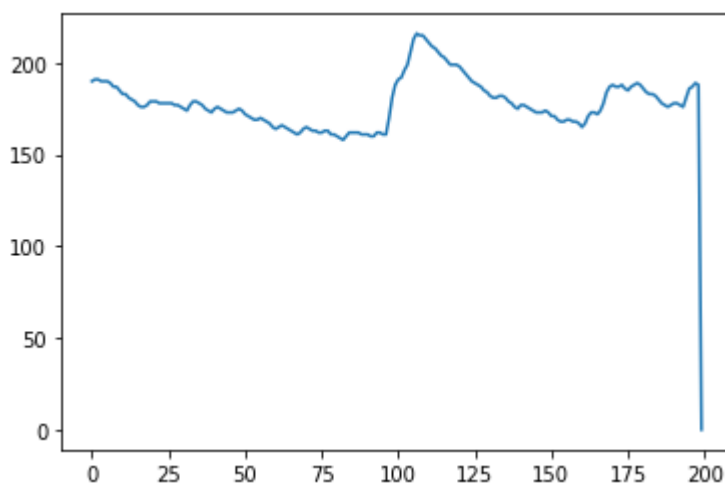
Une fois les données envoyées par l'Adafruit, il faut à présent voir comment l'ordinateur les reçoit et les traite. Pour cela, nous avons mis au point un programme python qui permet de ranger dans un document CSV les différentes données reçues. C'est ce qui nous permet de construire une première base de données destinée à l'entraînement. Notre base de données est composée de 50 expériences différentes pour l'entraînement, et une seconde base de données destinée au test, composé d'une dizaine de signaux.

Pour construire la base de données, ouvrez à l'aide de Spyder (Anaconda) le code *ConstructionDataBase.py*. Assurez-vous qu'à la ligne 21 le nom du port 'COM_' corresponde bien au numéro de port indiqué dans le logiciel Arduino. Fermez Arduino s'il est ouvert. Lancez le programme (flèche verte simple en haut) et contractez un ou plusieurs doigts **une** fois. Lorsque cela vous sera demandé, entrez les doigts correspondants. Si un doigt a été contracté pendant l'essai avant que la question soit posée, notez 1. Pour ceux étant restés décontractés, entrez 0. Si vous n'êtes pas sûrs, contentez-vous de faire un retour à la ligne ou d'entrer du texte, l'essai ne sera pas ajouté à la base de données. Si vous souhaitez recommencer, vous pouvez taper "dataBase.close()" dans la zone de texte à droite et vous rendre à l'endroit où

vous avez enregistré le code python. Supprimez ou modifiez à l'aide d'Excel le fichier DataBase.csv.

Etape 3.5 : Débruitage du signal sEMG obtenu

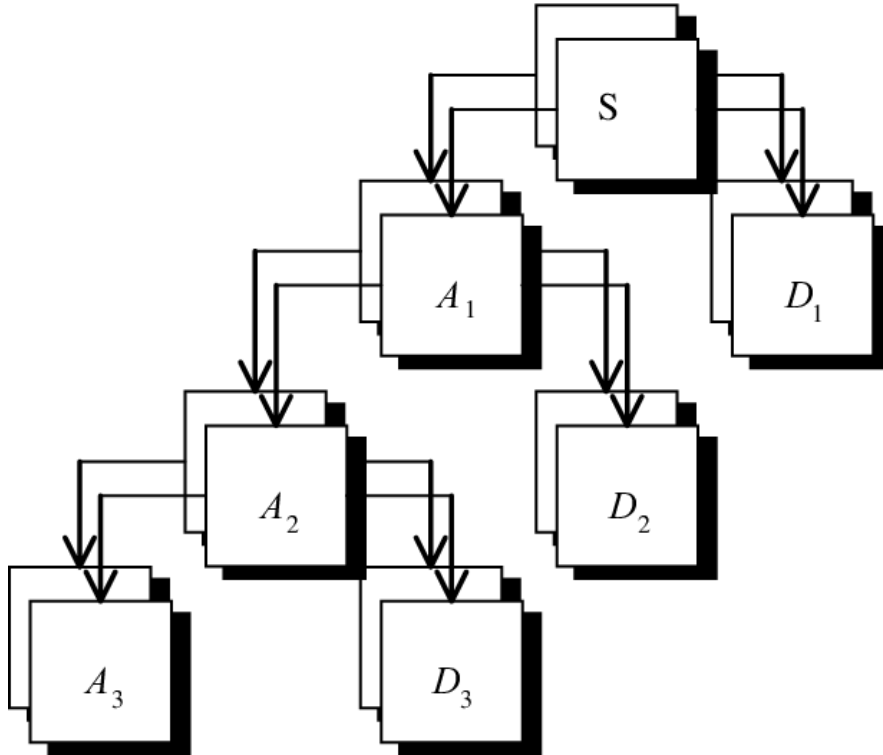
Toutefois, les signaux reçus par le microcontrôleur possèdent un bruit important. Il y a toujours une variation du courant électromagnétique sur notre peau, et certains micromouvements, indépendants de notre volonté, peuvent empêcher notre algorithme de bien différencier les fréquences de mouvement d'un doigt en particulier. Les imprécisions de branchement (soudure...) peuvent aussi entraîner un bruit conséquent. Le graphique ci-contre montre le second signal de notre base de données :



On voit donc qu'il existe de nombreuses variations électromagnétiques et nous ne voulons pas que ces variations impactent notre programme de machine learning. En effet, elles peuvent cacher la tendance du signal. Nous ne pouvons donc pas nous contenter d'effectuer un filtre passe-haut, puisqu'un tel dispositif pourrait enlever des informations cruciales. Pour réduire le bruit, nous avons donc décidé d'utiliser la méthode des Wavelets.

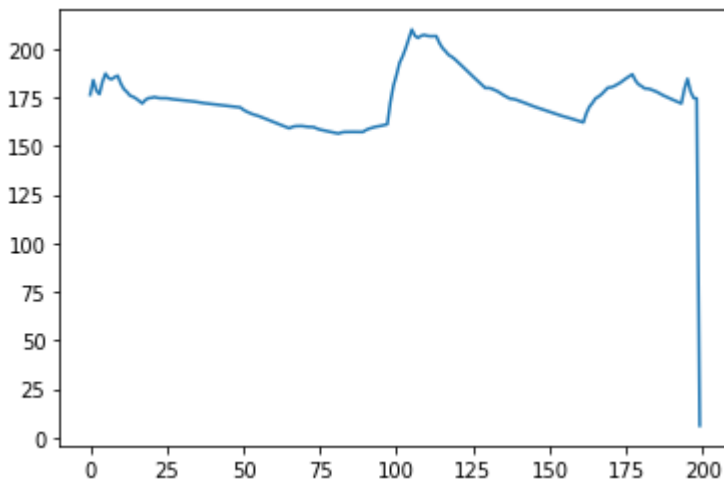
Il s'agit d'une méthode similaire à un celle de la transformée de Fourier, mais qui offre une meilleure solution face au principe d'incertitude d'Heisenberg. Ainsi, cette transformée offre à hautes fréquences une meilleure résolution temporelle, et à basses fréquences une meilleure résolution spatiale, ce qui est dans la plupart des cas le plus important. Comme pour les Short Time Fourier Transform, on utilise des fenêtres spécifiques pour bien isoler les fréquences souhaitées. Dans notre cas, nous avons utilisé une wavelet de Daubechies, qui est plus adaptée au débruitage. Puisque nous souhaitons obtenir le meilleur débruitage possible, nous allons faire une décomposition à plusieurs niveaux (3 exactement) dont le principe est expliqué dans le schéma ci-dessous. On applique la transformée en wavelet sur le nouveau A à chaque fois, qui regroupe les basses fréquences de notre signal S.

$$\begin{aligned}
 S &= A_1 + D_1 \\
 &= A_2 + D_2 + D_1 \\
 &= A_3 + D_3 + D_2 + D_1
 \end{aligned}$$



Source : https://www.researchgate.net/figure/Multilevel-wavelet-decomposition-of-a-signal-sdtTH_fig7_222706033 (Mettre en forme ça plus tard)

On passe ensuite les coefficients obtenus dans un filtre passe-bas, afin de filtrer correctement. On obtient ainsi un signal qui semble plus lisse et donc les microvariations ne devraient plus affecter le résultat final. Ici, le même signal que plus haut, après l'application des transformées et du filtre :



On voit bien que le signal est plus lisse mais que les tendances sont gardées, ce qui permet donc à notre algorithme de machine learning de faire plus attention aux fréquences qui ressortent. Il est important de savoir que nous avons rajouté cette étape assez tardivement dans notre projet et qu'il a permis d'améliorer légèrement nos résultats. Néanmoins, cette méthode, bien qu'elle offre des avantages, ne peut pas réduire certains bruits. En effet, des mouvements involontaires peuvent faire apparaître des fréquences sur le signal. Par exemple, lorsqu'on bouge l'auriculaire, l'annulaire avance partiellement également, ce qui cause un artefact qui peut faire penser à l'algorithme que l'annulaire a bougé entièrement.

Une piste pour corriger ce problème serait d'élargir notre base de données de test afin d'améliorer la distinction entre le mouvement de ces doigts, ou de ne pas prendre en compte certains mouvements involontaires.

Etape 4 : Le machine Learning et l'entraînement

Maintenant que nous avons les données dans deux bases différentes, nous pouvons enfin passer à la réalisation de notre algorithme de *machine learning*. Le but est de voir s'il est possible que l'ordinateur trouve une corrélation entre les signaux lorsqu'on bouge un certain doigt.

Le fonctionnement d'un algorithme de Machine Learning est le suivant : un ensemble de matrices avec n variables sont reliées entre elles et seront entraînées à reconnaître des patrons donnés dans un set de données appelé "set d'entraînement". Cette étape n'est rien de plus qu'une optimisation par descente de gradient sur un très grand nombre de paramètres. Un très grand entraînement, mené de manière pertinente³ permettra d'exploiter au mieux les données que nous avons créées auparavant.

Pour cela, nous allons, dans un premier temps, concevoir les différentes composantes du réseau de neurones. La première étape est la création de deux matrices : X_{train} la donnée récoltée et Y_{train} , le résultat escompté. Notre algorithme devra alors faire une optimisation sous contrainte de $f(X_{train}[n]) = Y_{train}[n]$, où n est une donnée quelconque du set, le tout en prenant en compte les résultats antérieurs à n . Le Machine Learning apparaît donc comme une évolution de l'optimisation mathématique sous contrainte :

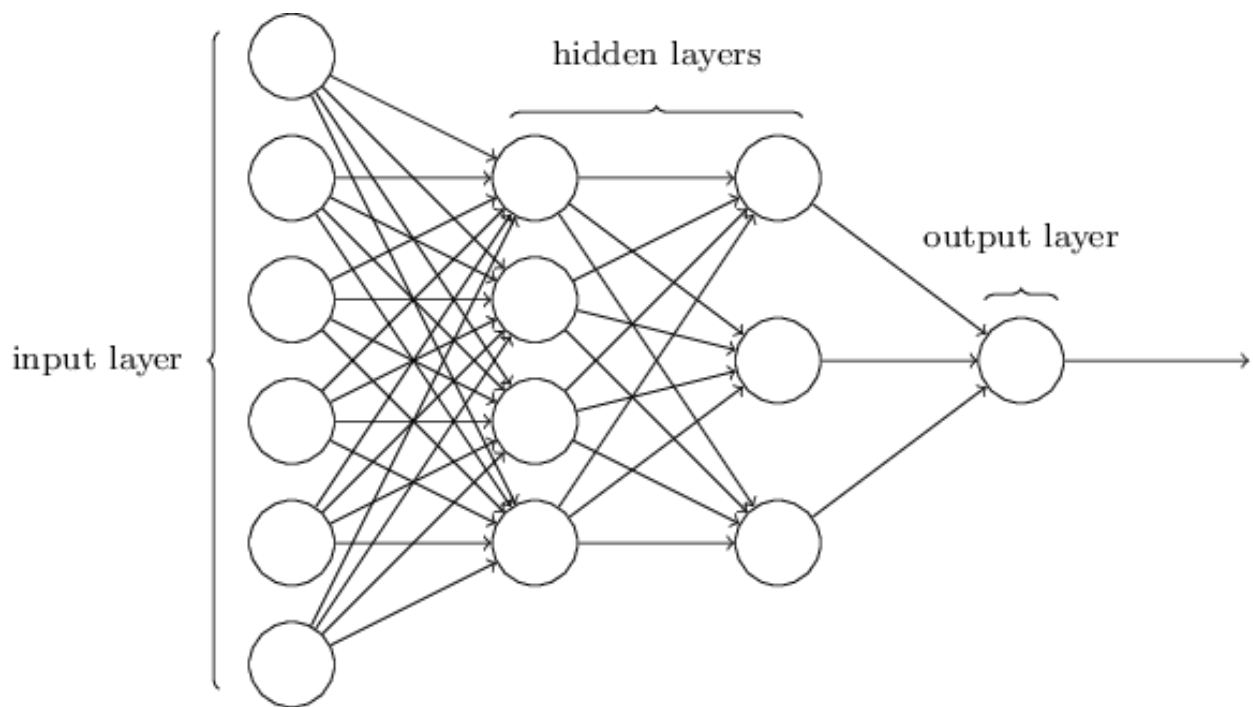
$$\begin{cases} f(X_{train}[n]) = Y_{train}[n] \\ \text{sachant les valeurs précédentes} \end{cases}$$

Une fois ces deux matrices créées, nous passons à l'architecture du réseau. Celui-ci étant adapté à notre problème, il ne possède pas de degrés de liberté sur les entrées et sorties : le

³ Eviter l'overfitting, comparer les méthodes de loss et le nombre d'époques sont autant de paramètres à prendre en compte pour espérer avoir un résultat cohérent.

nombre de neurones d'entrée seront le nombre de points de notre échantillon, et les neurones de sorties seront les activations (ou non) des doigts ou de la main entière. Dans notre cas, nous avons 200 neurones d'entrée et 1 à 4 neurones de sortie (4 pour l'activation indépendante des doigts ou un seul pour une activation globale de la main, ces modes sont modifiables à notre guise).









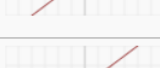
Illustrons par le schéma suivant :



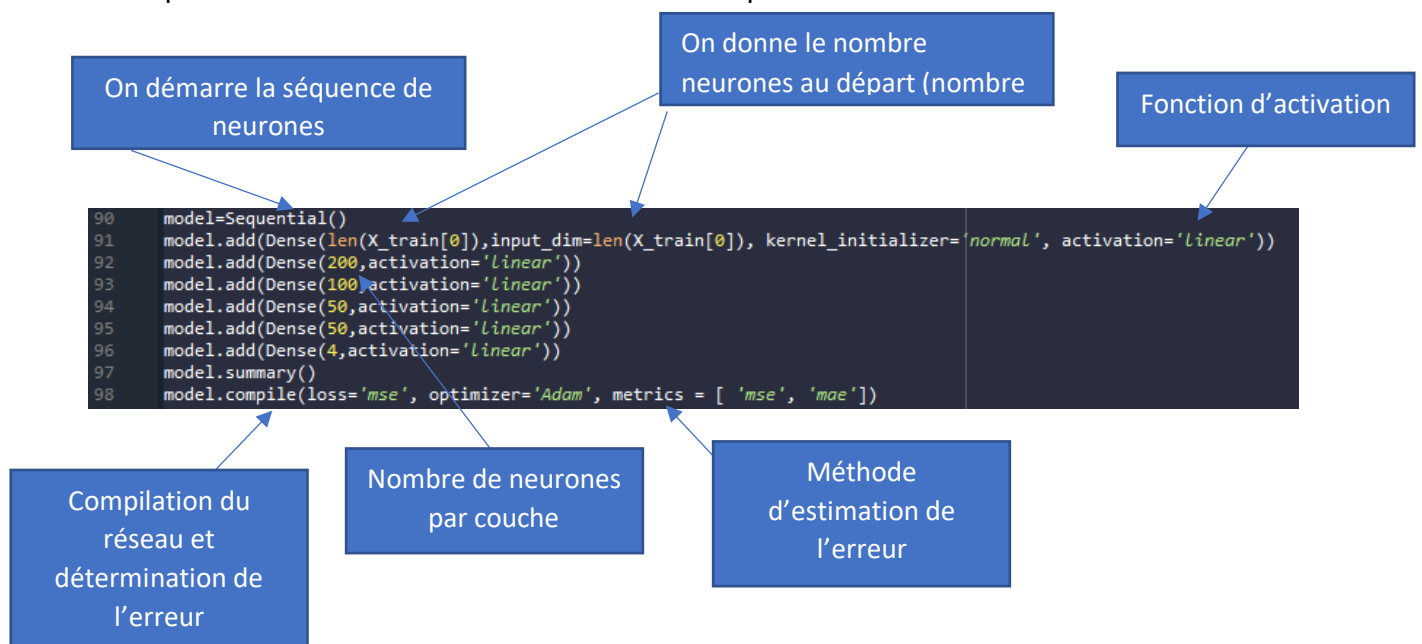
Chaque neurone d'entrée est donc un nombre entier dont la valeur sera connue de tous les neurones suivants. Cette valeur sera multipliée par un poids α , puis additionnée à un biais β . Chaque neurone i de la couche k étant relié à chaque neurone de la couche $k-1$, ils représentent eux-mêmes un nombre réel tel que :

$$Neurone_{[couche\ k]}^i = f(\sum \alpha_{k-1}^j * Neurone_{k-1}^j + \beta_{k-1}^j)$$

La fonction ci-dessus est appelée « *fonction d'activation* », dont les plus courantes sont les suivantes :

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) ^[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) ^[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Notre code fait alors l'inventaire du nombre de couches, le nombre de neurones par couches ainsi que les fonctions d'activations utilisées à chaque couche.



On lance alors l'algorithme de fitting du réseau en précisant le nombre d'itérations et on obtient alors un modèle que l'on peut sauvegarder !

```
history= model.fit(X_train,Y_train, epochs = 800, batch_size = 60, verbose=1, validation_split=0.2)
```

Pour vérifier la pertinence de nos paramètres, nous créons un nouveau set de données inédit que le réseau n'a jamais vu : c'est le set de test. Nous connaissons aussi sur ce réseau le résultat attendu, mais pas notre algorithme. L'objectif est de comparer les prédictions faites par l'ordinateur et la réalité des données.

Pour cela, nous avons créé une « matrice de la différence », avec nos propres codes : dans cette matrice, Python va nous donner le résultat échantillon par échantillon du réseau en comparaison avec ce qu'il aurait dû être.

- Un **0** est la marque d'aucune différence entre prédiction et réalité : *le réseau a rempli sa tâche*
- Un **1** est le signe d'une activation parasite : *le réseau a activé un doigt alors qu'il n'aurait pas dû*
- Un **-1** représente une occasion ratée : *le réseau n'a **pas** activé un doigt alors qu'il aurait dû*

Un exemple parlant est le suivant :

```
The model error
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0. -1.  0.]
 [ 0.  0.  0.  1.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [-1.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 1.  0. -1.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  1.  1.  1.]
 [ 0.  1.  0.  1.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [-1.  0.  0.  0.]
 [-1.  0.  0.  0.]
 [ 0. -1.  0. -1.]
 [ 0.  0. -1.  0.]
 [-1.  0.  1.  1.]
 [ 0. -1.  0.  1.]
 [ 0.  0.  0.  0.]
 [ 1.  0.  0. -1.]]
```

Sur la première ligne, le réseau fait un sans-faute !

Sur la troisième ligne, le réseau garde la main fermée alors que la commande est d'ouvrir l'annulaire.

En tout, on compte tout de même 12 « sans-faute » sur 20 au doigt près ! Si l'algorithme était redirigé vers une commande « main ouverte OU main fermée », notre score passerait de 12/20 à 15/20.

Ce score est sans aucun doute améliorable avec davantage de données à fournir à notre Machine Learning. Mais, manquant de temps, nous nous sommes contentés d'une centaine de relevés « fait-maisons » par Rodolphe.

On voit alors la puissance de notre réseau : malgré la qualité des capteurs et la faible quantité de données nous obtenons des résultats très encourageants !

En conclusion, nous sommes assez confiants de la viabilité de notre projet, il peut être fait avec plus de moyens, jusqu'au prototypage de la main par impression 3D !

Etape 5 : Machine Learning embarqué, un sujet d'études

Notre avancement s'arrête ici, mais il reste beaucoup à faire.

Il s'agit désormais d'embarquer le programme du réseau neuronal sur le microcontrôleur. Le nouveau code Arduino être organisé comme ceci :

- Acquisition des données
- Enregistrement des données pendant 2 secondes au même intervalle de temps que le code python
- Une fois l'enregistrement terminé, passage de l'échantillon dans le programme de réseau neuronal, et détermination des doigts à activer
- Activation des ports de sortie correspondant à la dernière combinaison de doigts reçue

Ainsi, la prothèse ne sera pas vraiment en temps réel. En augmentant le nombre de mesures par seconde et en réduisant l'expérience à 1 seconde, il est possible d'améliorer sa rapidité. Toutefois, il deviendra alors plus difficile de réaliser des expériences précises pour la construction de la base de données.

Quel que soit le choix réalisé pour l'acquisition des données, la seule chose dont il faut s'assurer est que toutes les expériences aient exactement la même taille, y compris l'acquisition finale embarquée sur la carte.

Les différents tests effectués nous montrent que notre algorithme de machine learning parvient à prédire correctement le mouvement dans % des cas, ce qui est un résultat prometteur mais qui nécessiterait une optimisation, telle que l'augmentation du nombre de capteurs, ainsi qu'une façon de réduire le bruit des différents signaux.

4- Devis et coût de production

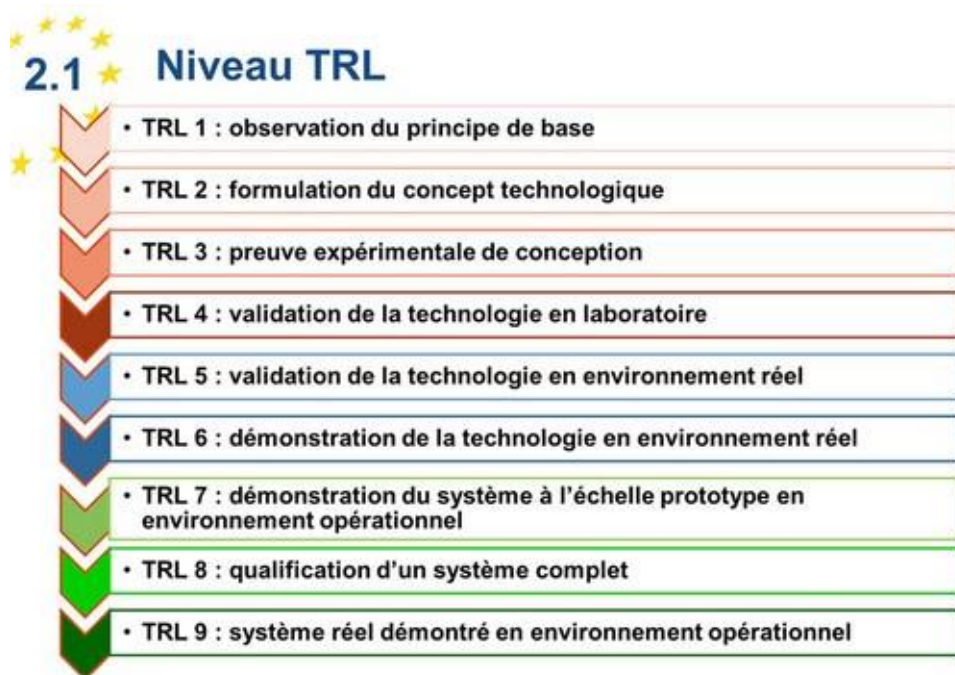
Afin d'éviter les éventuels frais de douane, il est préférable d'acheter ces équipements sur des sites français.

- MyoWare Muscle Sensor
 - ➔ Un capteur Electromagnetique pour récupérer les données de contraction des muscles
 - ➔ Prix : **40,80€** TTC hors livraison
 - ➔ Pourquoi celui-ci ? fiable et très documenté, il nous permettra de nous appuyer sur un ensemble de tuto et de discussion sur forum en cas de problème
 - ➔ Lien : <http://www.advancertechnologies.com/p/myoware.html>
- 3 EMG Electrodes
 - ➔ Réutilisables, ces électrodes viennent se clipser sur le capteur

- ➔ Pack de 6 à **4,95\$** TTC hors livraison
- ➔ Pourquoi celui-ci ? Compatible avec notre capteur, c'est celui utilisé dans la documentation du site constructeur du MyoWare
- ➔ Lien : <https://www.adafruit.com/product/2773>
- Adafruit Feather 32u4 Basic Proto (or any other Arduino-compatible that has analog input pins)
 - ➔ Microcontrôleur aidant le passage du capteur vers l'ordinateur. *Equivalent Arduino*
 - ➔ Prix : **19,95\$** TTC hors livraison
 - ➔ Pourquoi celui-ci ? marqué comme indispensable dans le manuel de la MyoWare
 - ➔ Lien : <https://www.adafruit.com/product/2771>
- Adafruit USB isolator
 - ➔ Relie le microcontrôleur à l'ordinateur pour l'acquisition des données dans le but de créer un réseau neuronal
 - ➔ Prix : **35,95\$** TTC hors livraison
 - ➔ Indispensable à l'acquisition de données
 - ➔ Lien : <https://www.adafruit.com/product/2107>
- USB mini cable
- USB micro cable
- Alimentation : batterie Lithium-Ion 500mAh

5- Conclusion

Par rapport aux niveaux TRL, utilisés dans l'industrie pour représenter l'avancement d'un projet, nous dirions que notre partie se situe entre TRL 3 et TRL 4 : nous avons prouvé que nous pouvions construire une base de données exploitable à partir d'un seul capteur Myoware.



La suite du projet consisterait à imaginer des méthodes permettant d'embarquer le neural network dans le microcontrôleur, et d'automatiser son fonctionnement si possible sans exploser les coûts en matériel supplémentaire. Il ne s'agit pas ici de créer un ordinateur de bord sur la main du patient.

Concernant la partie mécanique de la main, de nombreuses idées sont bonnes à noter. Des designs de mains imprimables en 3D existent, et peuvent être améliorés avec les connaissances acquises à l'IPSA. Nous pensons qu'un système passif actionné uniquement lorsqu'un doigt est à contracter est préférable. De nombreux systèmes de ressorts et engrenages sont aussi à l'étude dans ce domaine.

En conclusion, nous sommes entièrement satisfaits de ce projet : pour la première fois, nous nous sommes sentis ingénieurs sur un projet concret avec une application concrète. Ce projet a aussi été pour nous l'occasion d'aller plus loin que les disciplines étudiées à l'IPSA. Obtenir des résultats aussi encourageant dans un domaine que nous ne connaissions pas il y a de cela quelques mois, a été pour nous à la fois autant gratifiant qu'instructif. Faire de la robotique en dehors de l'aéronautique a été pour nous une expérience rafraichissante et nous a ouvert l'esprit sur la multitude des enjeux que nous pourrions soulever demain.