

---

# Robot Segway

---

Dossier de travail

by

Groupe 8 : "Les Recrutés"

**Membres**

DOL, Timothée  
INCACUTIPA PEREZ, Cristian  
ROUDEIX, Roméo  
WU, Kaiwei

**Encadrant(s)**

Florent Di Meglio

novembre 2023

# Abstract

Dans le cadre de ce projet, nous avons conçu et construit un segway. Afin d'assurer sa stabilité, nous avons automatisé le système en lui appliquant des contrôleurs. Pour ce faire, nous avons dû déterminer les équations modélisant le système et ses contrôleurs ; et nous avons dû utiliser le logiciel Arduino IDE pour implémenter les équations d'automatique et envoyer les commandes au robot.

La finalité du Segway est de réaliser deux épreuves : parvenir à rester en position haute pendant 30 secondes avec le moins de déplacements résiduels possible et monter une rampe de 30cm de large dont l'inclinaison peut varier entre  $0^\circ$  et  $28^\circ$

# 1 | Prototypage des pièces

## 1.1 Conception des roues

### 1.1.1 Première version

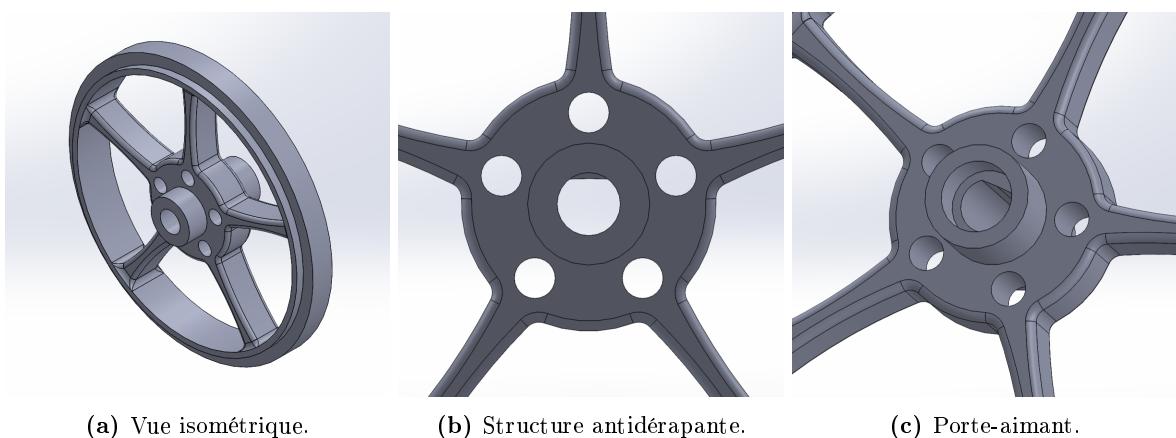
- La roue a le même diamètre et la même épaisseur que la roue proposée par le FabLab. Nous avons choisi cela pour pouvoir utiliser (dans un premier temps) les pneus proposés au Fablab.
- Au centre de la roue, il y a un trou servant à faire passer l'essieu. Une pièce (cale) a été ajoutée pour empêcher le glissement de la roue avec l'essieu.
- Les trous supplémentaires ont été rajoutés autour du trou central pour réduire la masse de la roue.

#### Problèmes avec la première version

- En raison de la précision de l'imprimante 3D, le trou central est de diamètre insuffisant pour pouvoir y faire rentrer l'arbre du moteur.
- Également, nous prévoyons d'améliorer la conception de la roue en plaçant un support permettant de positionner l'aimant à une bonne distance de l'arbre (contrainte de bon fonctionnement des encodeurs).

### 1.1.2 Deuxième version

- Les problèmes de la première version sont résolus.
- L'arbre moteur rentre dorénavant dans le trou central de la roue et la cale permet bien l'empêchement de tout glissement.
- C'est la nouvelle pièce destinée à fixer et positionner l'aimant est fonctionnelle.
- Le support de l'aimant est étendu jusqu'à toucher le encodeur, ce qui empêche la distance entre le encodeur et l'aimant de varier.



(a) Vue isométrique.

(b) Structure antidérapante.

(c) Porte-aimant.

FIGURE 1.1 – Modélisation des roues en 3D.

## 1.2 Conception du support moteur

### 1.2.1 Première version

- Il est proposé de faire un seul bloc pour éviter au maximum l'utilisation de colle ou de vis.
- L'avantage d'utiliser un seul bloc est également que la distance entre le codeur et l'aimant sera fixe et qu'aucun ajustement ne sera nécessaire.
- Pour le support du codeur, une cavité est prévue pour le montage en force du codeur afin d'éviter l'utilisation de vis.

#### Problèmes avec la première version

- L'encodeur ne rentre pas dans la cavité prévue.
- Il n'y a pas moyen de fixer le moteur et la roue sur le support.

### 1.2.2 Deuxième version

- La nouvelle conception du support de l'encodeur permet à l'encodeur de s'adapter parfaitement et d'être fixé par des pinces à pression.
- Des découpes permettent de fixer le moteur et la roue sur le support.
- Deux cavités ont été ajoutées à l'endroit où seront placés les pions de centrage.

#### Problèmes avec la deuxième version

- L'agrafe a des dimensions de 1 et 0,6 mm, ce qui en fait une pièce très fragile.

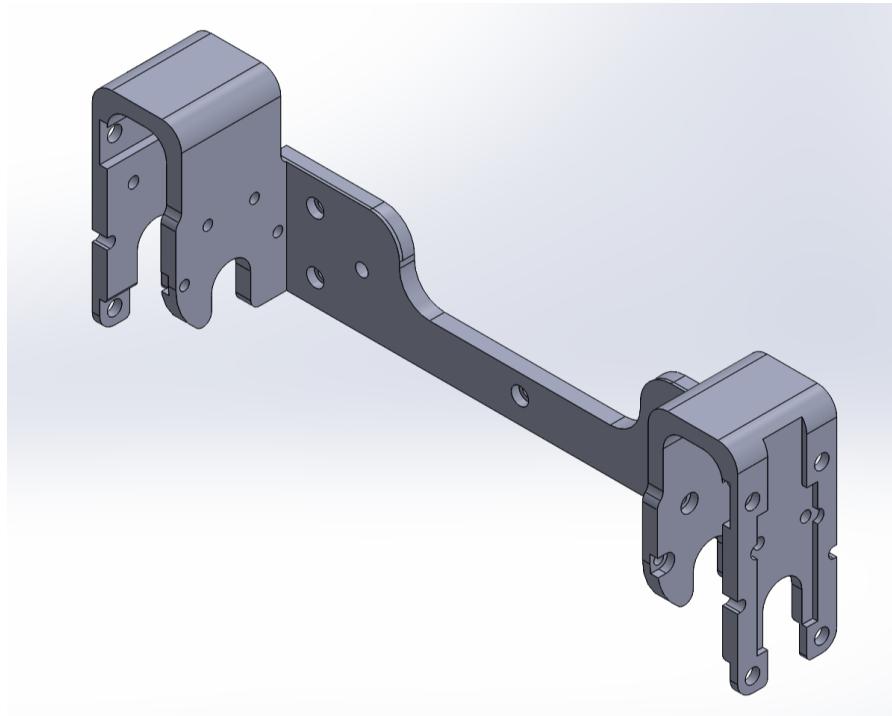


FIGURE 1.2 – Vue isométrique du support moteur.

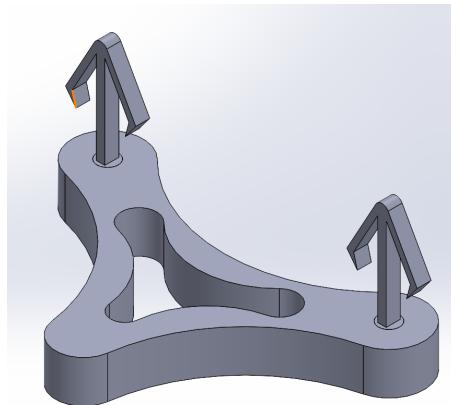
## 1.3 Conception du support encodeur

### 1.3.1 Première version

- Afin de rendre l'assemblage plus rapide, nous avons conçu ces supports qui n'ont pas besoin de vis et d'écrous.

#### Problèmes avec la première version

- La structure est très fragile.



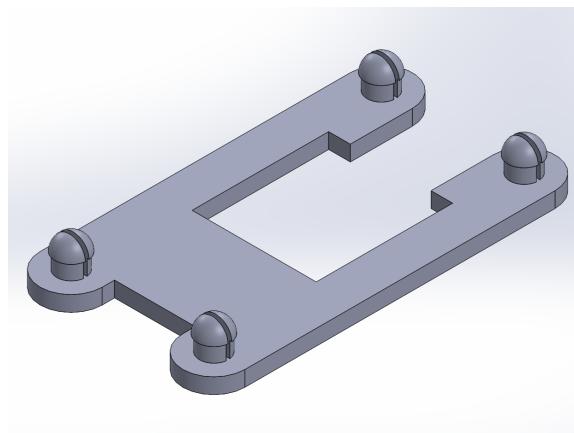
**FIGURE 1.3** – Vue isométrique du support encodeur.

### 1.3.2 Deuxième version

- Une structure plus robuste qui a pour fonction de fixer solidement l'encodeur.

#### Problèmes avec la deuxième version

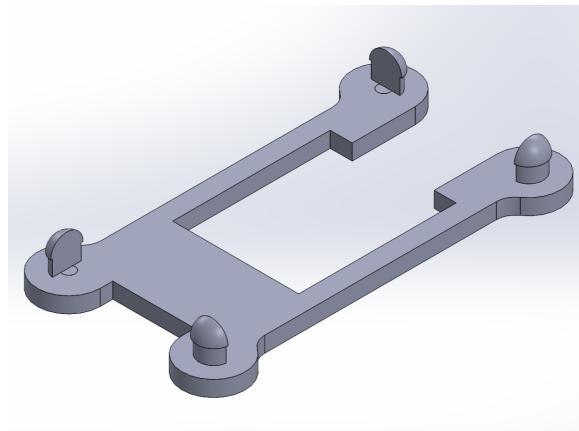
- Lorsque deux languettes flexibles sont montées, il n'y a pas de problème lors du montage et elles sont fixées correctement, mais lors du démontage, l'une des deux languettes se détache.



**FIGURE 1.4** – Vue isométrique du support encodeur.

### 1.3.3 Troisième version

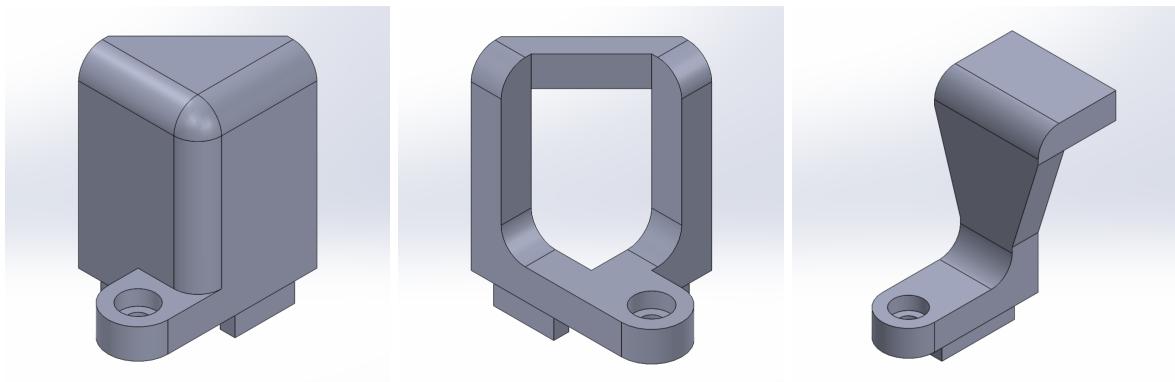
- Les problèmes de démontage et de désassemblage ont été résolus, ce qui permet de les réaliser plus rapidement sans utiliser de vis.



**FIGURE 1.5** – Vue isométrique du support encodeur.

### 1.4 Conception du support de batterie

- Nos modèles sont dotés d'une partie saillante qui sert de pièce de positionnement sur la planche de bois, tout en assurant une plus grande stabilité.
- Sur le côté droit de notre batterie, il y a un trou pour la sortie des câbles d'alimentation.



(a) Support sur le côté gauche. (b) Support sur le côté droite. (c) Support de la partie inférieure.

**FIGURE 1.6** – Modélisation des roues en 3D.

## 2 | Assemblage

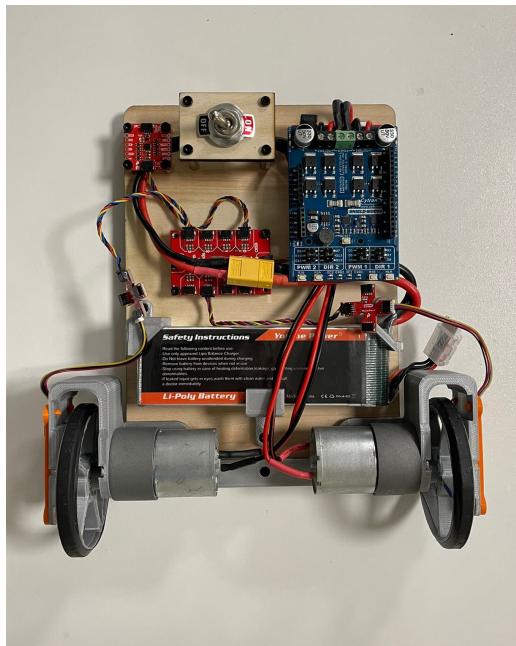
### 2.1 Difficultés

1. Après avoir imprimé le support de moteur, nous avons été informés qu'il fallait utiliser des goupilles de centrage, pour lesquelles nous n'avions pas prévu d'espace dans la conception et nous ne voulions pas réimprimer une pièce qui prend environ 4 heures à imprimer, afin d'économiser du temps et du matériel.
2. Les câbles de l'encodeur ne sont pas assez longs pour être raccordés au multiplexeur.

### 2.2 Solutions

1. 2 trous ont été agrandis dans le support moteur afin de pouvoir installer des pions de centrage (au lieu de vis de fixation). Pour ce faire, nous avons utilisé une perceuse à colonne pour agrandir 2 trous de 3 mm à 4 mm.
2. Deux extensions multiport en forme de croix ont été utilisées.

## 2.3 Photos



(a) Vue de face.



(b) Vista postérieur.



(c) Vue de côté.

**FIGURE 2.1** – Assemblage du robot Segway.

## 3 | Électronique

Ce rapport vise à expliquer la partie électronique du projet MECATRO. Ce rapport est découpé en 5 parties : monitorer, allumer / éteindre les moteurs depuis l'arduino, faire tourner les moteurs à une vitesse proche de celle désirée, interfacer les encodeurs avec l'arduino, interfacer l'IMU avec l'arduino. La réalisation de ces tâches est à la base de la construction d'un robot pouvant se stabiliser lorsqu'il est placé dans une position d'instabilité et lorsqu'il monte une pente allant de 0 à 28°.

### 3.1 Monitorer

Le Monitor s'agit de le Serial Monitor dans le logiciel Arduino. Cette partie est pour voir les transmissions des signaux pendant que l'on utilise Arduino : tout d'abord, selectionnez le port serie appropriée, puis le board Arduino approprié, lancez le fichier d'affichage "Hello World", reglez le debit en bauds du moniteur serie pour qu'il soit coherent avec le programme, et puis on peut voir dans Serial Monitor qu' un nouveau message est constamment affiché, ce qui signifie que nous avons mis en place une communication Serial Monitor.

### 3.2 Allumer / éteindre les moteurs depuis l'Arduino

Grâce à la librairie Mecatrouils, nous avons pu allumer les moteurs avec la ligne de code : mecatro : setMotorDutyCycle(1.0, 1.0) Cette fonction possède 2 arguments, se référant à chacun des deux moteurs. Un argument de 1.0 permet de faire tourner le moteur correspondant à sa vitesse maximale ; respectivement, un argument de 0.0 permet de l'éteindre. Pour plus de détails, veuillez vous référer au code Arduino IDE TelemetryDemo (utilisation de la fonction setMotorDutyCycle à la ligne 56).  
Un lien associé (une vidéo montrant les moteurs en train de tourner est aussi disponible).

### 3.3 Interfaçage des encodeurs avec l'Arduino

Pour pouvoir lire les données mesurées par les encodeurs, nous avons utilisé le code "EncoderDemo" sur ArduinoIDE (disponible dans la librairie Mecatronics). En utilisant la télémétrie permettant d'envoyer les données mesurées sur matlab, on obtient la courbe suivante :

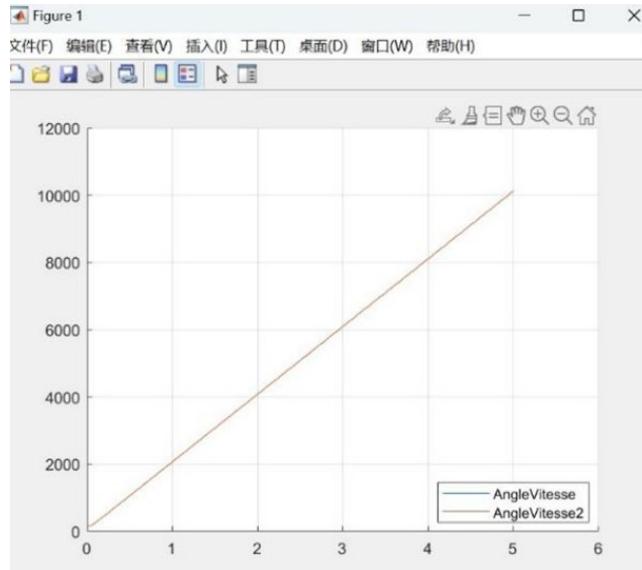


FIGURE 3.1 – Courbe des données mesurées par le encodeur.

Il est à noter que, pour parvenir à lire les données sur Matlab, nous avons dû fermer le moniteur série sur Arduino IDE

### 3.4 Interfaçage de l'IMU avec l'Arduino

Pour pouvoir lire les données mesurées par l'IMU, nous avons utilisé le code "IMUAcquisition" sur ArduinoIDE (disponible dans la librairie Mecatronics) en suivant la même logique que précédemment. Pour la connexion de l'IMU, nous avons effectué les branchements suivants :

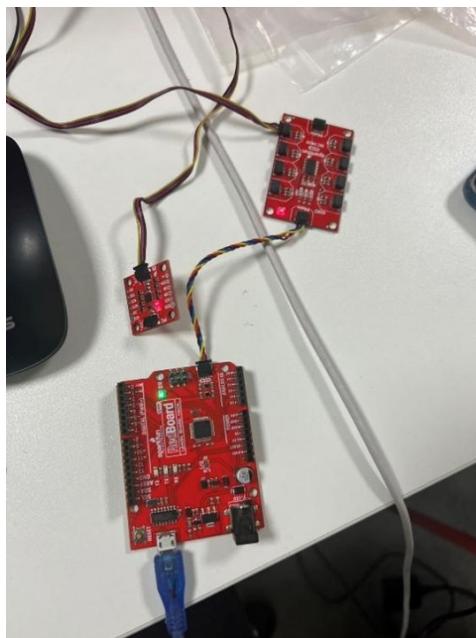
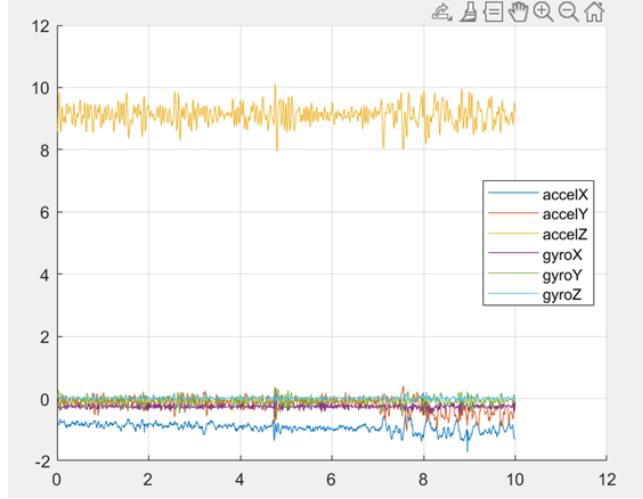


FIGURE 3.2 – Connexion de l'IMU.

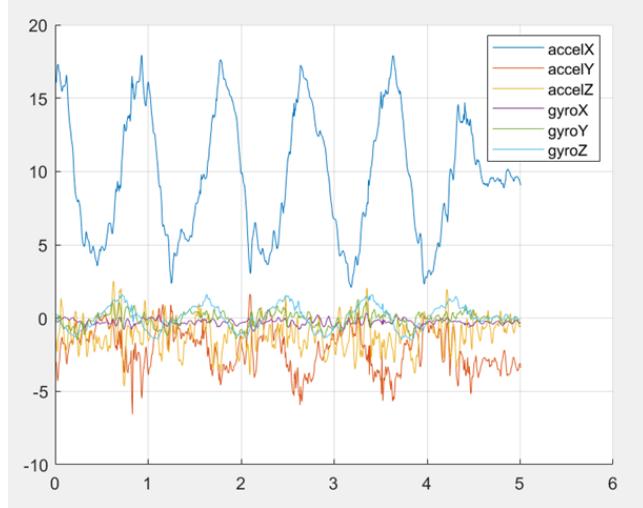
En gardant l'IMU à peu près immobile, on obtient la courbe suivante :



**FIGURE 3.3** – Courbe des données mesurées par l'IMU.

Sur la courbe ci-dessus, on peut voir que `accelZ` se situe entre 9 et 10, ce qui correspond à l'accélération gravitationnelle.

Par la suite, on réalise une nouvelle expérience dans laquelle on effectue un mouvement vertical avec l'IMU, alternativement vers le haut puis vers le bas. Dans cette expérience, "`accelX`" correspond à l'axe vertical (précédemment `accelZ`).



**FIGURE 3.4** – Courbe des données mesurées par l'IMU.

### 3.5 Faire tourner les moteurs environ à une vitesse désirée

On peut faire tourner les moteurs environ à une vitesse désirée en réglant les valeurs des arguments dans la fonction `mecatrol :setMotorDutyCycle(var1, var2)`. Mais cette tâche reste difficile car la librairie AS5600 n'est pas particulièrement performante pour lire les vitesses angulaires (fonction `getAngularSpeed()`). Nous devons donc concevoir notre propre méthode de lecture avec la formule ci-dessous. On utilise la transformée de Tustin pour effectuer cette partie (qui vient de la transformé en z et de l'approximation des trapèzes) :

$$s \longrightarrow \frac{2}{\delta t} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \quad (3.1)$$

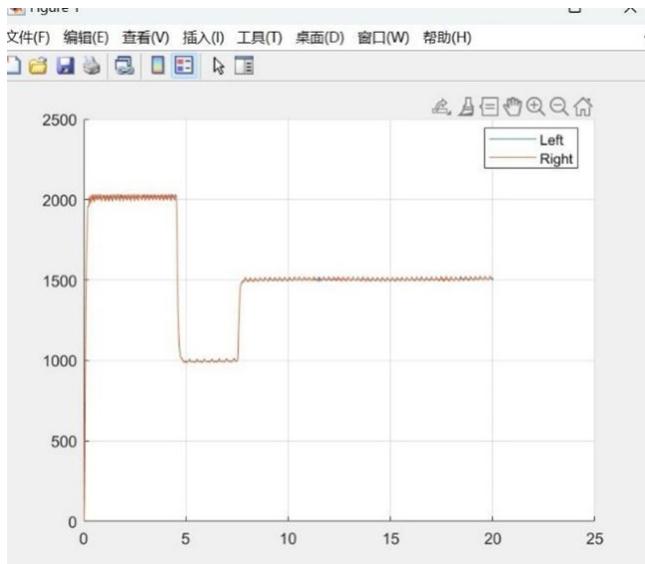
Après les calculs, on trouve que les vitesses angulaires peuvent être écrites sous la forme suivante :

$$x_k = \frac{2\tau^2 x_{k-1} + -\Delta t y_{ek} - x_{k-1} \tau \Delta t - y_{ek-1} \Delta t}{2\tau^2 + \Delta t \tau} \quad (3.2)$$

Et,

$$\Omega_k = x_k + \frac{y_{ek}}{\tau} \quad (3.3)$$

On peut implémenter ces formules dans l'arduino. Par la suite, nous avons fixé les paramètres " $\tau$ ", "d" aux valeurs  $\tau = 0,03$  et  $d = 2,5$  mm où "d" correspond à la distance entre l'encodeur et l'aimant. Puis, nous avons fait l'acquisition des données avec plusieurs échelons de vitesse (Nous n'avons mesuré les données que d'un seul moteur car l'un des capteurs était défectueux). Nous avons constaté que le bruit y était assez faible, ce que nous avons jugé satisfaisant.



**FIGURE 3.5 – Courbe de mesure du moteur.**

Puis, nous utilisons la fonction "getCumulativePosition()" pour obtenir les angles cumulés dans le temps. Pour voir les détails, veuillez voir les codes et vidéos dans les dossiers ci-joints.

## Identification des paramètres électriques :

Paramètres à déterminer : inductance  $L$  des moteurs, résistance  $R$  des moteurs, constante de couple  $k$ .

Afin de déterminer  $R$  et  $k$ , nous avons eu recours à la documentation technique des moteurs utilisés, c'est-à-dire le modèle « Pololu 37 mm numéro 4742 ». La documentation technique fournissait le courant  $I_d = 5,5 A$ , pour une tension nominale de 12 V. Il s'agit des valeurs du courant dans le moteur en régime permanent lorsqu'il n'y a pas de charge sur le rotor et que la vitesse de rotation de celui-ci est nul. Dans ces conditions, on obtient :  $R = \frac{U}{I_d} \approx 2,2 \Omega$ .

Afin de déterminer  $k$ , nous avons utilisé la donnée du courant  $I_v = 0,2 A$  et de la vitesse de rotation à vide  $\Omega_v = 330 \text{ tr/min}$  pour une tension de 12 V. Du fait de la résistance,  $I_v \neq 0$ . Et on a toujours la relation :  $U - RI - k\Omega = 0$ , d'où  $k \approx 0,33 A \cdot \Omega^{-1}$ .

## Identification des paramètres intertiels :

Afin de déterminer le moment d'inertie de l'ensemble {roue + rotor + engrenages} dans un des moteurs, autour de son axe de rotation, nous nous sommes servis de l'équation mécanique d'un moteur à courant continu, qui est :  $I_y^w \dot{\Omega} = ki$ . En régime permanent, on peut exprimer  $i$  en fonction de  $\Omega$  grâce à l'équation électrique du moteur pour obtenir :  $\dot{\Omega} + \frac{k^2}{I_y^w R} \Omega = \frac{kU}{I_y^w R}$ .

Pour une tension d'alimentation constante, cette équation s'intègre facilement sous la forme :

$$\Omega(t) = K e^{\frac{t}{\tau}} + \frac{U}{k} \quad \text{avec } \tau = \frac{k^2}{I_y^w R}.$$

Il s'agit donc de déterminer le temps caractéristique du système. Pour ce faire, nous avons fait tourner le moteur avec un échelon de tension d'alimentation et étudié le régime transitoire grâce à la fonction matlab « tfest », prenant en entrée l'allure vitesse de rotation mesurée, et le modèle de simulation voulu, ici un système d'ordre 1, qui nous a ainsi fourni une estimation de  $\tau$ , donnant accès à  $I_y^w$ , puisque l'on connaît déjà  $k$  et  $R$ . On obtient donc  $I_y^w \approx 0,0067 \text{ kg.s}^{-2}$ . Nous avons également eu accès au gain de la fonction de transfert du système, avec comme entrée  $U$ , et comme sortie  $\Omega$ . Il s'avère analytiquement que le gain  $G$  est égal à  $\frac{\tau}{R}$ , ce qui a permis d'affiner la valeur de  $k$ , que l'on a trouvée égale à 0.35.

Pour le processus de détermination des paramètres, nous utilisons les fonctions tfest et iddata de matlab pour l'ajustement des paramètres. Cette fonction nécessite l'entrée d'une tension en fonction du temps et d'une vitesse de moteur en fonction du temps. Nous avons saisi (1, 0) dans tfest pour représenter le type de fonction nécessaire à l'ajustement des paramètres.

$$s \rightarrow \frac{a}{s + b}$$

Le code Matlab est présenté ci-dessous:

```
legend_labels = data_values.keys;
d1 = legend_labels{1};
d2 = legend_labels{2};
y = data_values(d1);
u = data_values(d2);
data = iddata(y', u', 0.005);
sys = tfest(data, 1, 0);
```

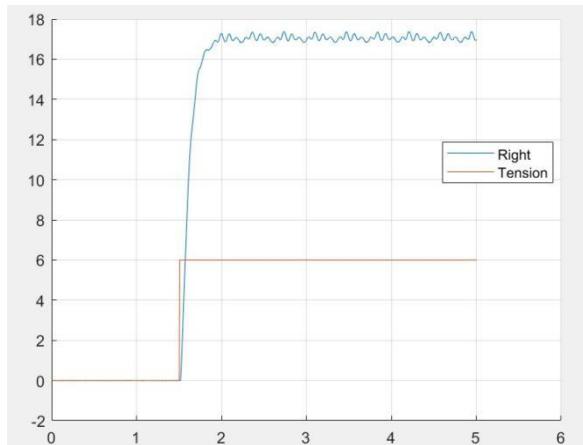


Fig. La fonction de tension et de la vitesse angulaire

Après les calculs de Matlab, nous pouvons obtenir les coefficients pertinents a et b, qui sont respectivement de 24,3 et 8, à partir desquels le moment d'inertie de l'arbre est calculé comme étant de 0,067 kg • m<sup>2</sup>.