CPNV - Centre Professionnel du Nord Vaudois

MCT - Modules complémentaires techniqeus

Rapport PID

P2213

Rédacteur : Quentin Surdez

Relecture : Rafael Dousse

École : CPNV

Date : Yverdon-Les-Bains, le 13 juin 2022

Table des matières

1 Introduction

Ce document a pour but de rassembler toutes les données concernant les essais faits avec le PID. Il permet de revenir sur la méthodologie utilisée pour acquérir les données ainsi que les graphiques des données. Cela va nous permettre d'élaborer une stratégie pour déterminer les différents facteurs P, I et D.

2 Méthodologie

Nous allons ici présenter la méthodologie utilisée pour récolter les données. Il est important de concevoir un cadre dans lequel l'expérience peut être répétée avec le moins de facteurs changeants. Nous allons voir que nous n'avons pas complétement réussi à correspondre à cette définition. Les limites de l'envrionnement ne pouvaient pas être dépassées.

2.1 Environnement

L'envrionnement dans lequel les expériences ont été faites est les couloirs du centre de St-Roch à Yverdon. Un espace peut être créé en déplaçant tables et chaises afin que notre robot rencontre le moins d'obstacles.

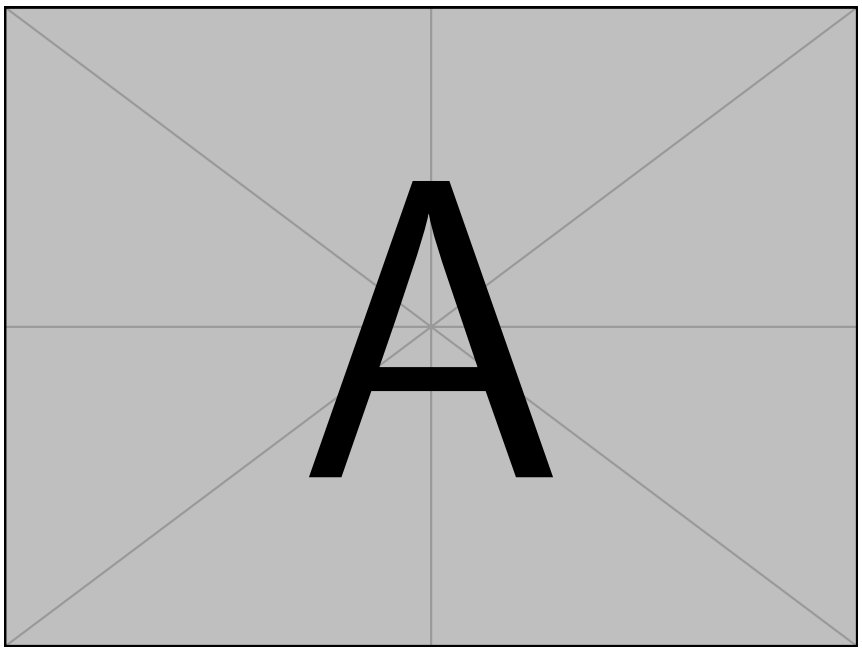


FIGURE 2.1 – Photo de l'espace avec tables et chaises rangées

Nous pouvons voir sur cette image que l'espace d'expérimentation est restreint. Cela a pour conséquences que toutes les expériences ne se déroulent pas exactement dans les mêmes conditions. Parfois, le robot s'approche d'un obstacle et nous le déplaçons rapdidement pour qu'il ne rentre pas en collision avec un objet. Cela ammène inévitablement des erreurs dans les données. Nous verrons plusieurs fois des pics dans les graphiques expliqués par ce problème.

L'installation du robot a aussi changé entre le mode roue libre, ou roue avec frottements dans l'espace ci-dessus. En roue libre, l'Arduino est connecté à l'ordinateur, ce qui facilite la transmission des données. Lorsque le robot roule, l'Arduino est connecté au Raspberry PI qui est lui-même connecté via VNC à l'ordinateur.

2.2 Code

Pour récupérer les données que nous souhaitons analysées, il a fallu faire quelques modifications au niveau du code. Premièrement une variable $Deboggage$ a été créé pour que le code soit facile d'accès, mais aussi facilement évité lorsque nous ne sommes pas en phase de test. Ensuite, il a fallu définir quelles valeurs nous souhaitions utiliser. Après réflexion nous utilisons la valeur des PWM envoyés aux deux moteurs, soit $controleur$, pour le moteur A, et $controleur1$. pour le moteur B. À ces valeurs s'ajoutent les ticks pour chaque moteur. Cela nous permet de connaître la vitesse lue et ce qui est envoyé via le PID.

Ensuite, nous avons choisi de saisir ces données dans un tableau à deux dimensions. Avec, au début 250 lignes et 4 colonnes, puis avec la fréquence d'acquisition qui a baissée, nous sommes passés à un tableau de 100 lignes pour 4 colonnes. Le choix de changement de fréquence sera expliqué par la suite.

Pour écrire chaque colonnes pour chaque ligne voici notre code appelé à chaque fois que la fonction $asservissement()$ s'execute :

```
// Ecriture dans un tableau
if (Deboggage)
{
    tableau[i][0] = controleur;
    tableau[i][1] = vitesse;
    tableau[i][2] = controleur1;
    tableau[i][3] = vitesse1;
    i++;
}
```

FIGURE 2.2 – Code d'écriture dans le tableau

Nous voulons que le tableau, une fois rempli, soit affiché dans le Serial pour faciliter l'extraction de données. Nous voulons aussi créer un fichier .csv, cela impact notre routine d'impression. Voici le code source :

```
//Ecriture du tableau dans Serial
if (Deboggage)

{
    if (i == 250 && on == 0)
    {
    stop();
    for (i = 0; i < 250; i++)
    {
        for (j = 0; j < 4; j++)
        {
        Serial.print(tableau[i][j]);
        Serial.print(", ");
        }
        Serial.println("");
        on = 1;
    }
    }
}
```

FIGURE 2.3 – Code pour transférer les données du tableau

Nous pouvons observer que nous envoyons une virgule et un espace entre chaque valeur et un retour à la ligne entre chaque ligne de notre tableau. Cela nous permet de créer un fichier .csv facilement.

2.3 Data

Avec notre fichier .csv sur notre Raspberry PI, nous pouvons l'enregistrer sur une clef USB. Le nom du fichier donne tous les paramètres de l'expérience. Pour que le fichier de données soit lisible par LaTeX nous devons exécuter plusieurs commandes. Tout d'abord copier les données bruts dans Excel. Ensuite convertir les données en mettant comme séparateur la virgule. Ensuite ajouté une ligne tout en haut pour nommer les colonnes, $y1$, $y2$, $y3$, $y4$. Insérer une colonne au début pour le temps nommé x. Convertir chaque colonne en texte et exporté le fichier sous format .prn pour avoir un séparateur par espace.

Après avoir fait ces différentes opérations, il est nécessaire de créer un grapique. Cela nous permet de visualiser concrètement ce qu'il se passe dans notre système et d'en tirer des conclusions.

3 Extrapolation des données

Nous allons montrer les différents graphiques que nous avons obtenu avec nos différentes expériences. Les premiers graphiques sont faits en roue libre. Le moteur A est le moteur gauche du robot et le moteur B est le moteur droite du robot.

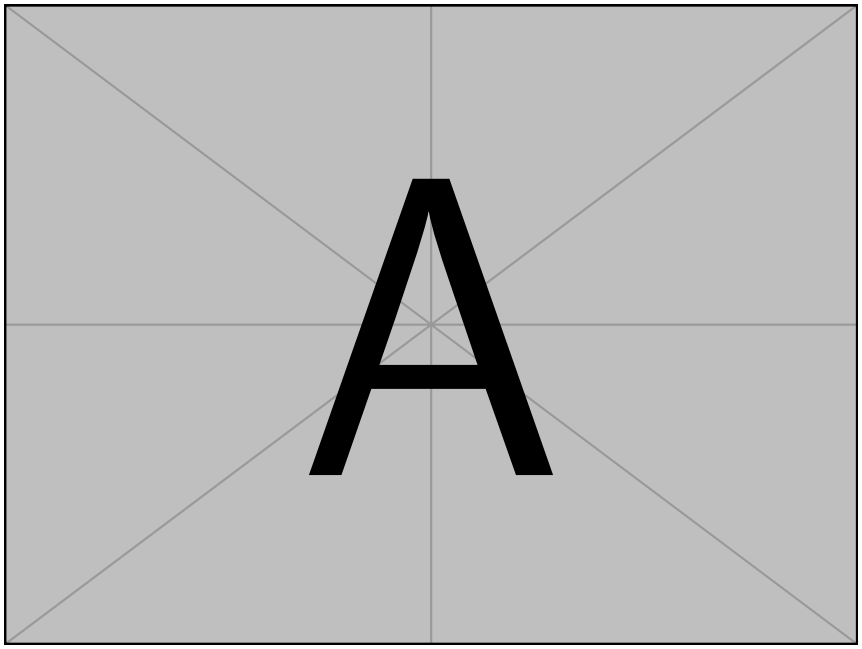
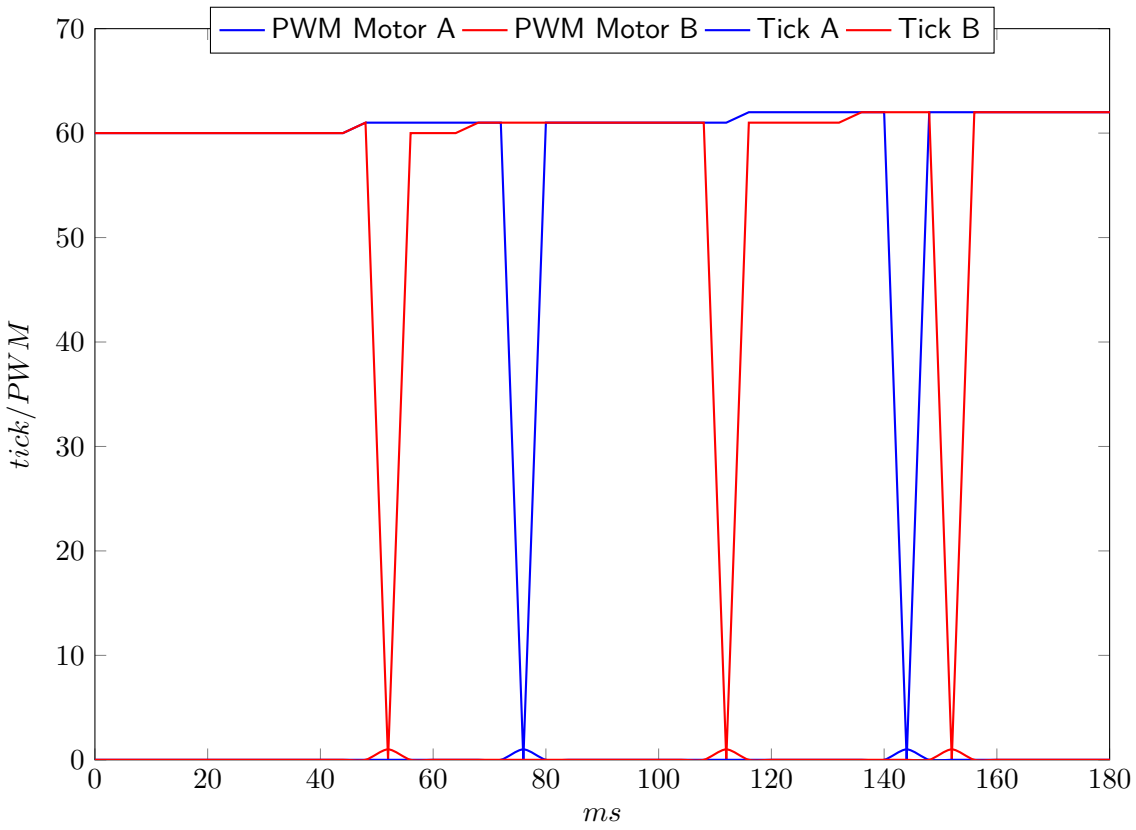


FIGURE 3.1 – Photo du robot avec des annotations pour voir le moteur A et le moteur B

3.1 Graphiques



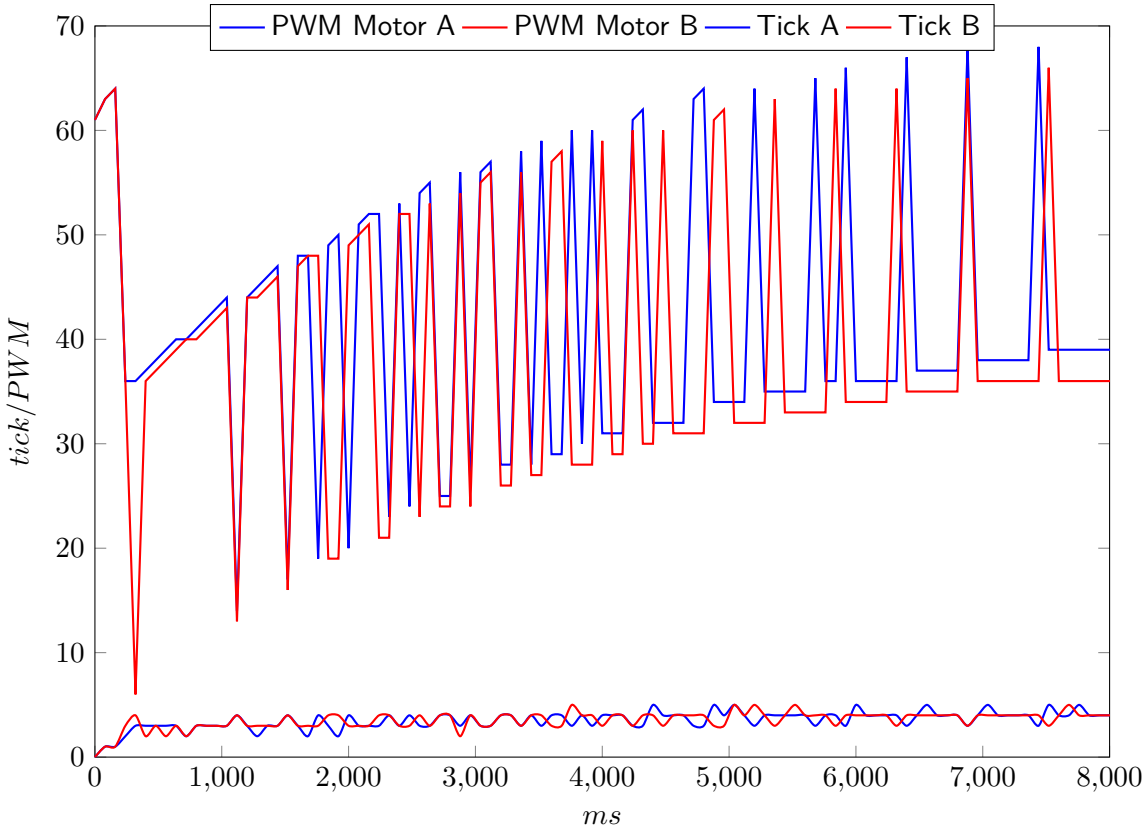
- Fréquence : 250Hz
- P : 30
- I : 0.01
- D : 0.001
- tours/s : 3

Ce premier graphe nous montre comment se comportait notre système avec les paramètres que nous avions choisi dans le cours du développement. On peut voir que le nombre de tick est toujours égal à 0 sauf lorsque les PWM tombent à 0. Là les ticks sont à 1. C'est à partir de ce graphique que nous avons refait nos calculs pour qu'au minimum 3 ticks se fassent entre deux échantillonages par la fonction d'asservissement.

$\varnothing\ 90 = D$
$D \cdot \pi = P$
$90 \cdot 10^{-}3 \cdot \pi = 0.2827m$
$1tour = 0.2827m$
$3\frac{tour}{s} \cdot 3.6 = 0.8481\frac{m}{s} \cdot 3.6 = 3.05\frac{km}{h}$
Une roue compteuse possède 24 trous, donc 24 incrémentations par tour
$24 \cdot 3 = 72$
$3/72 = 0.04s$
Pour être sûr d'avoir suffisamment de trous nous mettons une fréquence à 12.5Hz, donc $80ms$

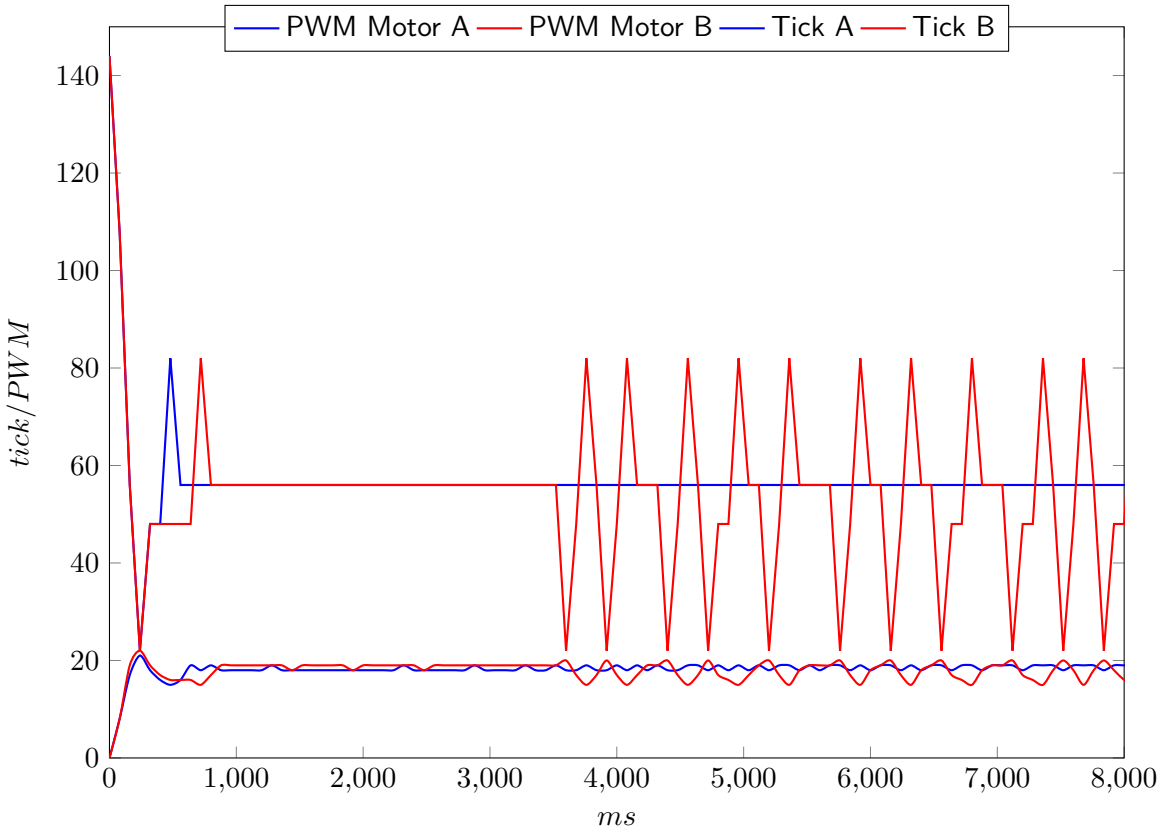
FIGURE 3.2 – Équation du PID

Nous avons donc changé la fréquence d'acquisition à 12.5Hz. Voici le graphe que nous avions avec nos paramètres :



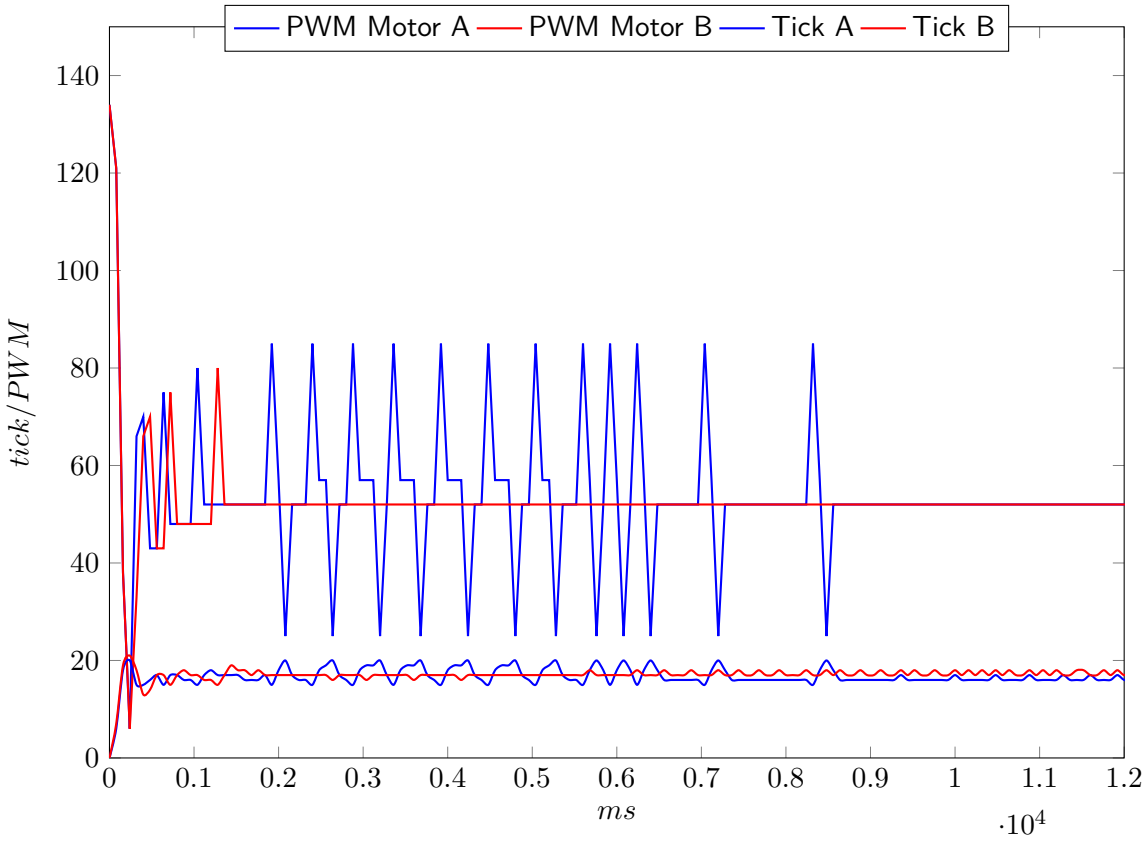
— Fréquence : 12.5Hz
— P : 30
— I : 0.01
— D : 0.001
— tours/sec : 3

Le comportement du PID se rapproche de ce que nous souhaitons voir. Le nombre de tick moyen est 4, ce qui est raisonnable, en sachant que pour créer une parabole, nous avons besoin de 2 points minimum et plus est toujours mieux. À partir de ce graphique nous avons décidé de refaire notre cheminement pour obtenir un PID robuste. Le P est resté inchangé.



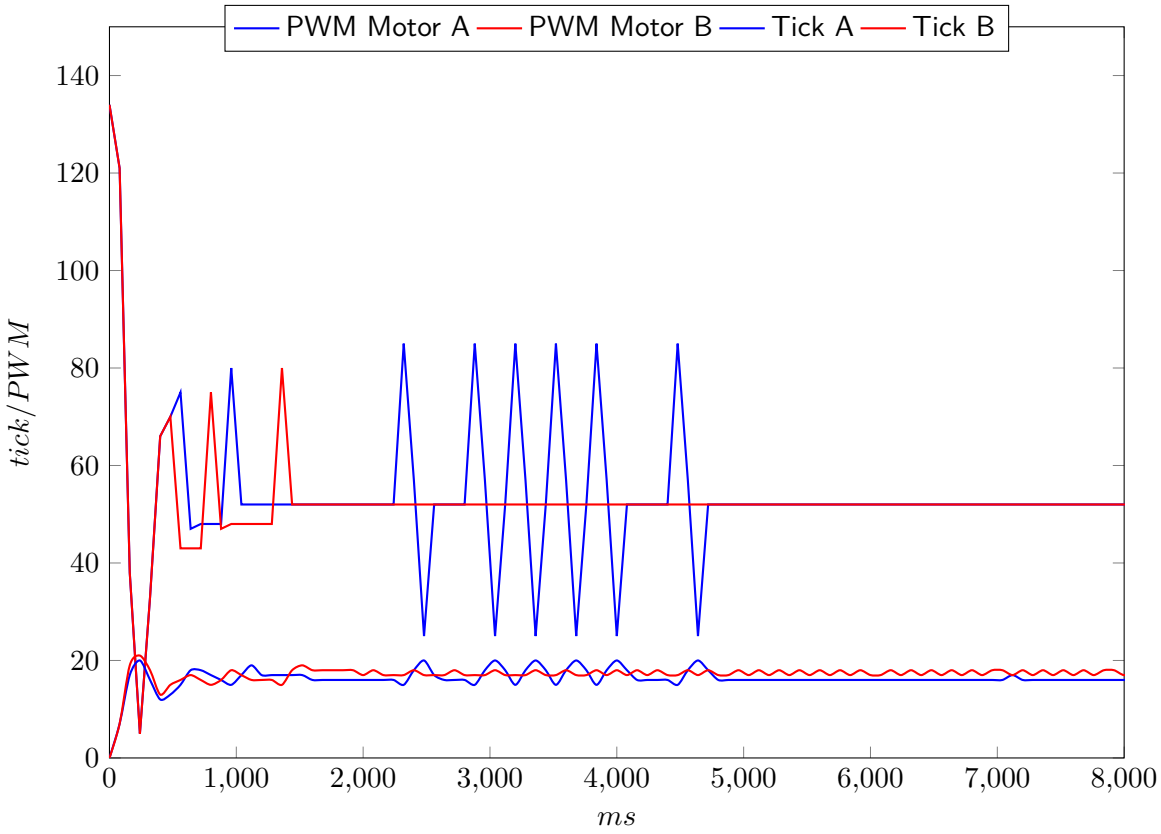
- Fréquence : 12.5Hz
- P : 30
- I : 0.05
- tours/s : 4

Nous avons changé quelques paramètres. Le I est à 0.05 et les tours/s sont de 4. On peut observer que ces sursauts de la part des PWM proviennent du I qui est trop grand pour le système à roue libre. Les 4 tours/s aident pour la vitesse de prise des données. Nous avons donc descendu le facteur I.



- Fréquence : 12.5Hz
- P : 30
- I : 0.03
- tours/s : 4

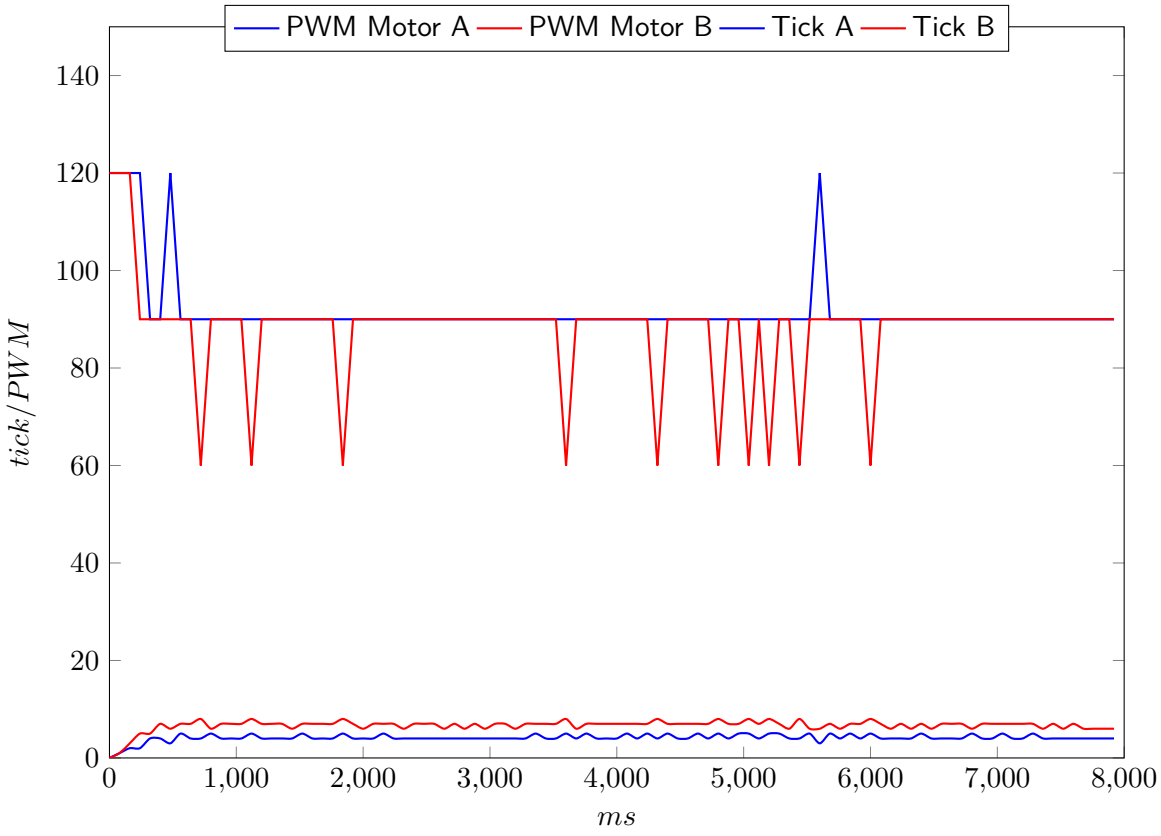
Nous avons choisi ici de présenter une plus grande plage de données. En effet, ce que nous recherchons est la stabilisation du système. Nous l'avons après $10s$. À partir de ce graphe nous avons essayé une valeur pour le D.



- Fréquence : 12.5Hz
- P : 30
- I : 0.03
- D : 2
- tours/s : 4

Nous voyons ci-dessus une stabilisation bien plus rapide avec le facteur D. Ce graphe nous est satisfaisant et nous passons sur les mêmes paramètres mais avec des frottements. Nous avons recommencé le même processus en sachant l'ordre de grandeur des différents facteurs dont nous aurons besoin grâce aux précédents graphiques.

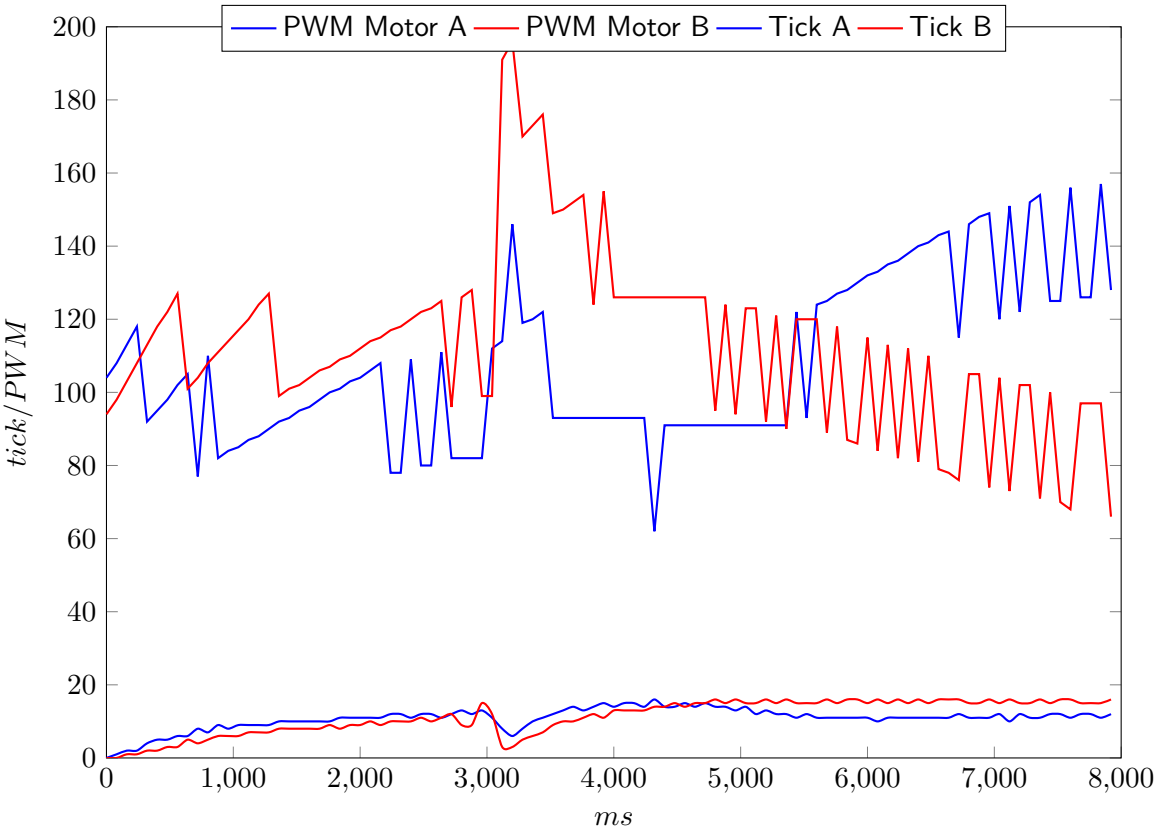
Dans les prochains graphiques, nous allons surtout nous intéresser au fait que les courbes des ticks soient le plus similaire possible, ce qui se traduit par une vitesse sembable.



— Fréquence : 12.5Hz
— P : 30
— Frottements
— 4 tours/s

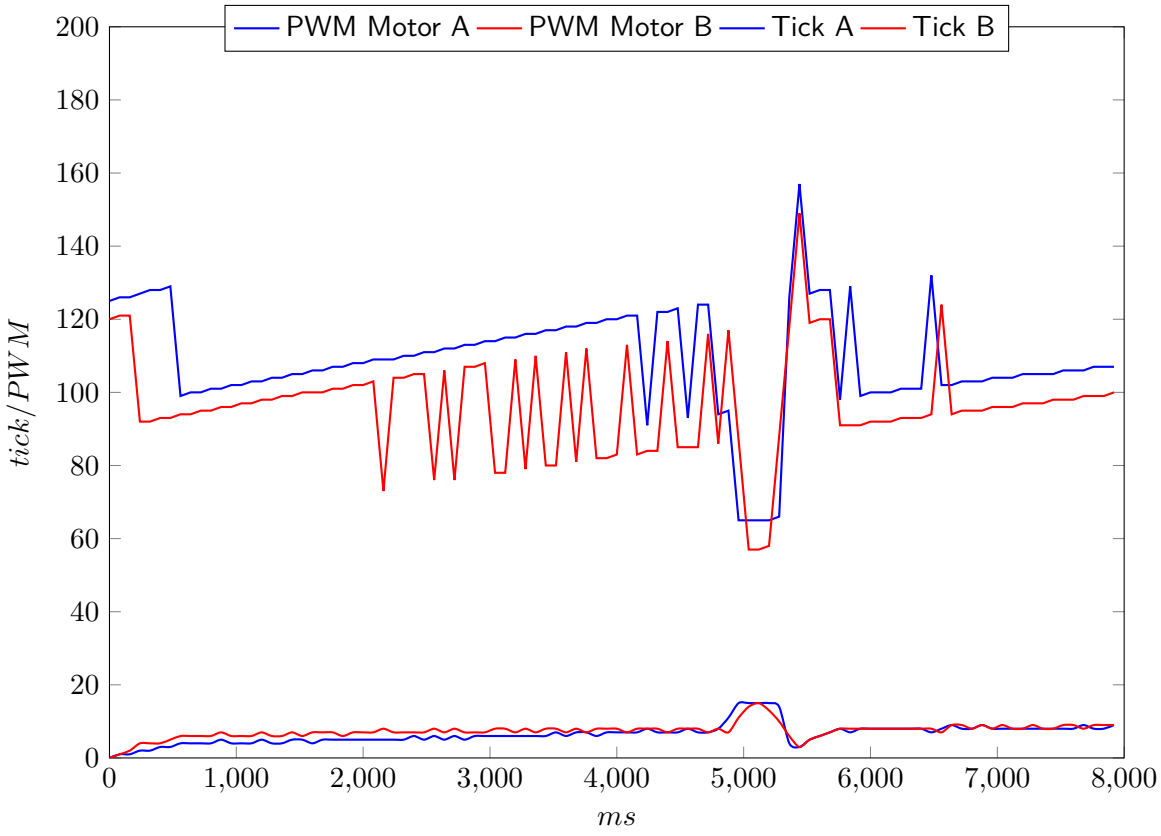
Le comportement du robot observé était qu'il tournait sur lui-même dans le sens anti-horaire. Nous voyons que les courbes des PWM sont semblables, mais les ticks sont décalés. Le moteur B tournait plus vite que le moteur A. Nous allons voir le comportement avec le facteur I.

Le prochain graphique comporte beaucoup de changement. Nous avons ajouté 10 PWM pour le contrôleur du moteur A et changé la vitesse à 3 tours/s. Si c'était à refaire, nous ferions plus d'acquisition de données. Il y a un changement des conditions trop brusques.



- Fréquence : 12.5Hz
- P : 30
- I : 0.01
- Frottements
- 3 tours/s
- Moteur A + 10

Lors de ce test le robot a dû être replacé pour éviter un obstacle. Cela explique le changement brutal au milieu. Le robot a d'abord tourné à gauche, puis après le repositionnement à droite. On en déduit que la valeur ajoutée au moteur A est trop grande. Nous allons la changer ainsi que le nombre de tours et le facteur I qui reste encore trop abrupt.



— Fréquence : 12.5Hz
— P : 30
— I : 0.001
— Frottements
— 4 tours/s
— Moteur A + 5

Le robot a d'abord tourné sur la gauche, puis est rentré dans un obstacle. Après l'avoir reposé à terre, il est allé tout droit. On peut voir que les courbes des ticks se rejoignent bien après l'erreur du repositionnement. Nous nous approchons des valeurs qui nous permettraient d'avoir un robot qui avance tout droit.

4 Conclusion