Projet de recherche

Asservissement en position et en vitesse :
ajustement des coefficients de PID par algorithme

Responsable du projet : Maxime Bavencoffe

Table des matières

I. Objectifs

L'objectif de ce projet est de réaliser une maquette pédagogique afin de mettre en place l'asservissement d'un moteur à courant continu à l'aide d'un correcteur de type PID (Proportionnel Intégrale Dérivé). Ce projet est réalisé à l'aide de composants simples et une attention particulière a été portée pour faciliter son adaptation à d'autres composants.

Dans ce document est détaillé l'ensemble des étapes permettant de reproduire la maquette réalisée.

La maquette est constituée d'un moteur à courant continu et son driver, d'un encodeur pour suivre les mouvements du moteur et d'une carte Arduino pour calculer l'asservissement et commander le moteur. L'Arduino se connecte en USB à un ordinateur pour recevoir les valeurs des coefficients du PID et la consigne de vitesse. En échange, elle renvoie la vitesse instantanée du moteur toutes les 10 ms, ce qui permet de tracer la réponse du moteur en fonction des coefficients.

II. Matériel nécessaire

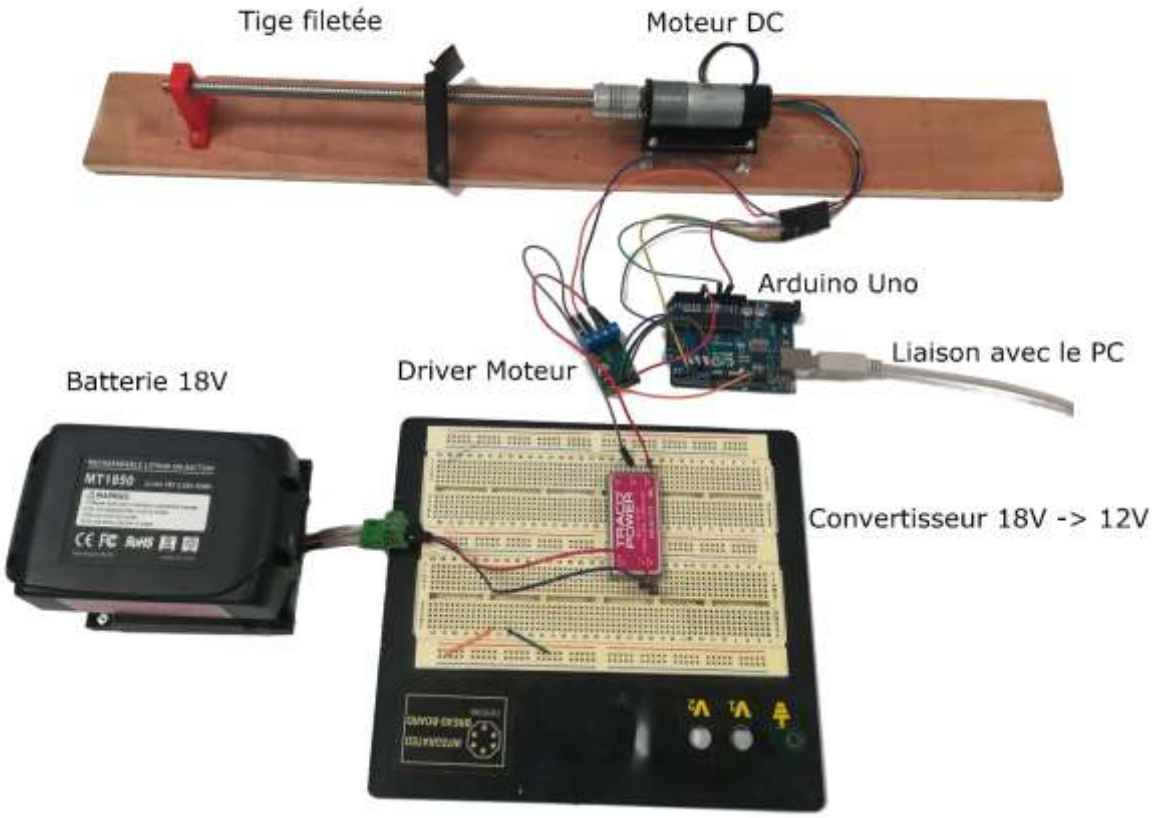
La maquette présentée ici a été réalisée à l'aide du matériel disponible au club robotique de l'INSA CVL. D'autres composants peuvent être plus adaptés à la réalisation d'une seconde version de la maquette.

Matériel	**Lien d'achat**
Arduino UNO	https://fr.rs-online.com/web/p/arduino/7154081
Moteur DC 12V avec encodeur	https://www.pololu.com/product/4863
G2 High-Power Motor Driver 24v13	https://www.pololu.com/product/2992
Vis Trapézoïdale 8mm et écrou en laiton	https://www.amazon.fr/dp/B07CXNK2Z6
Coupleur 5mm -> 8mm	https://www.amazon.fr/dp/B07T1CCVZH

Le moteur et le driver peuvent facilement être remplacés par d'autres modèles. Un pont en H est suffisant pour contrôler le moteur et ne demande que peu de modifications dans le code de l'Arduino.

A cette liste doit également être rajouté un système d'alimentation pour le moteur. Dans notre cas, nous avons utilisé une batterie de perceuse type Makita 18V reliée au moteur au travers d'un convertisseur de tension 18V -> 12V TRACO TEN 30-1212 (https://fr.rs-online.com/web/p/convertisseurs-dc-dc/1665548). Ce système peut par exemple être remplacé par un générateur de tension en salle TP.

Finalement, nous obtenons le montage suivant :



III. Principe de fonctionnement d'un asservissement de type PID

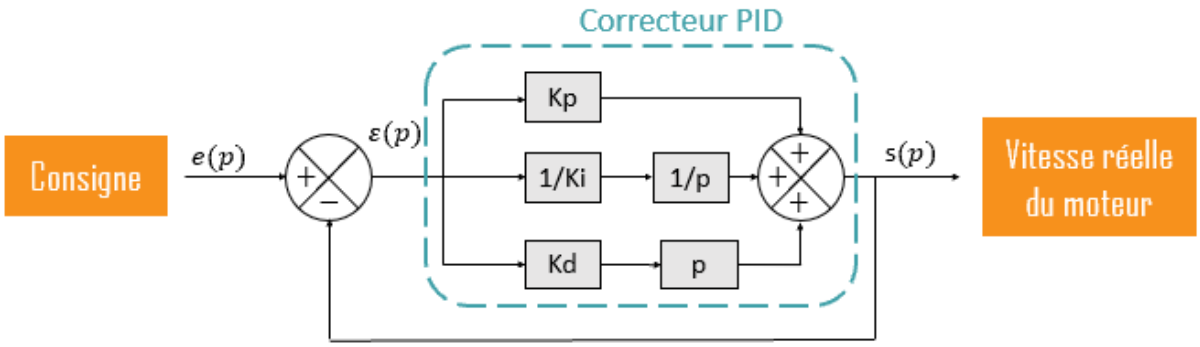
A. Etude théorique

Qu'est-ce que l'asservissement en vitesse ?

L'asservissement en vitesse d'un moteur est un processus qui vise à réguler et maintenir la vitesse d'un moteur électrique à une valeur de consigne prédéterminée. Cela implique l'utilisation d'un système de contrôle qui mesure en continu la vitesse réelle du moteur et ajuste en conséquence sa tension ou son courant d'alimentation pour maintenir la vitesse souhaitée.

Qu'est-ce que l'asservissement en position ?

L'asservissement en position d'un moteur vise à commander sa distance de rotation plutôt que sa vitesse, ce qui permet le contrôle précis de la position du système mécanique qui lui est lié. Au début du mouvement, l'erreur est très importante, car la position est loin de celle souhaitée. L'asservissement cherchant à réduire l'erreur, la position se rapproche de la consigne jusqu'à avoir une erreur nulle.



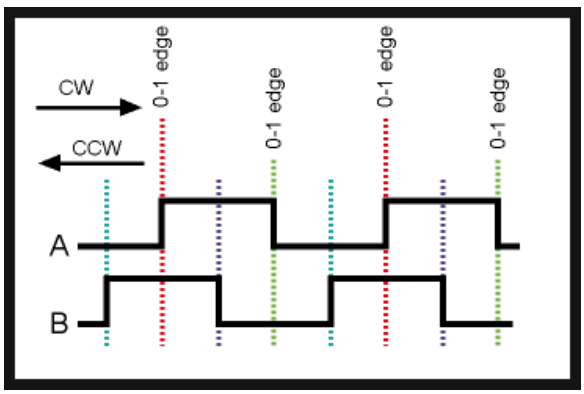
Les principaux éléments d'un système d'asservissement en vitesse et en position sont :

- Un capteur (ex : commutateur optique, encodeur magnétique, ...) : permet de déterminer la vitesse réelle de rotation du moteur en comptant un nombre d'impulsion.
- Un comparateur : compare la valeur mesurée de la vitesse (ou de position) avec la valeur de consigne prédéterminée. La différence entre la vitesse mesurée et la consigne est appelée erreur.
- Un régulateur : reçoit l'erreur du comparateur et génère un signal de commande en conséquence. Ce signal est généralement un signal de tension ou de courant qui va agir sur le moteur.
- Un actionneur (ex : driver moteur, pont en H, ...) : reçoit le signal de commande du régulateur et génère une tension ou un courant approprié pour alimenter le moteur.
- Un moteur

Un microprocesseur (ici une Arduino Uno) sert à la fois de comparateur et de régulateur.

B. Etude pratique

Pour réaliser un PID, l'Arduino compte le nombre de fronts présents sur les 2 canaux de l'encodeur à effet Hall grâce à des interruptions sur fronts montants et descendants.



Une fonction de calcul de l'asservissement est appelée toutes les $\Delta t = 10\ ms$ pour calculer le PID et en déduire une nouvelle consigne pour le moteur. Les étapes pour un asservissement en vitesse sont les suivantes :

1. Calcul de la vitesse réelle du moteur à partir du nombre fronts montants et de fronts descendants du codeur incrémental pendant la période Δt

$$V_{relle} = \frac{2 * \pi * nb\_Fronts\_Codeur}{nb\_Impulsion\_Par\_Tour * Rapport\_Reduction} * \frac{1}{\Delta t}$$

   nb_Fronts_Codeur correspond au nombre de fronts pendant l'intervalle de temps Δt

2. Calcul de l'erreur instantanée $\varepsilon_x = Vitesse\_Consigne - Vitesse\_Reelle$
3. Calcul du terme issu du correcteur proportionnel $P_x = Kp * \varepsilon_x$
4. Calcul du terme issu du correcteur intégral $I_x = I_{x-1} + Ki * \varepsilon_x * \Delta t$
5. Calcul du terme issu du correcteur dérivé $D_x = \frac{Kd*(\varepsilon_x - \varepsilon_{x-1})}{\Delta t}$
6. Calcul de la nouvelle consigne moteur $Consigne\_Moteur = P_x + I_x + D_x$
7. Vérification que la consigne est dans la plage autorisée par le driver (ici $[-400; +400]$), sinon on borne la consigne moteur à l'intervalle autorisé
8. Application de la nouvelle consigne au moteur

Les coefficients Kp, Ki et Kd sont à déterminer expérimentalement.

Pour réaliser l'asservissement en position, le procédé est identique à l'exception du calcul de l'erreur de l'étape 2. L'objectif à atteindre n'est plus une vitesse mais un nombre de fronts par l'encodeur. On calcule donc l'erreur tel que :

$$\varepsilon_x = Consigne\_Nb\_Fronts - Nb\_Fronts\_Total$$

Nb_Fronts_Total correspond au nombre total de fronts depuis l'origine

$Consigne\_Nb\_Fronts$ correspond au nombre de fronts pour atteindre la position désirée

On peut transformer une distance en nombre de fronts en connaissant les caractéristiques de notre système :

$$Consigne\_Nb\_Fronts = \frac{distance * nb\_Impulsion\_Par\_Tour * Rapport\_Reduction}{Pas\_Tige\_Filtee}$$

NB : Le code Arduino utilisé est disponible en annexe.

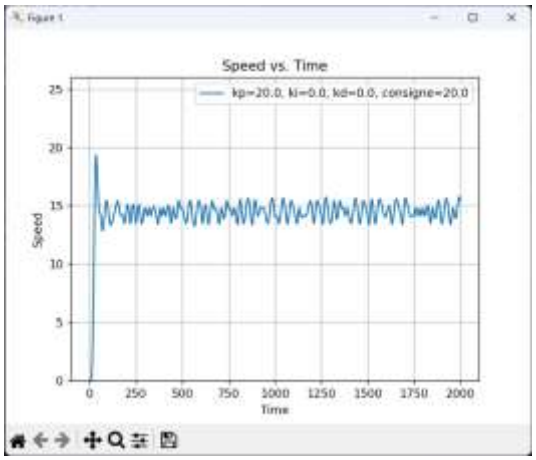
IV. Fonctionnement de l'application

Pour faciliter le réglage des coefficients, nous avons développé une application en ligne de commande qui permet de configurer les coefficients du PID sur l'Arduino et de tracer la réponse du moteur. Les réglages ne sont pas sauvegardés dans l'Arduino. Il faut donc les régler à chaque redémarrage du système.

Pour faire tourner le moteur avec de nouveaux coefficients, utilisez la commande « test ». Vous serez ensuite invité à indiquer les valeurs de vos coefficients, votre consigne de vitesse et le type d'asservissement (v = asservissement en vitesse, p = asservissement en position). Un coefficient dont la valeur est mise à 0 permet de le désactiver. Par exemple, si Ki et Kd sont à 0, l'asservissement réalisé sera de type P uniquement :

```
Enter a command: test
Enter kp (default 0.0): 20
Enter ki (default 0.0):
Enter kd (default 0.0):
Enter consigne (default 20.0):
Enter type asservissement (v/p) (default v):
```

Pour afficher la courbe obtenue, utilisez la commande « show ». Une fenêtre s'ouvre avec votre courbe :



Vous pouvez réaliser plusieurs tests avec des coefficients différents. Ils s'afficheront dans la même fenêtre pour faciliter leur comparaison. Vous pouvez également lister les tests réalisés avec la commande « list » et supprimer un test avec la commande « remove » :

```
Enter a command: list
0 TestVitesse(kp=20.0, ki=0.0, kd=0.0, consigne=20.0)
1 TestVitesse(kp=40.0, ki=0.0, kd=0.0, consigne=20.0)
Enter a command: remove
Enter index of test to delete: 1
```

L'application est développée en Python et le code source est disponible en annexe.

V. Format de communication entre l'Arduino et l'application

Le protocole de communication entre l'Arduino et l'application Python est repris de « TACTISOFT », le logiciel embarqué dans les robots du club robotique de l'INSA CVL. Il suit le format suivant :

PREFIX+nom_de_la_commande=argument1, argument2

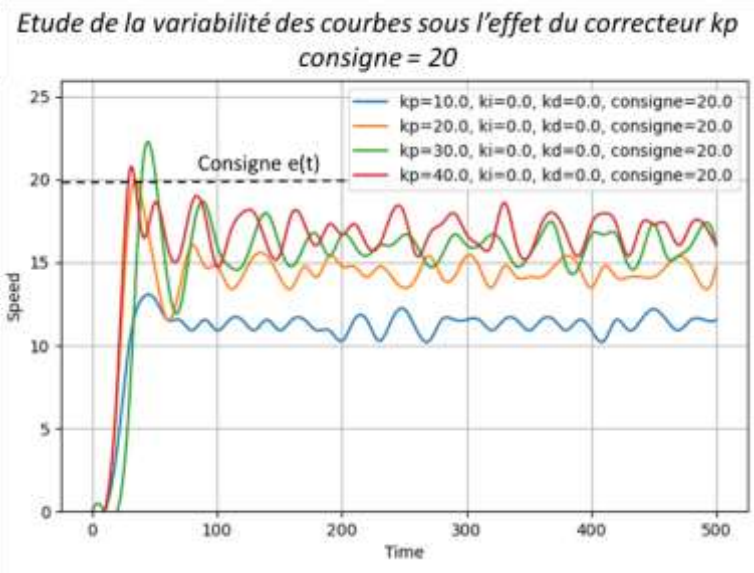
Le préfix choisit est « PR » (pour Projet Recherche). Une commande pourra donc prendre la forme suivante : PR+kd=20 (régle le coefficient Kd de l'asservissement à 20)

Le tableau suivant liste l'ensemble des commandes disponibles :

Nom de la commande	Description
ping	Retourne pong, permet de s'assurer du bon fonctionnement de la communication
kp=valeur	Permet de régler le coefficient Kp de l'asservissement à valeur
ki=valeur	Permet de régler le coefficient Ki de l'asservissement à valeur
kd=valeur	Permet de régler le coefficient Kd de l'asservissement à valeur
consigne=valeur	Permet de régler la consigne de vitesse du moteur à valeur
log=0/1	Permet d'activer l'envoi de la vitesse réelle du moteur toutes les 10ms 0 : Désactivation 1 : Activation
reset	Permet de réinitialiser l'asservissement
asservissement_position=0/1	Permet de choisir entre un asservissement en position et un asservissement en vitesse 0 : Asservissement en vitesse 1 : Asservissement en position

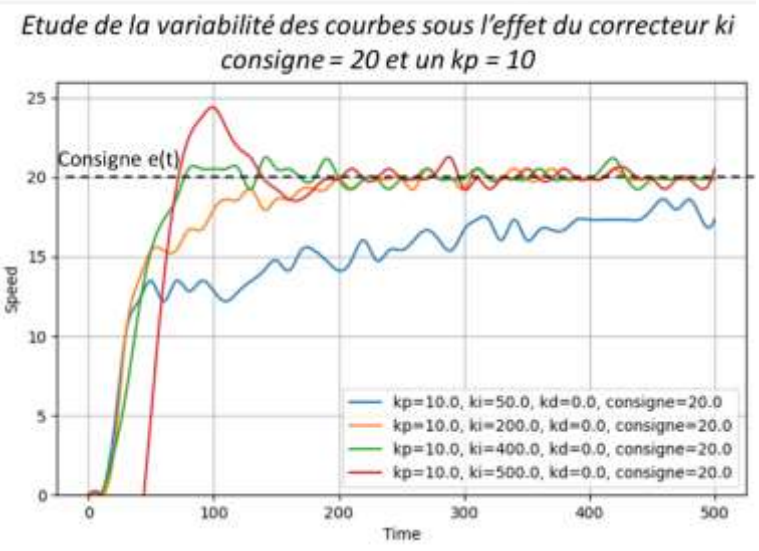
VI. Résultats obtenus

Pour régler nos coefficients, nous commençons par réaliser un asservissement de type P. Les tests pour différentes valeurs de Kp nous permettent d'obtenir les courbes suivantes :



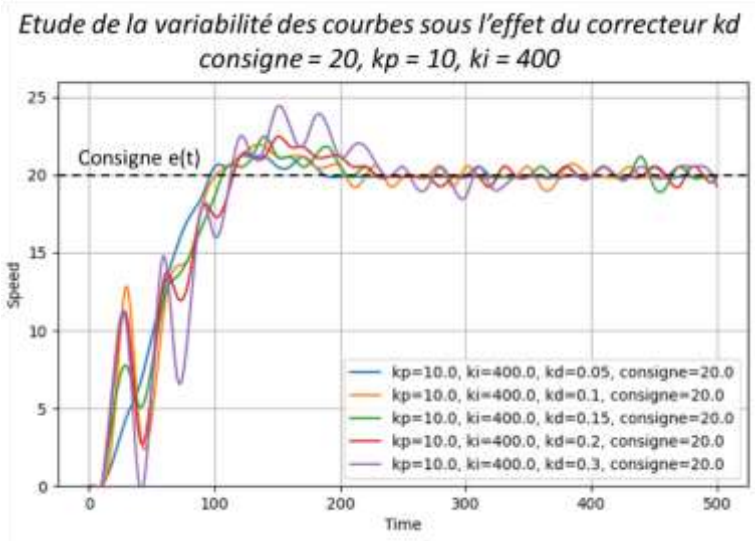
Une augmentation du coefficient proportionnel Kp diminue l'erreur statique et le temps de réponse du système. Toutefois, le système devient oscillant. Un Kp de 10 semble adéquat pour réduire au maximum ces oscillations, bien que l'erreur statique soit importante.

Pour réduire l'erreur statique, on ajoute un correcteur Ki :



Une augmentation de Ki entraine une réduction de l'erreur statique du système. Toutefois, cela augmente le temps de réponse du moteur et engendre des oscillations (voir un dépassement important si Ki est trop important). Un Ki de 400 semble ici un bon compromis.

Le coefficient dérivé Kd permet de rendre le système plus stable.



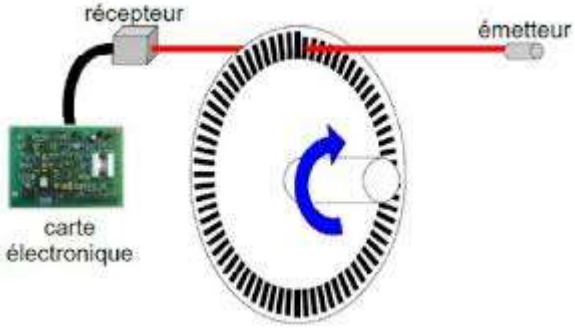
Dans notre cas, le coefficient dérivé à tendance à détériorer notre asservissement. Ce phénomène vient sans doute des défauts de notre maquette. Le couplage du moteur avec la tige filetée n'est pas parfaitement centré et la tige filetée repose dans un trou non ajusté de l'autre côté. Un défaut périodique apparait donc dans la rotation du moteur.

Nous obtenons toutefois de très bons résultats avec l'asservissement de type PI. Il n'est pas toujours nécessaire d'utiliser tous les coefficients (seul Kp est toujours présent). On retient donc les coefficients suivants pour notre asservissement : Kp = 10 Ki = 400 Kd = 0

VII. Difficultés rencontrées

Pour réaliser la maquette, nous avons d'abord utilisé un moteur sans encodeur, auquel nous avons fixé une roue codeuse sur l'arbre de sortie. Pour réaliser l'encodeur, nous avons utilisé un commutateur optique à fourche (TCST 2103) qui permet de détecter les trous dans la roue codeuse et d'envoyer une impulsion à l'Arduino pour calculer la vitesse de rotation du moteur. Cependant, cette méthode n'était pas suffisamment fiable, car il nous était difficile de positionner correctement le capteur et les impulsions étaient trop éloignées les unes des autres, car la roue codeuse était positionnée après le réducteur. Nous avons donc décidé de changer de moteur, le nouveau étant équipé d'un encodeur à effet Hall sur son arbre avant réducteur. Le nouveau capteur à effet Hall peut être installé directement sur les pins Arduino à la place du capteur optique à fourches.

Roue codeuse



Principe de fonctionnement d'un encodeur avec roue codeuse

VIII. Perspective d'améliorations

Le PID réalisé permet donc d'obtenir un contrôle précis du moteur. Cependant, il faut au préalable régler les coefficients. Il s'agit d'un processus long et fastidieux qui ne permet pas toujours d'obtenir la meilleure paire de coefficient. Des algorithmes pour les trouver ont donc été développé. Parmi eux, on retrouve notamment l'algorithme de recherche des corbeaux ou Crow Search Algorithm (CSA).

Il s'agit d'une méthode d'ajustement automatique des coefficients qui s'inspire des oiseaux pour résoudre des problèmes complexes. Des corbeaux dotés d'une mémoire sont positionnés dans l'espace de recherche. A chaque itération, la position des corbeaux est évaluée avec une fonction de coût. Si la position est meilleure que sa position en mémoire, alors le corbeau la mémorise, sinon il garde la précédente. De plus, les corbeaux vivent en groupe. A chaque itération, ils cherchent donc à se déplacer en direction d'un autre corbeau de manière aléatoire.

Les chercheurs ont obtenu les meilleurs résultats en utilisant 80 itérations ainsi que la fonction de coût ITSE. Cette fonction intègre le temps multiplié par le carré de l'erreur. Les résultats obtenus sont meilleurs que tous les autres algorithmes testés ou le réglage à la main.

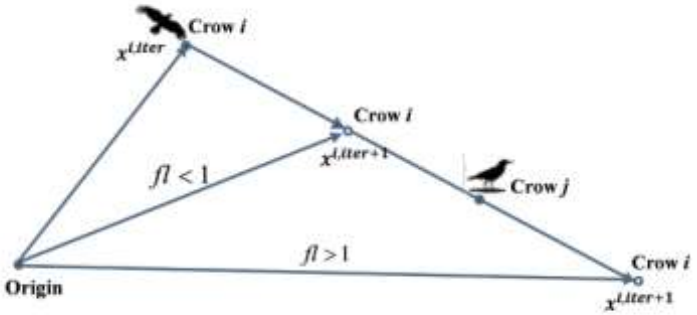


FIGURE 1. **The crow position update diagram in CSA case i.**

Ce projet constitue une première version d'une maquette pédagogique sur les correcteurs de type PID. De nombreuses améliorations sont possibles et pourront par exemple faire l'objet d'un projet de première année.

Afin d'augmenter la précision, il faudrait changer le couplage entre l'arbre de sortie du moteur et la tige filetée. L'arbre de sortie du moteur est actuellement de 4mm et le coupleur de 5mm. La tige filetée n'est donc pas centrée, ce qui induit un défaut périodique. L'autre extrémité de la tige filetée tourne librement dans un trou trop large. L'utilisation d'un roulement à billes permettrait de réduire le jeu.

Pour l'asservissement en position, d'anciennes pièces imprimées en 3D ont été récupérées et accrochées à l'écrou en laiton pour empêcher sa rotation. La conception d'une pièce adaptée à la maquette sera nécessaire pour réduire les jeux et faciliter la transition entre les asservissements en vitesse et en position.

Le club robotique de l'INSA CVL utilise des batteries de perceuse 18V type Makita pour alimenter en puissance les robots. Cette solution était donc simple à implémenter sur la maquette, mais ne convient pas forcément à un usage pédagogique. Le remplacement de la batterie par une alimentation de salle de TP pourra donc s'avérer nécessaire.

Pour protéger les usagers pendant la rotation du moteur, la fabrication d'un capot de protection sera sans doute à prévoir.

L'application en ligne de commande est fonctionnelle, cependant la réalisation d'une interface graphique permettra un accès plus facile aux différentes fonctionnalités (réglage des coefficients, affichage de courbes en temps réel, …).

Pour améliorer la maquette, on pourrait également réaliser l'implémentation d'un algorithme de recherche automatique des coefficients du PID et ajouter un capteur ultrason ou infrarouge pour vérifier les distances parcourues par l'asservissement en position.

IX. Code source et modèles 3D

L'ensemble du code du projet (Arduino et application PC) est disponible sur GitHub : https://github.com/DenisD3D/PIDControlModel

Les modèles 3D des pièces imprimés pour réaliser la maquette y sont également disponibles.