RAPPORT DE PROJET EN DÉMARCHE EXPÉRIMENTALE

L3 Sciences Pour l'Ingénieur parcours Électronique et Génie Informatique

Thomas CASCA
Lou LAURENT

Table des matières

Introduction

Ce rapport a pour objectif de résumer l'ensemble des connaissances transmises lors des cours de "Démarche Expérimentale". L'enseignement de cet UE se base sur l'approche d'un projet de robotique. Nous avons dû développer, pas à pas, différentes fonctionnalités afin de comprendre certaines notions élémentaires dans le domaine de l'électronique, la mécanique, l'étude et l'asservissement des systèmes et la programmation de micro-processeurs.

Ce rapport explique les processus et fonctionnalités, développés essentiellement en cours et au Fablab, qui se résument par :

- Le contrôle optimisé du moteur, qui peut être utile pour le déplacement d'un robot sur roue.
- La cartographie, qui sert à détecter les objets et obstacles. Avec un point important sur la notion de filtrage des données.

Pour réaliser ceci, nous utiliserons :

- Un environnement et une carte Arduino en utilisant principalement des fonctionnalités du microprocesseur ATMEL AT328P pour la partie informatique.
- Un moteur à courant continue à 2 encodeurs
- Un pont en H type L298N
- Un joystick
- Un capteur à ultrasons type HC-SR1
- Un servomoteur

Dans un premier temps, afin de pouvoir réaliser ces deux fonctionnalités, nous allons nous pencher sur la mesure de la vitesse angulaire. Cette mesure sera nécessaire pour tout le reste. On regardera également le montage électronique du moteur.

Dans un deuxième temps, nous étudierons la caractérisation du moteur. C'est-à-dire que nous allons déterminer les constantes régissant les équations électrique, mécanique et électromécanique du moteur.

Dans un troisième temps, nous allons nous concentrer sur la synthèse d'un correcteur PID. Au premier abord en le simulant afin d'extraire les paramètres. Enfin nous le réaliserons expérimentalement à la suite des simulations afin d'asservir au mieux notre moteur.

Dans un quatrième temps, nous verrons comment contrôler la vitesse de rotation ainsi que le sens de rotation du moteur à partir d'un joystick.

Dans un dernier temps, nous réaliserons la cartographie d'une enceinte à l'aide d'un capteur à ultrason et d'un servomoteur. Ce dernier point nous permettra également de voir la notion de filtrage et son utilité.

I. Mesure de la vitesse angulaire et montage du moteur

I.1. Mesure de la vitesse angulaire Ω du moteur

On utilisera les informations fournies par les deux encodeurs (PA et PB présentés en Figure 1) envoyées sur les pins 6 et 7 de la carte Arduino ainsi que les routines d'interruptions pour capturer la vitesse angulaire Ω (en rad/s) de notre moteur.

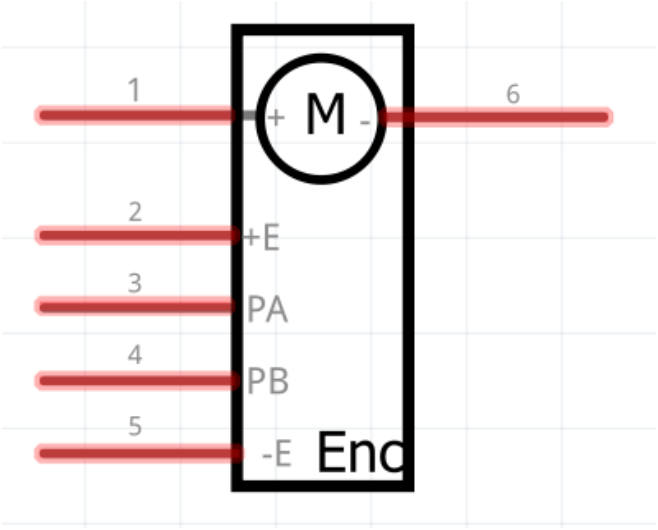

Figure 1 : Présentation des différents PINs du moteur

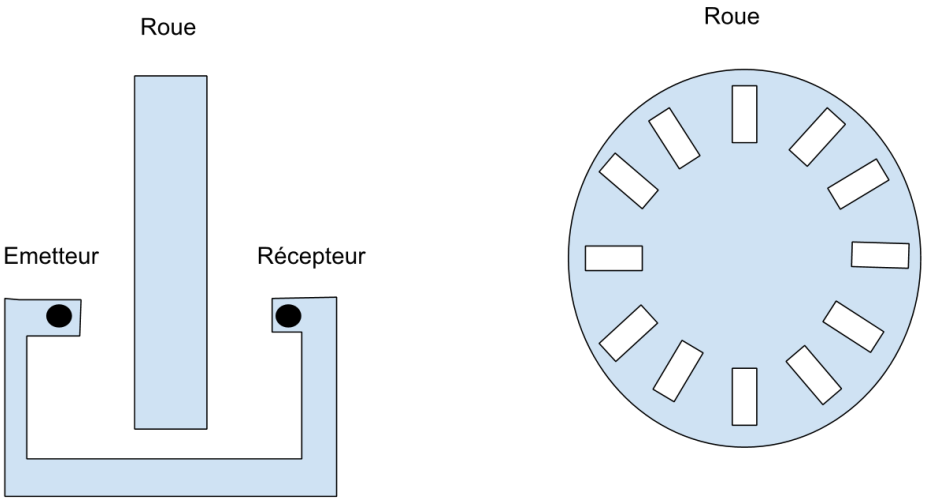

Figure 2 : Vue de profil et de face de la roue du moteur

L'encodeur envoie un signal à 5V lorsque l'émetteur envoie un signal au récepteur sinon, il envoie un signal de 0V (Figure 3). La roue possède des trous, lorsque qu'un trou passe devant le capteur, l'encodeur envoie un signal de 5V et 0V sinon.

On calcule la vitesse à partir de la formule suivante :

$$v = \frac{d\theta}{dt} = \frac{\Delta\theta}{\Delta t}$$

$\Delta\theta$ représente la différence en degré entre deux trous : cette différence est de 4°

Δt représente le temps en seconde parcouru entre deux trous : c'est le temps parcouru entre deux fronts montants (Figure 3).

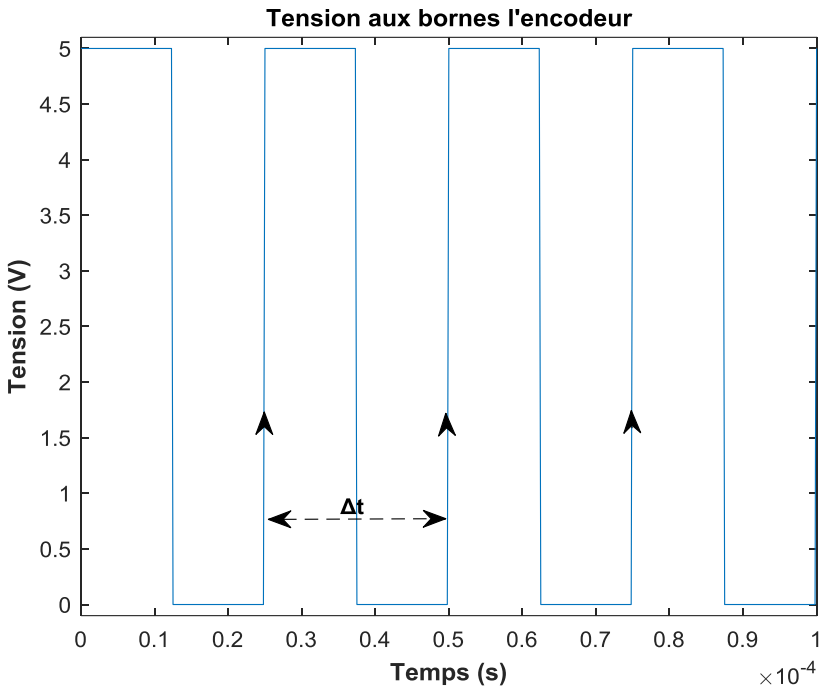

Figure 3 : Tension aux bornes de l'encodeur (flèches noires : front montant) avec Δt entre les fronts montants

Les deux encodeurs envoient la même information de manière décalée, ce qui nous permet de détecter le sens de rotation du moteur. Pour acquérir une information plus précise, on capture les temps Δt_1 et Δt_2 entre deux fronts montants pour les deux encodeurs et on moyenne ces deux temps.

Une fois les données mesurées et calculées *via* le programme, on obtient une vitesse angulaire v en deg/s. Pour obtenir une vitesse angulaire Ω en rad/s, on réalise la conversion suivante :

$$\Omega = v * \frac{2\pi}{360}$$

I.2. Montage du moteur

Pour réaliser nos mesures, nous avons monté le moteur suivant le montage présenté en Figure 4. Tout d'abord, le moteur est relié à un pont en H (puce L298N), représenté en rouge sur le schéma ; ce pont est branché à une alimentation stabilisée délivrant du 12V. Il est également connecté aux PINs de commande.

Pour commander le moteur, plus précisément sa vitesse, il faut envoyer un signal PWM (Pulse Width Modulation) sur la broche ENA du pont en H. Le signal est envoyé à partir de la PIN 11 de la carte Arduino. Le signal PWM permet de modifier le rapport cyclique d'un signal carré. Afin de le générer, on configure le Timer 2 associé à la PIN 11 et à partir du registre OCR2A, on fait varier le rapport cyclique. Les valeurs prises par le registre OCR2A varient de 0 à 2^8 ce qui correspond à des valeurs allant de 0 à 255 (c'est la taille du timer qui limite la plage de valeurs, ici le Timer 2 est sur 8 bits).

Le rapport cyclique α permet de gérer la tension U_{moy} à envoyer aux bornes du moteur. La valeur de la tension permet de faire varier la vitesse de rotation du moteur :

- Si $\alpha = 100\% \rightarrow U_{moy} = 5V$
- Si $\alpha = 0\% \rightarrow U_{moy} = 0V$

La Figure 5 montre un exemple de PWM avec un rapport cyclique $\alpha = 50\%$.

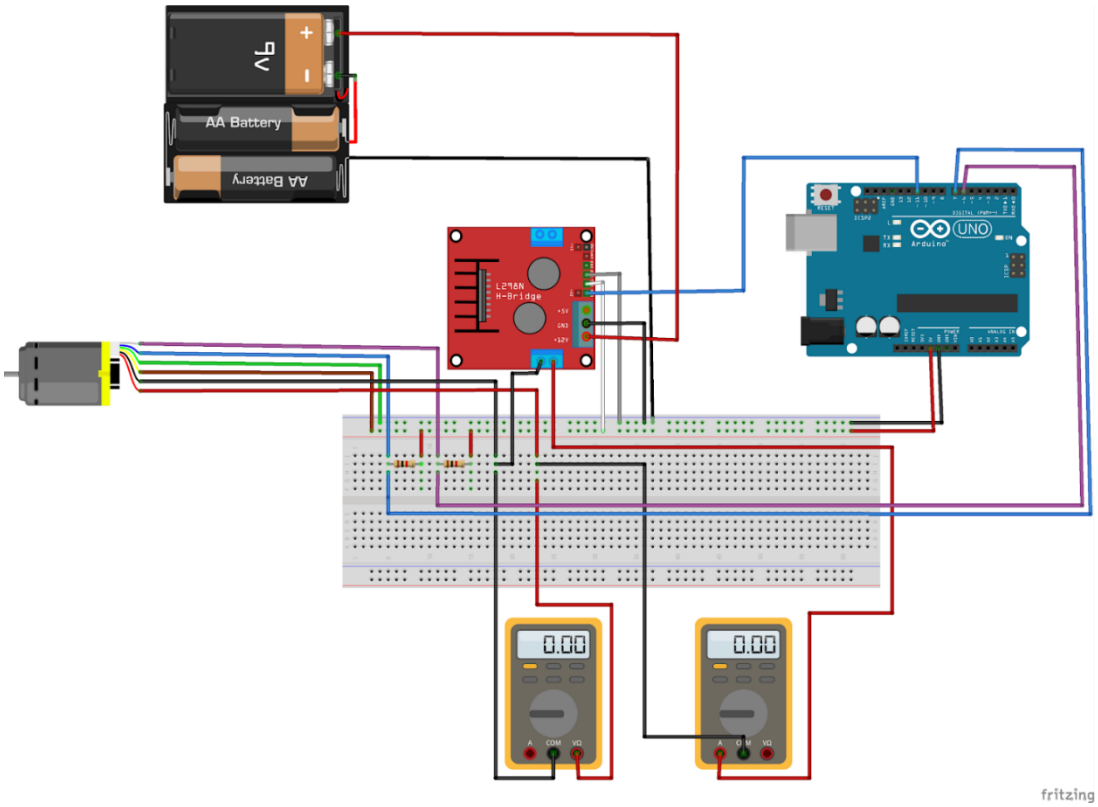

Figure 4 : Schéma complet du montage du moteur pour la prise des mesures

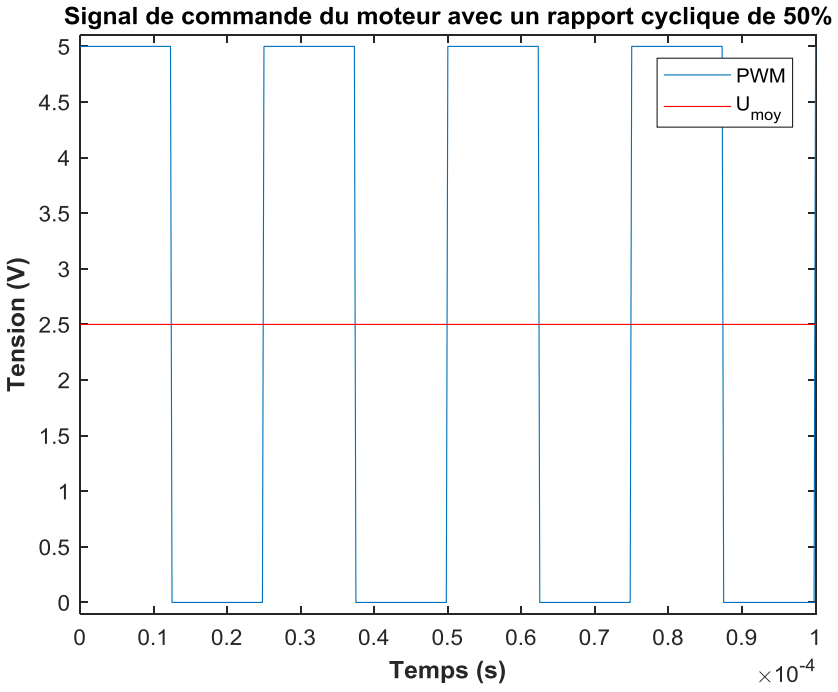

Figure 5 : Signal PWM avec un rapport cyclique de 1/2 et valeur moyenne de la tension

II. Caractérisation du moteur

La caractérisation du moteur permet de déterminer les constantes propres à chaque moteur et présentes dans les quatre équations régissant un moteur :

- L'équation électrique : $U = E + Ri + L\frac{di}{dt}$
- L'équation mécanique : $J\frac{d\Omega}{dt} = \Gamma_m - \Gamma_r - f\Omega$
- Les équations de couplage électromécanique :
  - $E = k\Omega$
  - $\Gamma_m = ki$

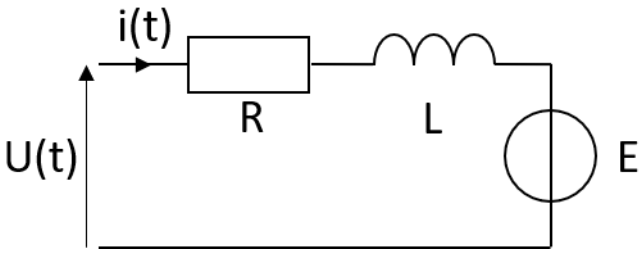

Figure 6 : Schéma électrique du moteur

Pendant les séances de projet, nous avons déterminé les constantes suivantes :

- La résistance électrique du moteur R (en Ω)
- La constante du moteur k (en J.A^{-1})
- Le couple de réaction Γ_r (en J)
- Le couple moteur Γ_m (en J)
- Le coefficient de frottement visqueux f (en J.s)
- Le moment d'inertie J (en kg.m²)

Pour retrouver ces constantes, le moteur est monté comme présenté sur la Figure 4.

II.1. Détermination de la résistance électrique du moteur

Intéressons-nous à l'équation électrique :

$$U = E + Ri + L\frac{di}{dt}$$

Pour déterminer la résistance du moteur, nous allons alimenter le moteur en 12V et utiliser un rapport cyclique de 1 pour la PWM afin d'avoir la vitesse angulaire maximale pour le moteur. Le moteur tourne à vitesse constante, le courant le parcourant est alors constant également. Il n'y a donc pas de variation de courant au sein du moteur.

Cela implique que le terme $L\frac{di}{dt} = 0$ car la dérivée du courant par rapport au temps est nulle. On obtient l'équation suivante :

$$U = E + Ri$$

Il nous reste alors à annuler E afin d'appliquer la loi d'Ohm est de trouver R. Utilisons l'une des équations de couplage électromécanique :

$$E = k\Omega$$

Le coefficient k n'est pas nul, il faut annuler la vitesse angulaire. En bloquant la roue, on annule Ω et E devient nulle. On obtient l'équation finale suivante :

$$U = R * i$$

On applique la loi d'Ohm pour obtenir R en mesurant le courant et la tension qui parcourent le moteur comme présenté sur la Figure 4. On obtient une résistance de :

$$\boxed{R = 5.1 \pm \Omega}$$

II.2. Détermination de la constante k du moteur

On mesure la tension E aux bornes du moteur pour différentes vitesses angulaires Ω. Pour faire varier la vitesse angulaire, on fait varier le rapport cyclique de la PWM. En prenant plusieurs valeurs, on obtient différentes valeurs de la constante du moteur k. Puis on détermine un k_{moy} qui correspond à la constante du moteur. Voici les résultats :

Tableau 1 : Valeurs de la tension E, de la vitesse angulaire Ω et de la constante du moteur k en fonction de la valeur de la PWM

PWM	E (V)	Ω (rad/s)	k (J.A^{-1})	PWM	E (V)	Ω (rad/s)	k (J.A^{-1})
255	10,47	50,615	0,207	217	9,47	45,379	0,209
252	10,45	50,615	0,206	215	9,2	43,633	0,211
248	10,44	50,615	0,206	210	8,8	41,888	0,210
242	10,45	50,615	0,206	207	8,6	41,015	0,210
240	10,45	50,527	0,207	205	8,4	40,143	0,209
237	10,43	50,440	0,207	200	7,9	37,699	0,210
235	10,353	49,742	0,208	195	7,5	36,303	0,207
233	10,32	49,393	0,209	190	7,08	34,034	0,208
230	10,2	48,869	0,209	185	6,7	32,114	0,209
227	10,15	48,520	0,209	180	6,2	29,322	0,211
225	9,9	48,346	0,205	175	5,8	27,925	0,208
223	9,8	47,124	0,208	170	5,6	25,133	0,223
220	9,6	46,077	0,208				

À partir des valeurs de k on calcule une constante du moteur moyenne :

$$\boxed{kmoy = 0.209 \pm J.A^{-1}}$$

II.3. Détermination du couple résistant Γ_r et des frottements visqueux f

Intéressons-nous à l'équation mécanique :

$$J\frac{d\Omega}{dt} = \Gamma_m - \Gamma_r - f\Omega$$

On se place à vitesse constante, l'accélération angulaire devient alors nulle, on obtient l'équation suivante :

$$\Gamma_m = \mathrm{f}(\Omega) = \Gamma_r + f\Omega$$

On prend différentes valeurs de vitesse angulaire, ce qui nous donne différentes valeurs de courant. D'après l'une des équations de couplage électromécanique :

$$\Gamma_m = ki$$

On obtient différentes valeurs de couple moteur. On peut alors tracer le couple moteur en fonction de la vitesse angulaire. Puis en faisant une régression linéaire à partir de notre nuage de points, on obtient l'équation suivante :

$$\Gamma_m = a * \Omega + b$$

avec $\boxed{a = f = 6.79 * 10^{-5}\ J.s}$ et $\boxed{b = \Gamma_r = 0.0206\ J}$

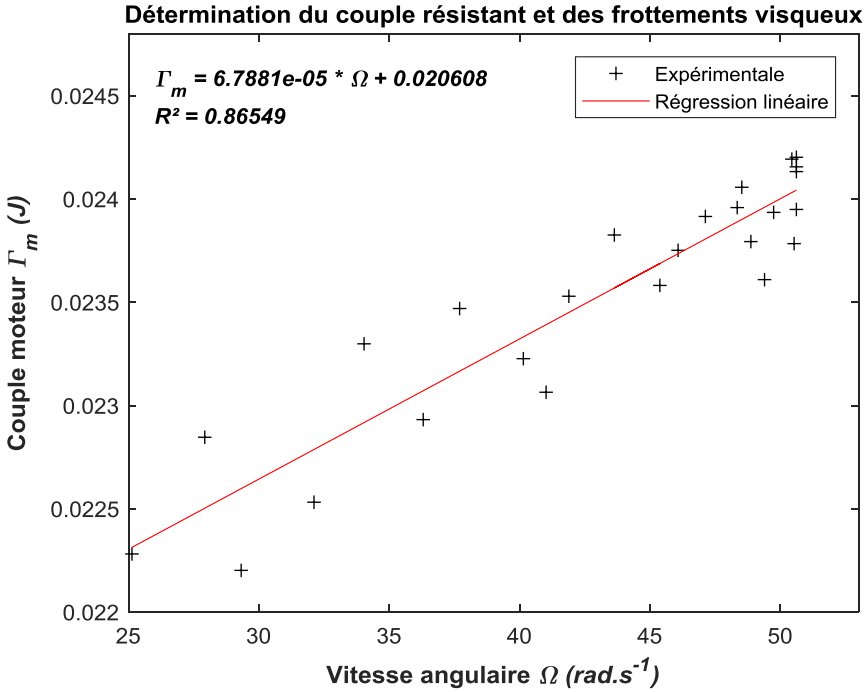

Figure 7 : Détermination de Γ_r et de f à partir d'une régression linéaire

A partir du coefficient de détermination R^2, on constate que la régression linéaire n'est pas satisfaisante. Cela est dû aux écarts importants entre les différentes valeurs du couple moteur. En effet, la mesure de la vitesse angulaire, de la tension et du courant aux bornes du moteurs sont sujettes à de brusques fluctuations de leurs valeurs. Ce qui complique la prise de mesure et apporte une source d'incertitude à nos mesures. Notre expérience est donc bruitée, cela va dans le sens du coefficient de détermination.

Les erreurs de mesures ne peuvent être réduites, de plus les valeurs obtenues pour le couple résistant et pour les frottements visqueux sont du même ordre de grandeur que les valeurs utilisées lors des simulations de notre moteur. On utilisera donc ces valeurs par la suite.

II.4. Détermination du moment d'inertie du moteur J

Pour pouvoir déterminer le moment d'inertie du moteur, il nous faut d'abord créer un signal carré de période $T = 4\ s$ afin d'avoir deux secondes à l'état haut et deux secondes à l'état bas. Après création de ce signal, on l'applique au moteur pour pouvoir mesurer son accélération (ou sa décélération). Après mesure de la vitesse angulaire en fonction du temps, on obtient le signal suivant :

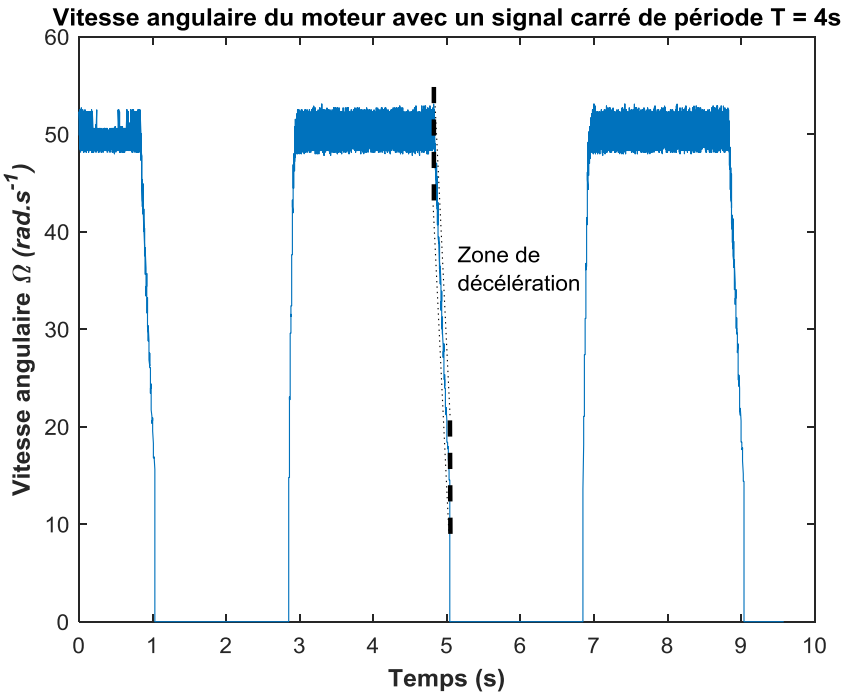

Figure 8 : Évolution de la vitesse angulaire pour un signal d'horloge de période $T = 4s$

On isole la zone d'accélération qui nous intéresse afin de pouvoir faire une régression linaire et obtenir la pente de la droite. Cette pente correspond à l'accélération angulaire $\frac{d\Omega}{dt}$. On obtient :

$$\frac{d\Omega}{dt} = 175.7\ rad.s^{-2}$$

A partir de l'équation mécanique du moteur, on détermine le moment d'inertie J du moteur :

$$\boxed{J = \frac{\Gamma_m - \Gamma_r - f\Omega}{\frac{d\Omega}{dt}} = 1.37 * 10^{-4}\ kg.m^2}$$

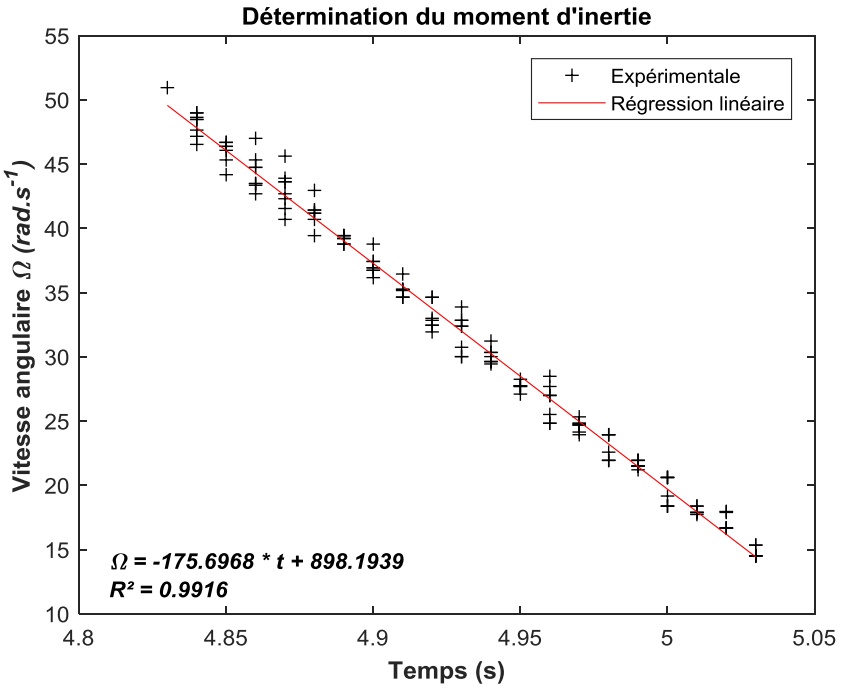

Figure 9 : Détermination du moment d'inertie à partir de la pente de la droite de décélération du moteur $\left(\frac{d\Omega}{dt}\right)$

On obtient alors les valeurs suivantes pour les paramètres du moteur :

Tableau 2 : Récapitulatif des valeurs obtenues pour les paramètres du moteur

Résistante électrique $R\ (\Omega)$	Constante du moteur $k\ (\mathrm{J.A^{-1}})$	Couple résistant $\Gamma_r\ (\mathrm{J})$	Frottements visqueux $f\ (\mathrm{J.s})$	Moment d'inertie $J\ (\mathrm{kg.m^2})$
5.1	0.209	0.0206	$6.79 * 10^{-5}$	$1.37 * 10^{-4}$

III. Correction du système

Dans cette partie, on se base sur la modélisation interne du moteur suivante :

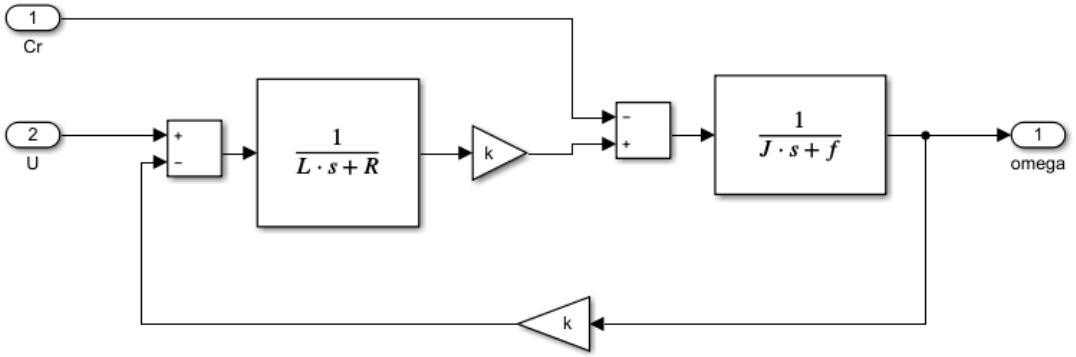

Figure 10 : Modèle SIMULINK de la fonction de transfert du moteur

III.1 Synthèse du correcteur PID

Cette modélisation représente une partie de notre système en boucle ouverte. Les valeurs des différents paramètres sont celles déterminées dans l'identification du moteur. On représente le système en intégralité et en boucle fermée comme suit :

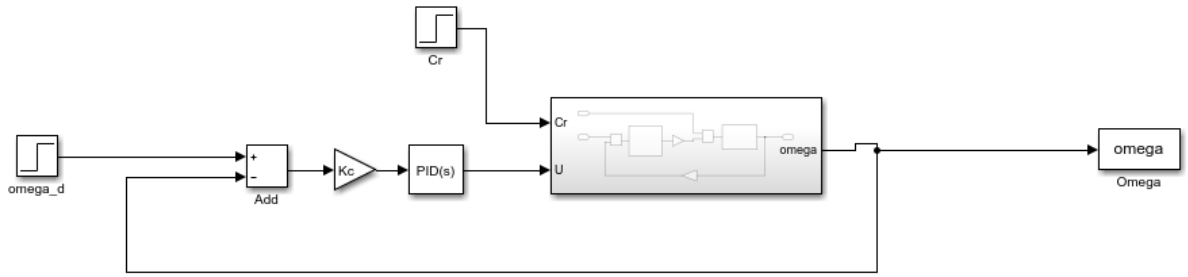

Figure 11 : Modèle SIMULINK de notre système

La modélisation interne (Figure 10) est représentée par le bloc grisé. On observe la présence d'un gain K_c, il permet de convertir la consigne *omega_d* (vitesse angulaire désirée) en une tension *U* entre 0V et 5V pour la carte Arduino. Il y a également la présence du bloc PID qui représente notre régulateur dont l'équation est la suivante :

$$K_p * \left(1 + K_i * \frac{1}{p} + K_d * \frac{p}{p+1}\right)$$

Pour trouver les valeurs adéquates du régulateur, on effectue plusieurs simulations en isolant les régulateurs P, PI et PID. Sans correction appliquée au système, on obtient la réponse indicielle suivante :

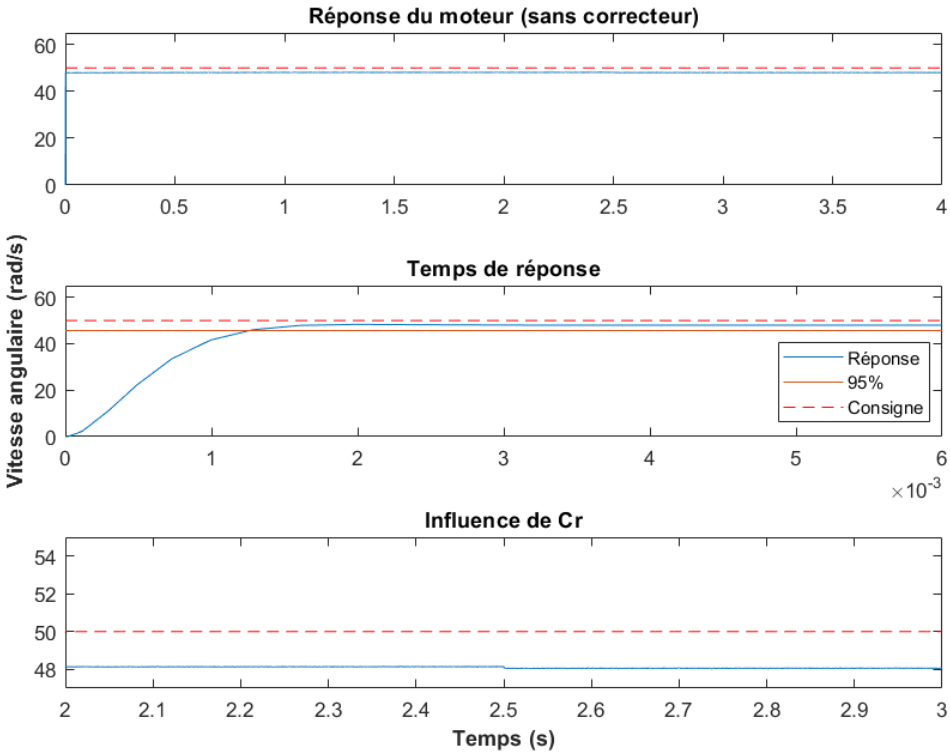

Figure 12 : Simulation de la réponse indicielle du moteur, temps de réponse et influence du couple résistant

À partir des courbes, on en déduit que :

- Le système est stable et ne présente pas de dépassement.
- Le temps de réponse est anormalement rapide.
- L'influence du couple résistant C_r est minime.
- L'erreur statique ε entre l'entrée (*omega_d*) et la sortie (*omega*) est inférieur à 2%

Malgré les bonnes performances du système, on souhaite annuler l'erreur statique. On obtient les résultats suivants pour le correcteur K_p :

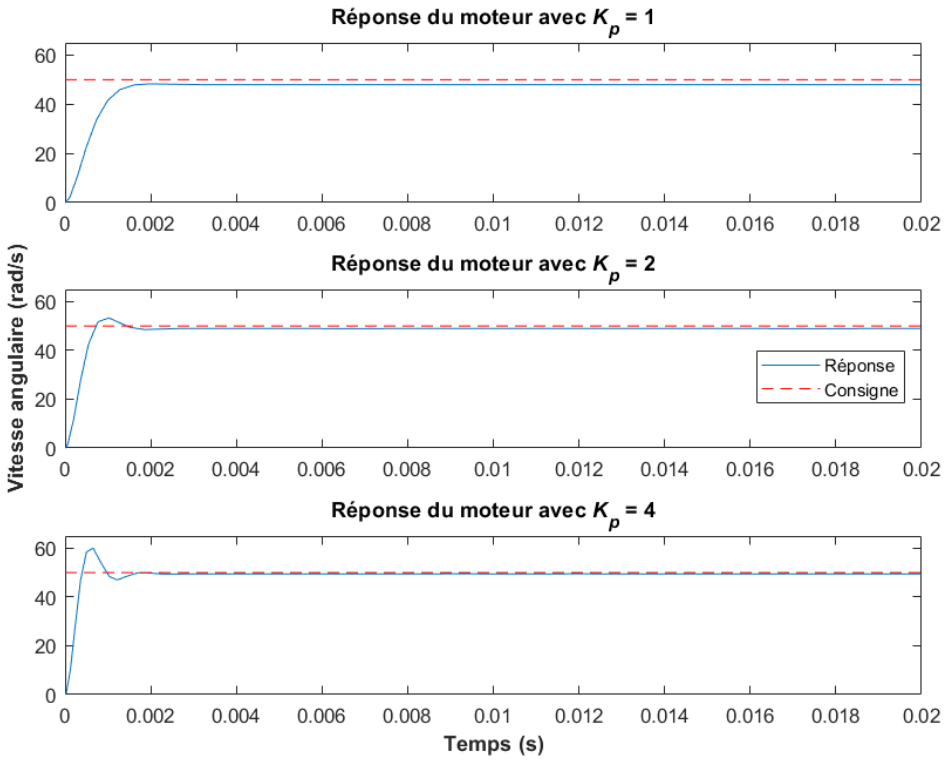

Figure 13 : Simulation de la réponse indicielle du moteur après une correction P (pour différents gain K_d)

On remarque que la valeur de K_p influe sur la stabilité du système (on peut voir un dépassement) mais également sur sa précision car l'erreur statique est réduite. Par la suite, on ajoute un intégrateur de gain K_i. Les simulations suivantes se font avec un gain $K_p = 4$. On obtient les réponses suivantes :

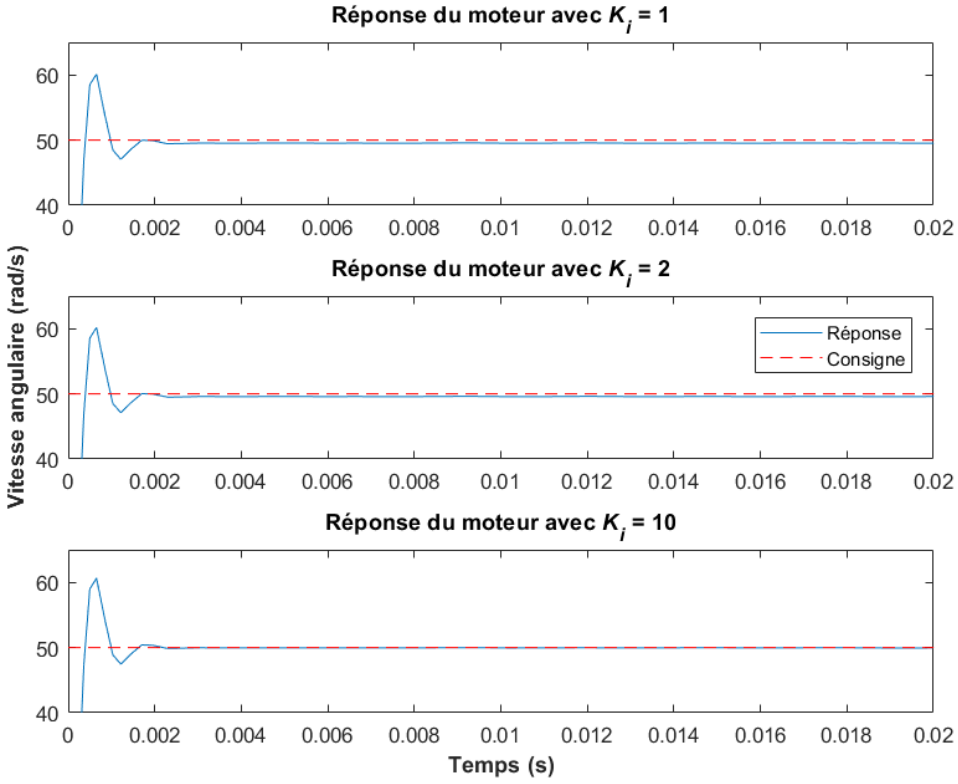

Figure 14 : Simulation de la réponse indicielle du moteur après une correction PI (différents gains K_i)

On constate que K_i n'a pas d'impact sur le dépassement mais il réduit au fil du temps l'erreur statique. On voit que plus ce gain est élevé plus l'erreur statique est annulée rapidement. Sachant que $K_p = 4$ et $K_i = 2$, on suit le même protocole pour le gain dérivateur K_d :

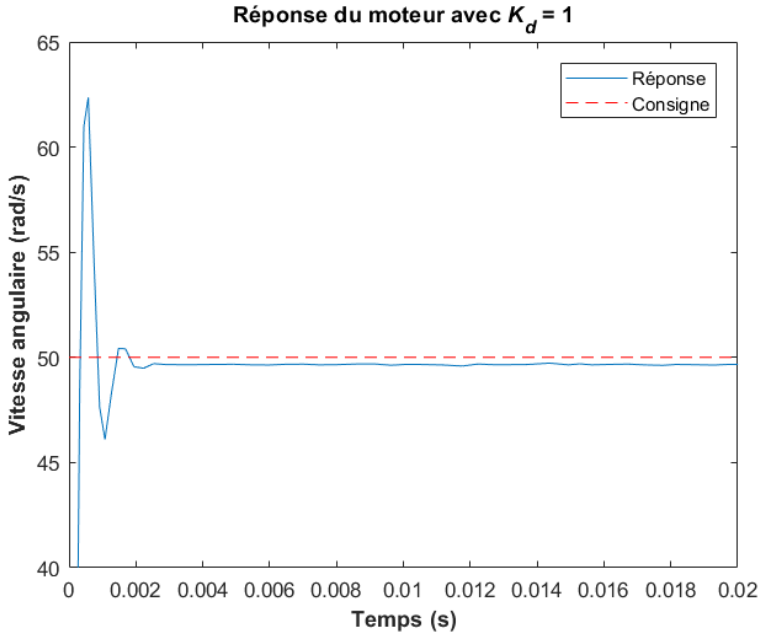

Figure 15 : Simulation de la réponse indicielle du moteur après une correction PID

Ajouter un gain dérivateur génère de l'instabilité sur notre système, on en conclut que ce gain est à négliger pour de bonnes performances. On se maintient donc à un régulateur PI. Le temps de réponse simulé n'est pas réaliste dans le cadre de ces simulations. Ce résultat est dû soit à la précision de nos mesures qui affecte directement les paramètres du moteur utilisés dans nos simulations soit à la négligence de certaines variables.

On réalise une dernière simulation à partir d'une fonction interne de MATLAB qui permet de calculer le PID « optimal » pour le système :

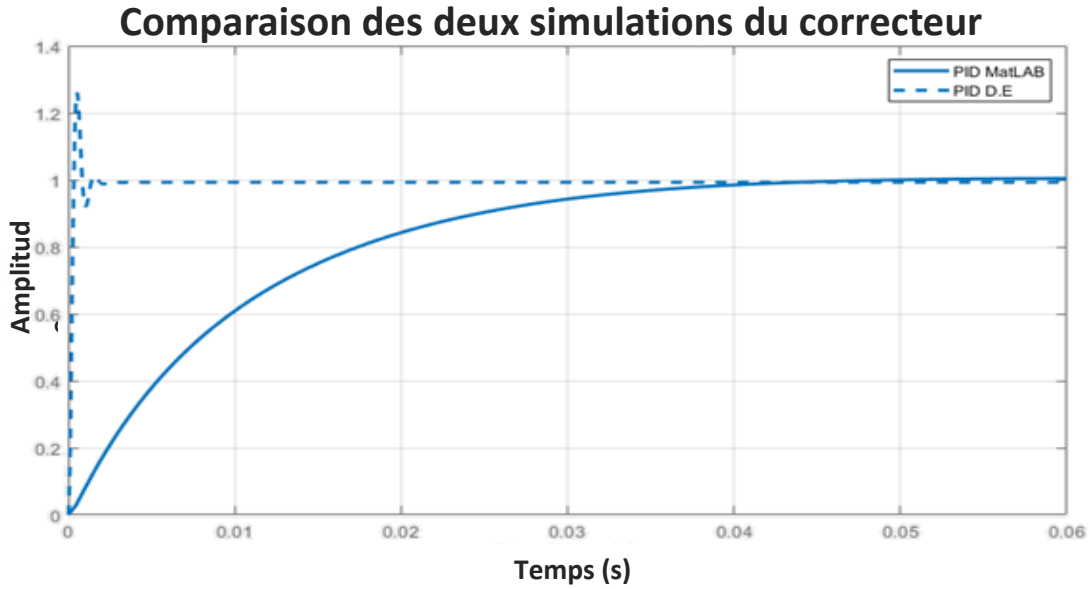

Figure 16 : Comparaison des correcteurs PID estimé par MATLAB et estimé par simulation

On distingue des performances plus réalistes du PID généré. Voici un tableau comparatif :

Tableau 3 : Comparatif du correcteur PID déterminé par MATLAB et du correcteur PID simulé

	Selon MATLAB	*Via* nos simulations
K_p	0.052	4
K_i	3.72	2
K_d	$5.20.10^{-4}$	1
Temps de montée (s)	0.024	0.00023
Temps de réponse à 95% (s)	0.038	0.0013
Dépassement (%)	0.44	26
Marge de phase (°)	90	42
Stabilité	Stable	Stable

III.2 Détermination expérimentale du correcteur PID

On souhaite maintenant trouver le bon régulateur PID, en réalisant un protocole expérimental qui consiste à, comme montré précédemment, capturé la réponse indicielle du moteur tout en faisant varier les gains K_p, K_i et K_d. On effectue la même méthode que précédemment, c'est-à-dire observer la réponse du moteur sans correction, puis avec un correcteur P, PI puis PID. Sans correction, on observe que la réponse du moteur est mauvaise, on a une erreur statique de 30%.

Pour réduire cette erreur, on souhaite rajouter un correcteur proportionnel P, on obtient les résultats suivants pour un gain $K_p = 8$:

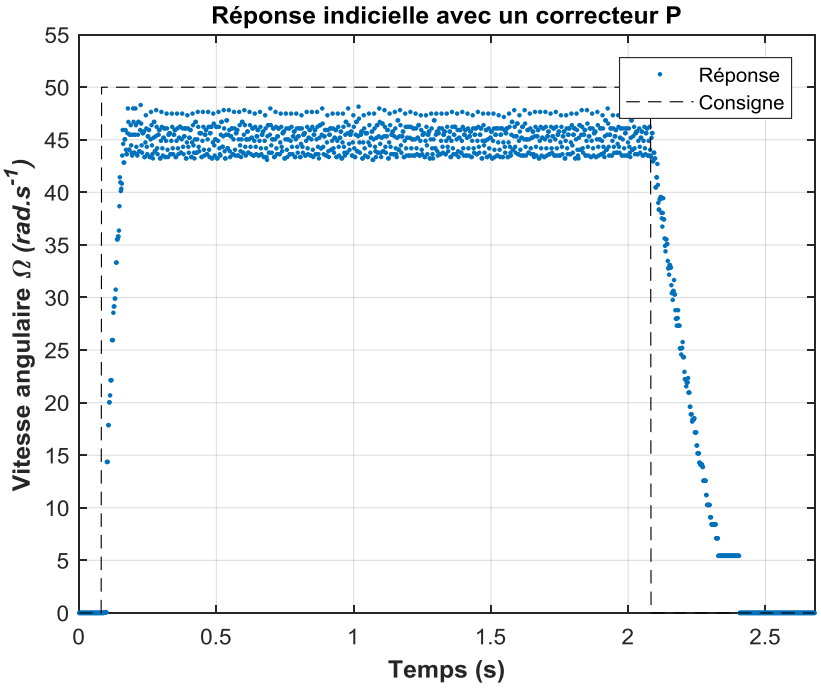

Figure 17 : Réponse indicielle de notre moteur avec un correcteur P (gain $K_p = 8$)

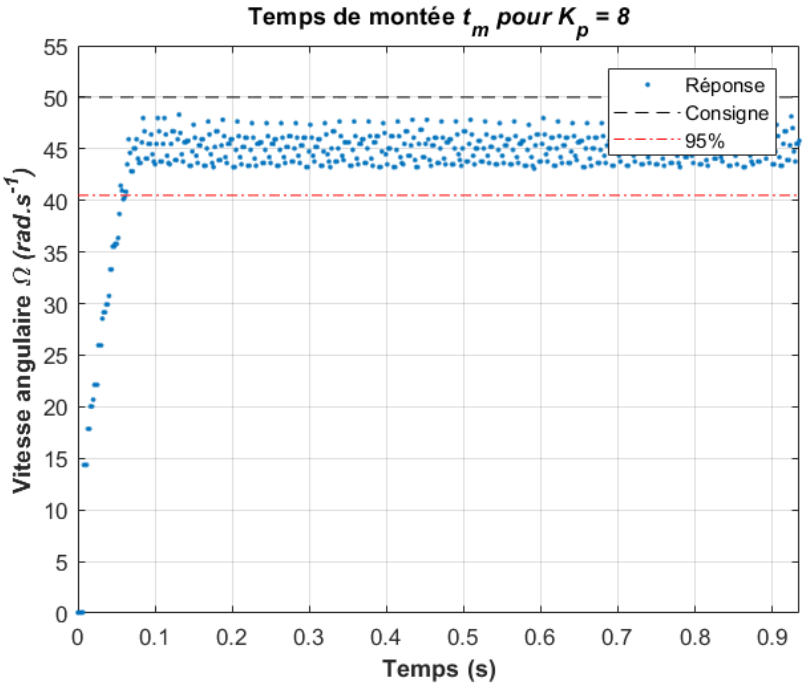

Figure 18 : Détermination du temps de montée à la suite d'une correction proportionnelle

On observe pour cette valeur que l'erreur statique est réduite mais elle est toujours présente (environ 10%). On voit cette différence entre la réponse et la consigne sur la Figure 18. Le temps de montée de cette réponse prend la valeur : $t_m = 0.06s$.

Lors de nos expérimentations, on remarque qu'augmenter la valeur de K_p réduit l'erreur statique. Si $K_p > 8$, l'erreur sera plus faible mais à certain moment la vitesse du moteur Ω dépasse la vitesse désirée Ω_d, on rentre donc dans le domaine de l'instabilité. Pour éviter ces évènements, on utilise des conditions de stabilités, pour que la vitesse du moteur ne dépasse pas un certain seuil. On effectue par la suite des simulations avec un correcteur PI en gardant $K_p = 8$ et en ajoutant $K_i = 2$:

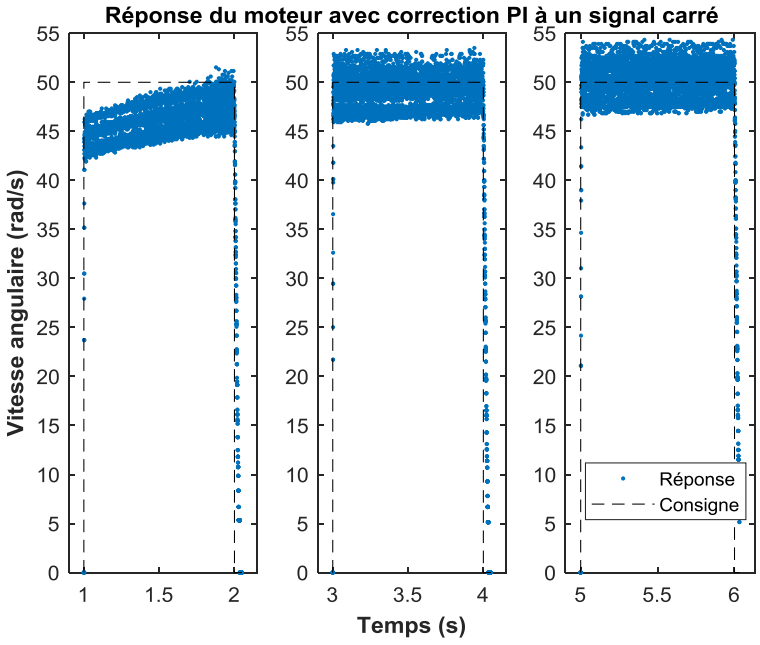

Figure 19 : Réponse indicielle du moteur à une consigne (signal carré de fréquence 1Hz) avec une correction PI

Ce qui est intéressant, lors de l'ajout de ce correcteur est qu'elle réduit l'erreur statique restante jusqu'à être quasiment nulle mais son application n'est pas immédiate. En effet, on observe que l'erreur statique diminue au fil du temps. Sur le graphique ci-dessus, on remarque que les correctifs sont appliqués au bout de 5 secondes. En effectuant des variations sur K_i, on s'est aperçu que sa valeur affecte le temps de correction.

Pour finir, en ajoutant un correcteur PID, nous n'avons pas observé de différence majeure sur la réponse de notre système. Cela est dû au temps de montée déjà très court de notre moteur. On constate que sur les simulations notamment l'estimation du meilleur PID par MATLAB que le gain du correcteur D est très faible. On se contente donc alors d'un correcteur PI pour notre système.

IV. Contrôle de la rotation du moteur

IV.1. Gestion du rapport cyclique

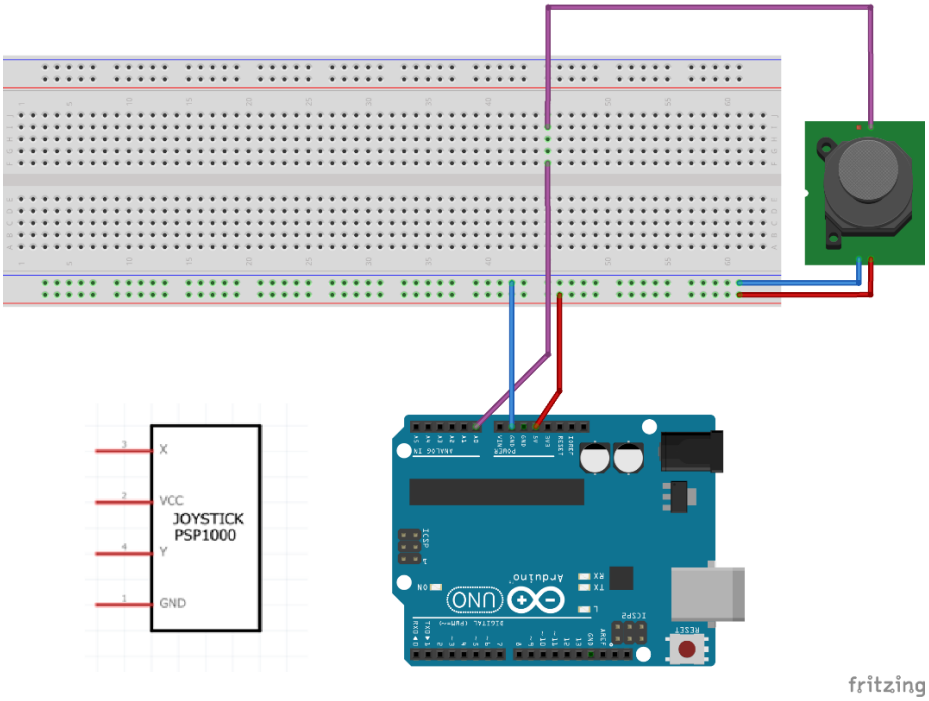

Figure 20 : Schéma de câblage du joystick et insertion dans le montage électronique

Le joystick doit être alimenté en 5V (par VCC et GND). Ce composant ressort deux valeurs X et Y. On utilise le Convertisseur Analogique Numérique (CAN) présent sur les PINs A0-A5 pour capturer les valeurs de X et Y. Dans le cas du schéma (Figure 20), on observe seulement la valeur de X (fil violet). On considère notre joystick comme un potentiomètre qui servira de pont diviseur de tension entre 0 et 5V pour la carte (au-dessus ou en-dessous il y'a risque de l'abimer). On utilise la fonction `analogRead()` pour capturer cette valeur. La fonction nous ressort en sortie une valeur comprise entre 0 et 1024 pour les valeurs équivalentes de 0 et 5V. On convertit cette valeur dans le registre OCRnX (avec n le numéro du TIMER, X prenant les valeurs A ou B) qui permet de configurer la valeur du rapport cyclique. Ce registre accepte les valeurs entre 0 et 2^n. On utilise également les registres nécessaires pour un signal PWM.

IV.2. Sens de rotation du moteur

On souhaite par la suite piloter le sens du moteur à partir du joystick. Pour réaliser ceci on a développé deux fonctionnalités :

- La détection du sens de rotation de la roue
- Le changement de sens de la roue

Pour la première fonction, on utilise le signal des deux encodeurs (voir Figure 20). On rappelle que les encodeurs reçoivent les mêmes signaux avec un décalage de phase. Ce décalage est utile pour les cas suivants :

- Si le signal 1 est en avance sur le signal 2 alors la roue tourne à droite
- Si le signal 2 est en avance sur le signal 1 alors la roue tourne à gauche

A partir de ces signaux, on observe, à l'aide de la routine d'interruption, le front montant des signaux 1 et 2 puis on les compare :

- Si lors du front montant du signal 1, sa valeur est différente à celle signal 2 alors le signal 1 est en avance.
- Si lors du front montant du signal 1, sa valeur est égale à celle signal 2 alors le signal 2 est en avance.

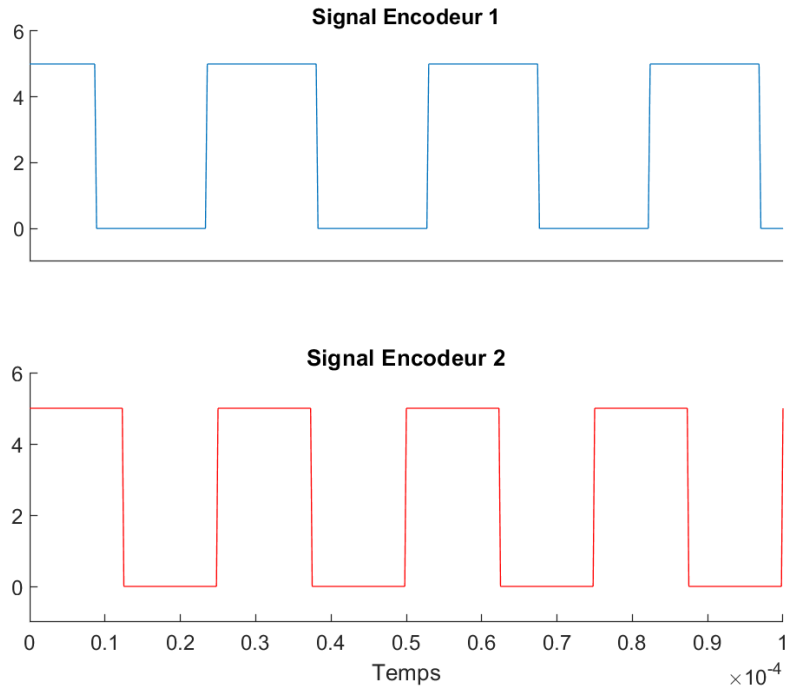

Figure 21 : Signaux des deux encodeurs (avec un déphasage entre les deux)

Pour la deuxième fonction, on utilise le même principe développé dans la partie précédente en ajoutant de nouvelles règles. Lorsque le joystick n'est pas utilisé, le moteur doit être à l'arrêt. Sur cette position, on observe, à l'aide de la commande `analogRead()`, que la valeur de $X \sim 512$. On souhaite donc que lorsque le joystick est orienté à gauche, équivalent à une valeur de $X < 512$, le moteur tourne à gauche et même principe pour la droite si la valeur de $X > 512$. Pour faire tourner le moteur dans un sens en particulier, il faut relier un signal continue de 5V et une masse respectivement sur les PINs IN1 et IN2 du pont diviseur (voir schéma puce L298N). En fonction de la valeur de X, on choisit sur quel PIN relié le 5V et le 0V.

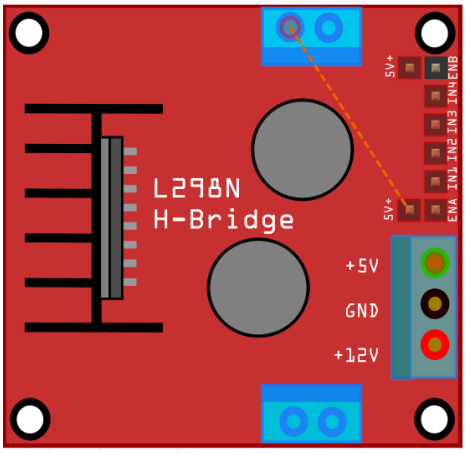

Figure 22 : Schéma de câblage de la puce L298N (pont en H)

La vitesse de rotation est pilotée grâce à un signal PWM relié à la broche ENA sur lequel on fait varier la valeur du rapport cyclique en fonction de la position du joystick. Pour faire varier correctement le rapport cyclique, il faut relier la valeur de X à OCRnX. On raisonne de la manière suivante pour le réaliser :

- Dans le cas où $0 < X < 512$, alors $0 < OCRnX < 255$ avec IN1 à 5V et IN2 à 0V pour le sens. On résume ce cas graphiquement :

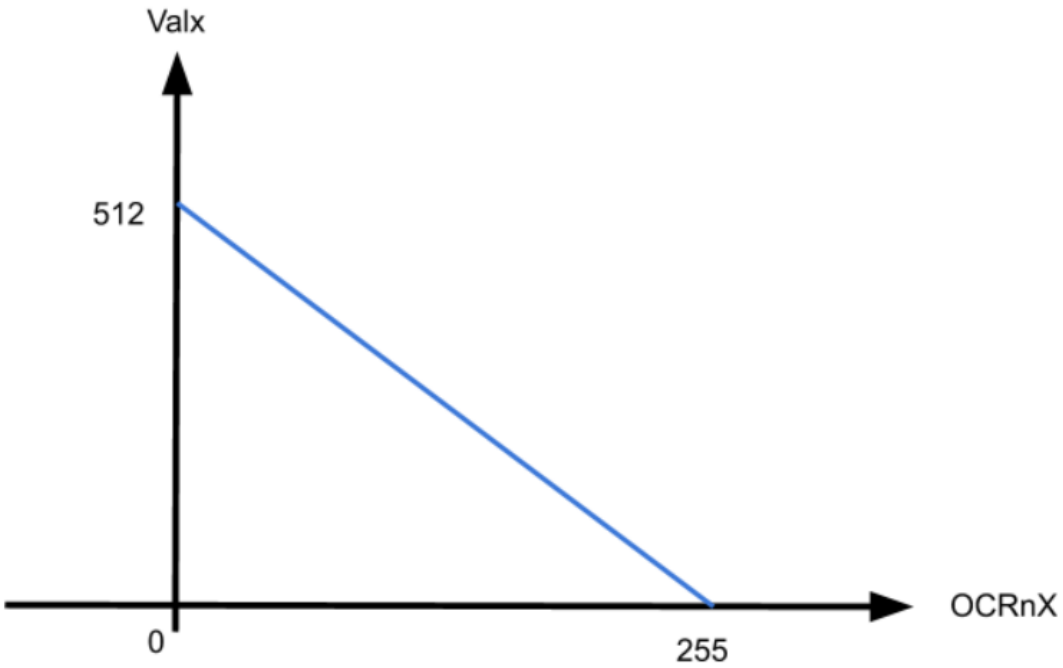

Figure 23 : Droite représentant le rapport en Valx et OCRnx (1er cas)

On obtient l'équation d'une droite telle que :

$$X = a * OCRnX + 512$$

Avec a le coefficient de la droite tel que $a = \frac{\Delta y}{\Delta x} = \frac{512}{-255} \approx -2$

On conclut que : $OCRnX = \frac{X-512}{-2}$

- Dans le cas où $512 < X < 1024$, alors $0 < OCRnX < 255$ avec IN1 à 0V et IN2 à 5V pour le sens. On résume ce cas graphiquement :

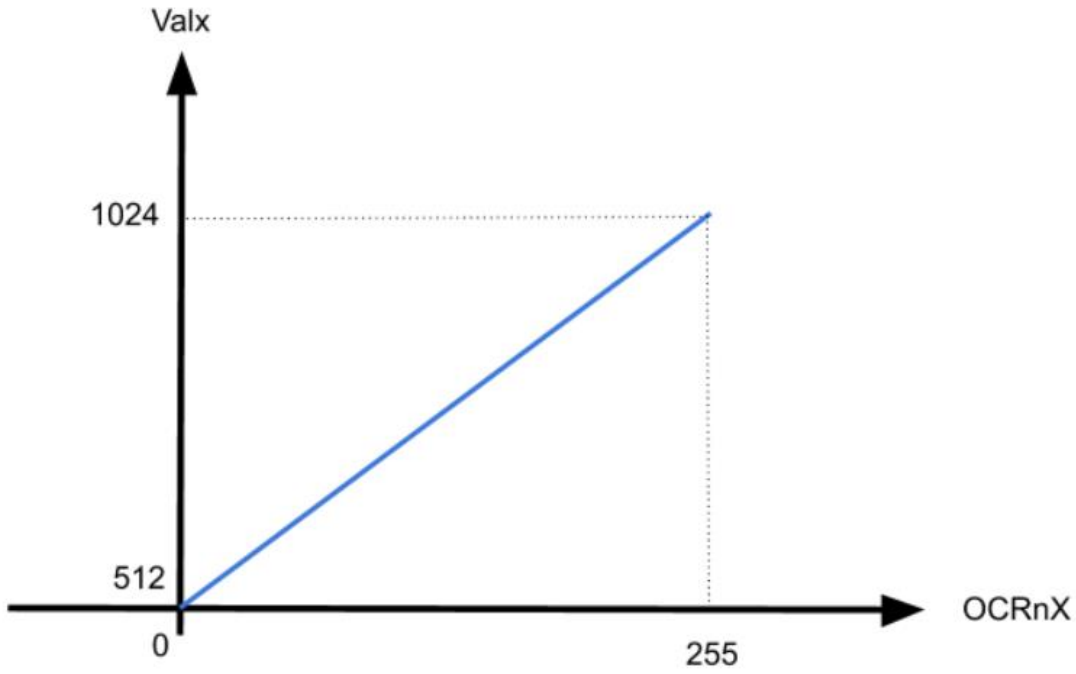

Figure 24 : Droite représentant le lien entre Valx et OCRnX (2ème cas)

On obtient l'équation d'une droite telle que :

$$X = a * OCRnX + 512$$

Avec a le coefficient de la droite tel que $a = \frac{\Delta y}{\Delta x} = \frac{1024-512}{255} \approx 2$

On conclut que : $OCRnX = \frac{X-512}{2}$

On transpose ce raisonnement dans un algorithme ce qui nous permet de contrôler le sens de rotation ainsi que la vitesse angulaire du moteur en fonction de la position du joystick.

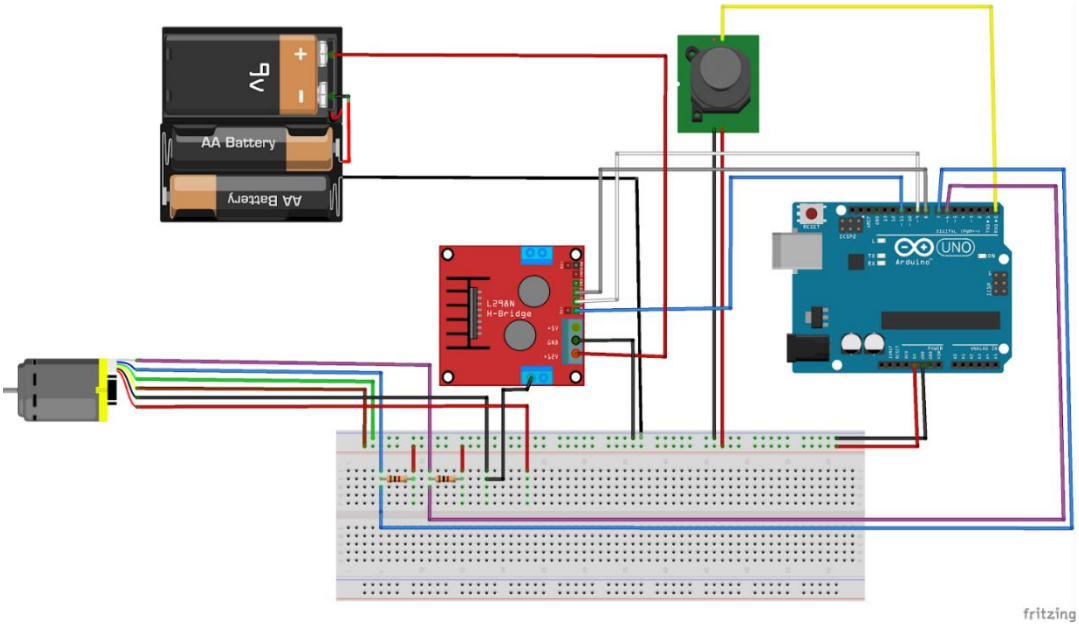

Figure 25 : Schéma de câblage du moteur, du joystick, du pont en H, de l'alimentation et de la carte Arduino

V. Cartographie

On a réalisé une cartographie à partir des registres ATMEL. Les composantes principales pour cette fonctionnalité sont le capteur à ultrasons ainsi qu'un servomoteur. Nous allons voir sur les prochains paragraphes comment nous les avons utilisés pour réaliser une carte des objet environnants. Nous conclurons également sur l'utilité de cette fonctionnalité pour un projet de robotique comme le nôtre.

V.1. Utilisation du capteur à ultrasons

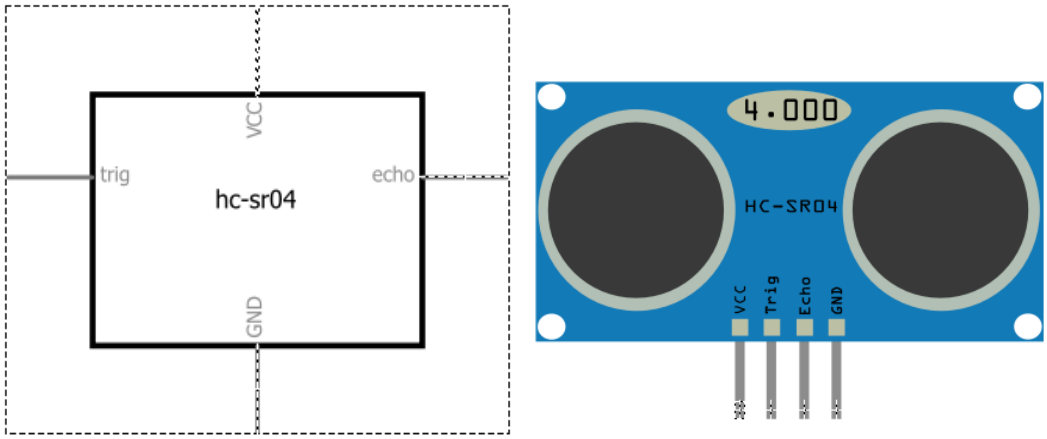

Figure 26 : Schéma de câblage du capteur à ultrasons HC-SR04

Ce capteur fonctionne à l'aide des broches "Trigger" et "Echo" (voir schéma capteur). On envoie une impulsion de 5V sur la broche "Trigger", en réponse le capteur émet une onde sonore sur une distance de 4m. Si l'onde rencontre un objet, elle a pour but d'être refléter vers la réception du capteur, connecté à "Echo". Echo va fournir un signal en fonction du temps d'émission-réception de l'onde. Et à partir de ce temps, on effectue un calcul de conversion (obtenue à l'aide de la documentation) pour obtenir la distance entre l'objet et le capteur. À l'aide du capteur, on peut effectuer une capture de plusieurs points mais pour obtenir une carte, il faut utiliser un servomoteur afin d'obtenir plusieurs points à différents angles.

V.2. Utilisation du servomoteur

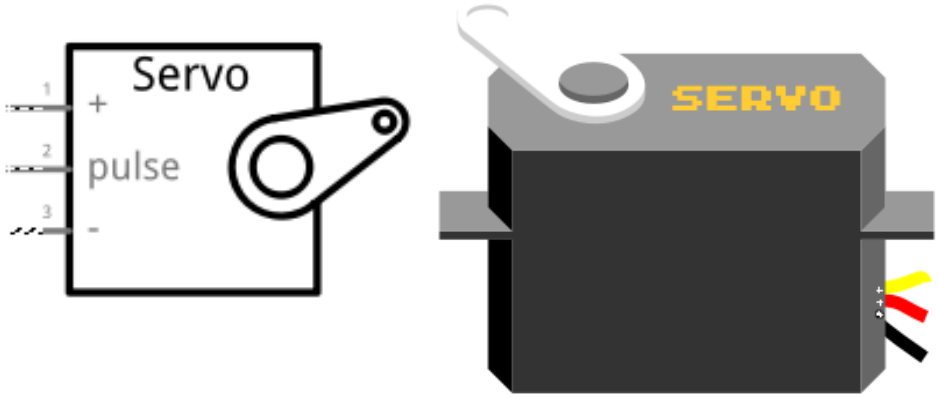

Figure 27 : Schéma de câblage du servomoteur

Pour effectuer notre carte, il faut que notre servomoteur tourne, sur un plan référentiel, entre 0 et 180°. Pour effectuer cette fonction, il faut improviser un signal type PWM en se basant sur un Timer et l'envoyer sur la broche "pulse" du servomoteur. En fonction de notre impulsion, on peut régler le servomoteur sur un angle précis. Ci-dessous vous pouvez voir deux types d'impulsion pour deux angles donnés en fonction du servomoteur utilisé en démarche expérimentale (Pas les mêmes pour chaque servomoteur).

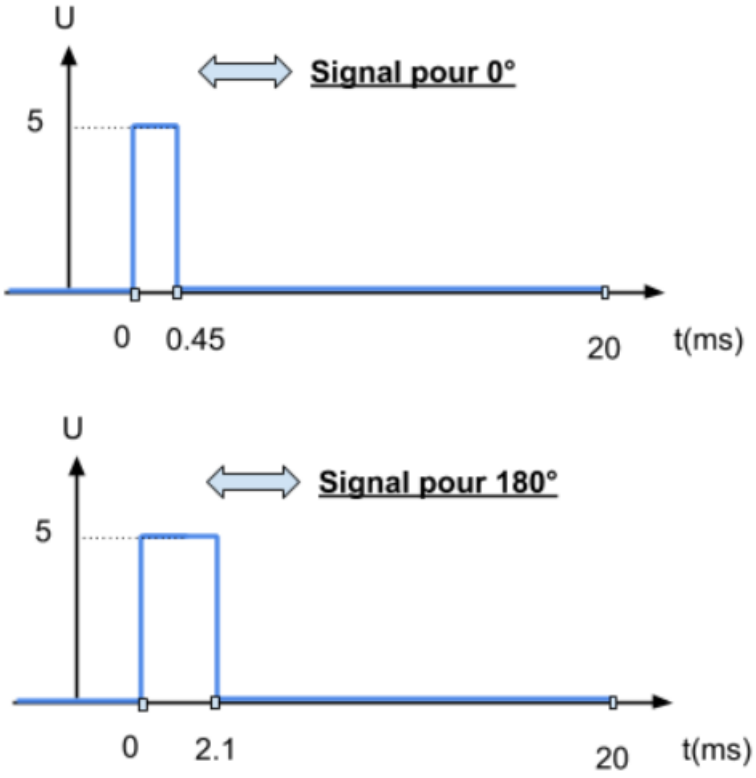

Figure 28 : Impulsion pour obtenir un angle de 0° et un angle de 180°

V.3. Capture des données et filtrage

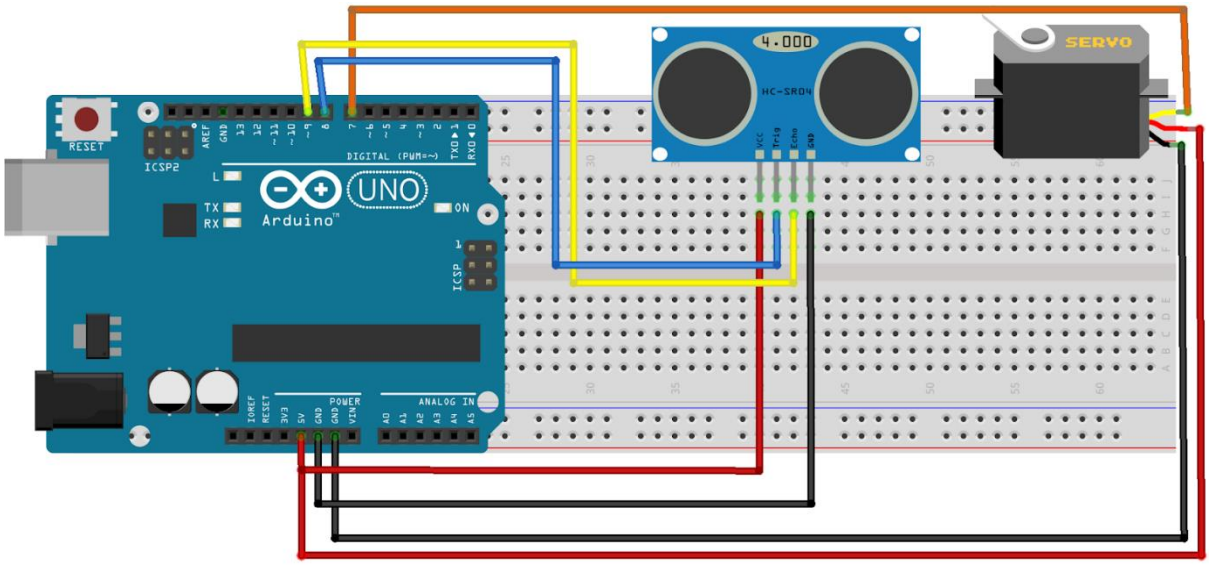

Figure 29 : Schéma de câblage du capteur à ultrasons, du servomoteur et de la carte Arduino

V.3.1. Capture des données

On peut par la suite fixer notre capteur sur le moteur ce qui nous permet de connaître la distance et l'angle de chaque point capturé. On peut déduire la position des points (x,y) grâce au règle de trigonométrie tel que :

$$distance = d\ (cm) = \sqrt{x^2 + y^2}$$
$$angle\ servomoteur\ (rad) = \theta$$
$$x = d * \cos\theta$$
$$y = d * \sin\theta$$

En capturant plusieurs distances et angles à l'aide du logiciel CoolTerm, on peut créer une cartographie des lieux :

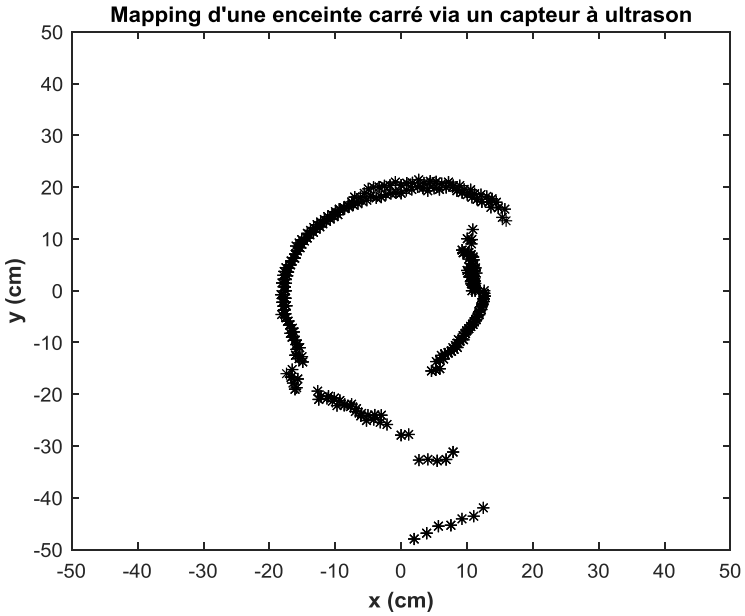

Figure 30 : Cartographique d'une enceinte carrée à partir d'un capteur à ultrason monté sur un servomoteur (sans filtrage)

On obtient une carte avec beaucoup d'imprécision due au capteur, car certaines ondes émises reflètent sur l'objet mais ne sont pas perçue directement par le récepteur ce qui allonge le temps d'émission-réception et augmente la valeur de la distance, la valeur est donc faussée. Pour pallier ce problème, on réalise un filtrage de nos données.

V.3.2. Filtrage

L'objectif ici est de filtrer nos données afin de limiter les nombres de valeurs aberrantes voire de les supprimer totalement. Cela permettra également de réduire le bruit de mesure. On va comparer trois méthodes de filtrage :

- Un filtre moyenneur à fenêtre glissante : c'est une méthode très simple mais qui a pour inconvénient d'écrêter énormément les pics.
- Un filtre médian à fenêtre glissante : plus performant que le filtre moyenneur car il limite l'impact des valeurs extrêmes dans nos données.
- Un filtre de Savitzky-Golay : plus performant encore car il permet de filtrer dans notre intervalle en effectuant une régression (linéaire ou non) correspondant le plus à nos valeurs. On cherche donc à minimiser l'erreur au sens des moindres carrés.

Dans le cas du filtre de Savitzky-Golay, il est nécessaire de préciser le degré du polynôme de la régression. Dans notre cas, on effectuera le filtrage avec un polynôme de degré 1.
Il faut également préciser la taille de la fenêtre utilisée lors du filtrage. Ici, nous avons utilisé différentes tailles de fenêtre : 11 points, 21, points, 31 points, 41 points, 51 points et 61 points. Cela nous permet de voir l'influence de la taille de la fenêtre sur le lissage de la courbe et l'écrêtage des pics.
L'ensemble du traitement de nos signaux est fait sur MATLAB, on utilise donc les fonctions internes au logiciel (**mean**, **median** et **sgolayfilt**). La comparaison des différents filtres et de la taille de la fenêtre se fait sur la réponse de notre moteur à un signal sinusoïdal présenté dans la figure suivante :

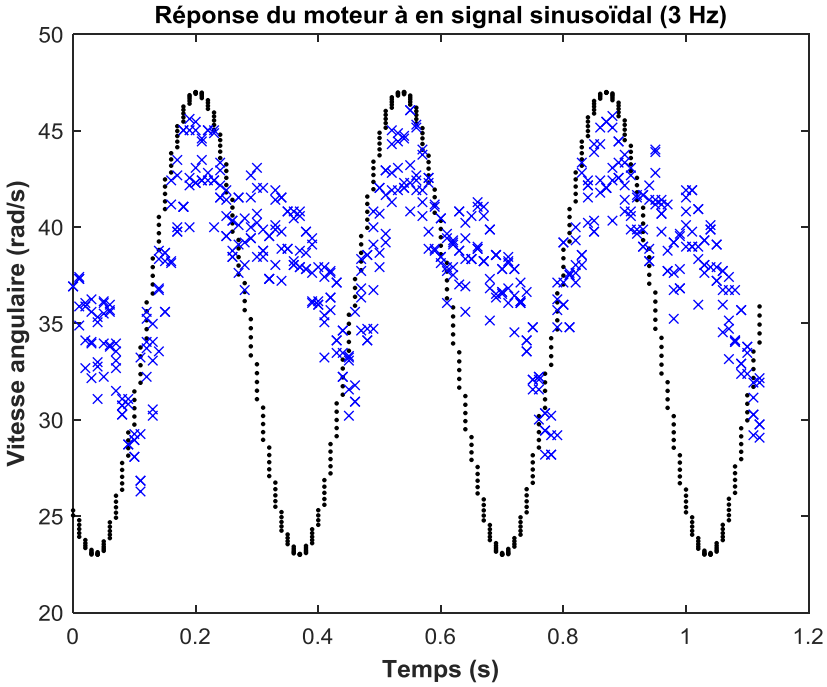

Figure 31 : Réponse du moteur à un signal sinusoïdale de 3Hz (sans filtrage)

Pour commencer, on va voir l'influence de la taille de la fenêtre sur le filtrage de notre signal. On sait que plus la fenêtre est grande et plus le signal filtré sera lissé avec un fort écrêtage des pics. Voici un exemple avec le filtre moyenneur à fenêtre glissante :

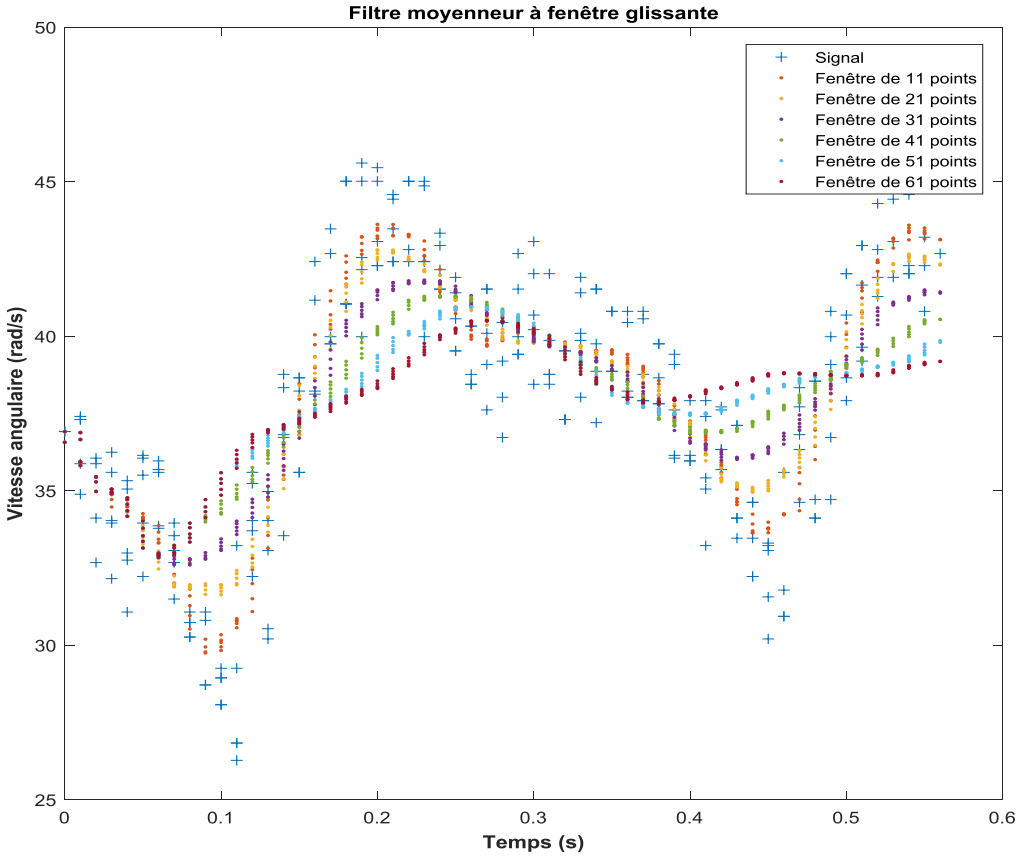

Figure 32 : Étude de l'influence de la largeur de fenêtre pour un filtre moyenneur à fenêtre glissante

On constate bien ce qui a été dit précédemment, et surtout que plus la fenêtre est large et moins le signal filtré correspond au signal mesuré. Dans notre cas, on constate que la fenêtre de 11 points est suffisamment performante pour enlever les valeurs aberrantes et que le signal filtré correspond à celui mesuré. Maintenant que la largeur de fenêtre est déterminée (11 points), on peut comparer les différents filtres :

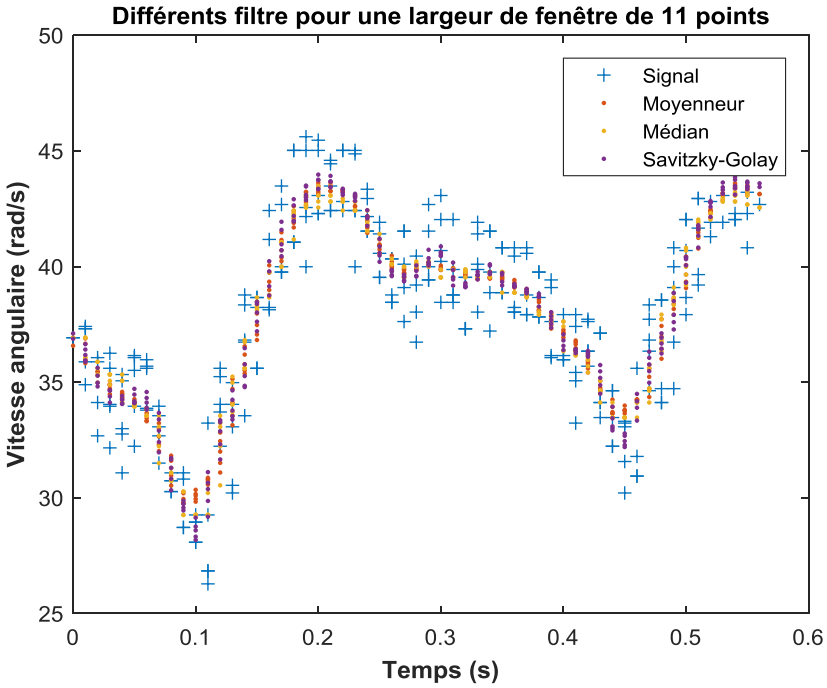

Figure 33 : Comparaison de l'efficacité de trois filtres sur notre réponse

On voit que le filtre de Savitzky-Golay est meilleur que le moyenneur étant donné que le moyenneur correspond *grosso modo* à un filtre de Savitzky-Golay de degré 0. Il est également meilleur que le médian qui a tendance à trop écrêter le signal lorsque le moteur atteint une vitesse trop importante. Cependant le moyenneur reste bon et il a l'avantage d'être simple à mettre en œuvre tant informatiquement qu'électroniquement. On va donc préférer ce filtre aux deux autres.
On peut également appliquer ce filtre à notre carte où certaines valeurs sont aberrantes dues à une mauvaise réflexion de l'onde sonore. On obtient la carte suivante :

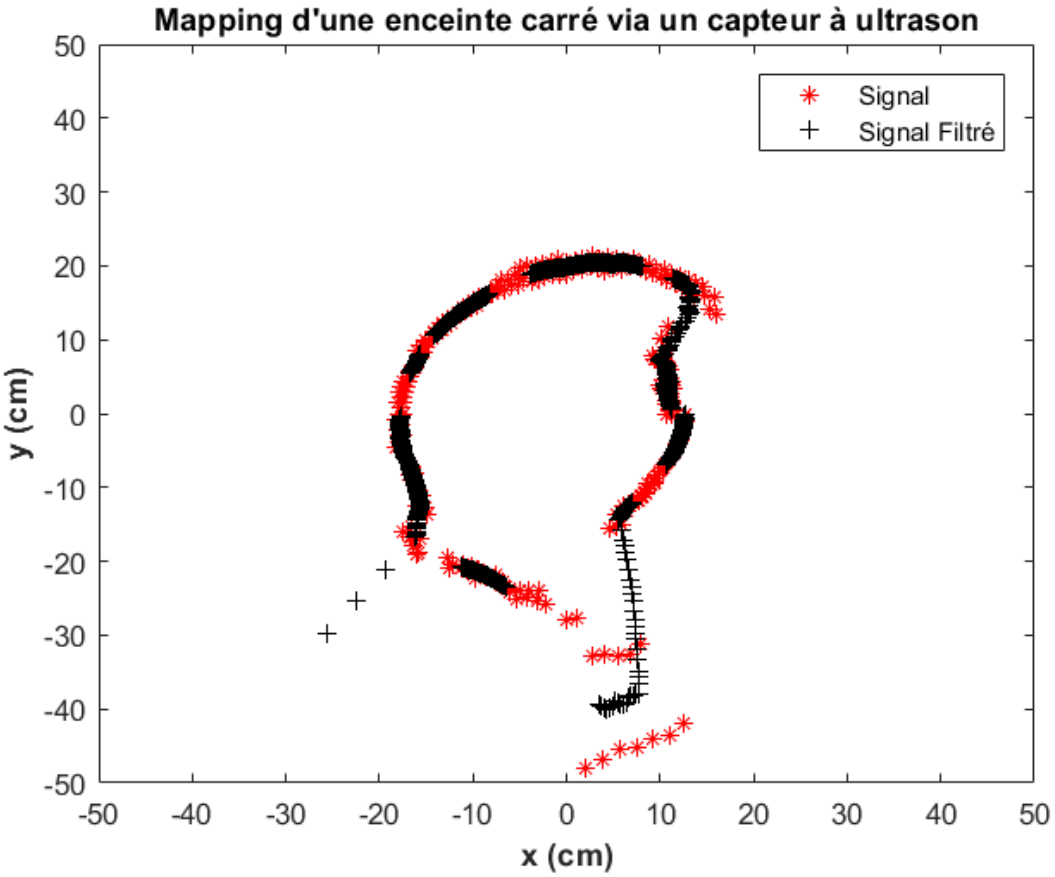

Figure 34 : Comparaison entre la cartographie non filtrée (en rouge) et filtrée (en noire)

On constate une forme plus régulière au niveau de la cartographie. Certaines valeurs apparaissent sur le signal filtré alors qu'elles n'étaient pas présentes sur le signal brut. On voit ici l'utilité du filtre qui permet de supprimer les valeurs extrêmes aberrantes.

Conclusion

Lors de ce projet, nous avons mis en œuvre une importante partie des connaissance vues lors de l'année de licence. Au travers de celui-ci, nous avons œuvré dans différents domaines tels que l'électronique, la programmation notamment d'un microcontrôleur, la robotique, le traitement du signal, la gestion de projet, le travail en autonomie et en groupe, etc.

La première mission a été de réussir à mesurer la vitesse de rotation du moteur. Pour cela, nous avons appris à réaliser des timers, des interruptions et des PWM. On a utilisé un pont en H dont le fonctionnement a été vu lors d'un enseignement du semestre 3. Une fois que les notions de timer, d'interruption et de PWM ont été intégrer, nous avons câbler notre montage avec le moteur afin de capturer la vitesse de rotation.

La deuxième mission a été de déterminer les paramètres du moteur. Pour ce faire, nous avons mis en place différents protocoles de mesure permettant de déterminer la résistance électrique interne, la constante du moteur, le couple résistant, les frottements secs et le moment d'inertie. Ces différents paramètres sont utiles pour la simulation du moteur.

La troisième mission a été de corriger notre système à partir d'un correcteur PID. Nous avons d'abord réalisé des simulations sous MATLAB de notre système afin d'en extraire les différents gains du correcteur. Puis nous avons réalisé expérimentalement le correcteur afin d'obtenir les vrais paramètres du correcteur et asservir au mieux notre moteur.

La quatrième mission a été de contrôler le sens de rotation du moteur ainsi que sa vitesse angulaire. Pour cela la notion de PWM a fortement été utilisée. Pour le contrôle, nous avons utilisé un joystick permettant de changer le sens de rotation en fonction du sens d'orientation du joystick. La vitesse de rotation est contrôlée par le degré d'inclinaison du joystick.

La cinquième et dernière mission a été de réaliser une cartographie d'une enceinte à l'aide d'un capteur à ultrason et d'un servomoteur. Il a fallu mesurer une distance à l'aide du capteur à ultrason ainsi qu'un angle à l'aide du servomoteur. Enfin une fois la cartographie réalisée, nous avons effectué un filtrage de nos données afin de se familiariser avec cette notion. Nous avons vu trois filtres différents (moyenneur, médian et Savitzky-Golay), l'influence de la largeur de fenêtre et comparer les performances de ces trois filtres.

Il resterait plusieurs points à réaliser pour finaliser le robot, notamment la gestion des deux roues du robots pour permettre à ce dernier de tourner, la détection d'objet pour éviter les collisions.

En conclusion, au travers du cadre transversal de cette matière, ce projet nous a permis de mettre en œuvre l'ensemble des connaissances acquises en licence et également d'en acquérir des nouvelles.

Table des illustrations