Introduction

Le but de ce document est de rassembler les informations nécessaires au développement du code informatique de la partie *Déplacement* du projet intégré.

La première partie de ce document contient les informations techniques concernant les périphériques d'interface entre le microcontrôleur et le robot.

La seconde partie décrit le développement du régulateur de position à implémenter.

Informations techniques

Le dsPIC Déplacement

Le dsPIC *Déplacement* est un dsPIC33FJ128MC804. C'est un processeur spécialisé dans le contrôle de moteur. Il possède 2 périphériques d'interface pour encodeur en quadrature et un périphérique *Output Compare* capable de générer 4 PWM ; il dispose donc de tout ce dont nous avons besoin pour contrôler simultanément 2 moteurs.

Remarque : le dsPIC33FJ128MC804 possède également un périphérique appelé *motor control PWM module*, qui permet de générer des formes de PWM complexes. N'ayant besoin que d'une "simple" PWM par moteur, nous utiliserons le *Output Compare*, plus simple à utiliser.

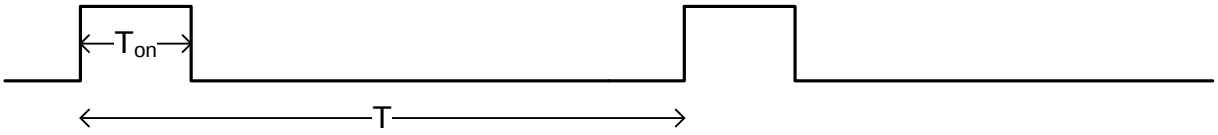
Il possède aussi deux périphériques d'interface UART que nous utiliserons pour la communication entre les deux processeurs.

Interface avec les moteurs

Le robot est équipé de servomoteurs commandés en tension. Un servomoteur est un système intégrant un moteur à courant continu, un réducteur pour obtenir une vitesse plus faible (et un couple plus important) à l'axe de sortie, un hacheur pour alimenter le moteur et un circuit de contrôle qui commande le hacheur.

Les servomoteurs ont 3 fils d'interface : une masse, une alimentation et un signal de commande.

Pour contrôler le moteur, le dsPIC devra envoyer une consigne au circuit de contrôle du moteur, en appliquant le signal de commande suivant :



$$\begin{cases} T_{on} = 1{,}5ms \Leftrightarrow V_{mot} = 0 \\ T_{on} = 2ms \Leftrightarrow V_{mot} = V_{alim} \\ T_{on} = 1ms \Leftrightarrow V_{mot} = -V_{alim} \end{cases}$$

Figure 1 : Signal de commande des servomoteurs

La période *T* du signal n'a pas d'importance, elle doit simplement être comprise entre 3ms et 20ms.

La consigne est codée dans la durée de l'impulsion T_{on} ; elle définit la tension qui sera appliquée au moteur suivant la relation donnée ci-dessus.

Principe de l'encodeur en quadrature

Dans les applications de robotique ou de motorisation, il arrive fréquemment que l'on souhaite mesurer le déplacement d'une roue. A titre d'exemple, cela permet de connaitre le nombre de tours parcourus par les roues d'une voiture et de suivre son mouvement à des fins de régulation et/ou de contrôle.

Le capteur de rotation peut prendre de nombreuses formes, mais une des plus usitées est l'encodeur incrémental en quadrature. Ce dispositif est composé d'une roue perforée solidaire de l'axe en rotation, et d'un capteur (souvent optique) permettant de détecter le passage d'une fente de la roue. Un second capteur, décalé du premier, permet de connaitre le sens de déplacement.

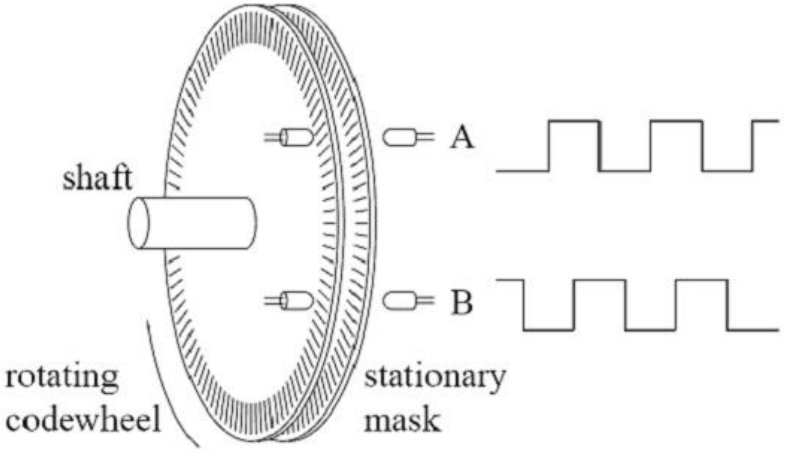


Figure 2 : Principe d'un encodeur en quadrature

Le passage de la roue génère deux trains d'ondes carrées, nommés A et B. Selon que le flanc montant de A arrive avant ou après le flanc montant de B, on sait dans quel sens se déplace la roue. Chaque flanc correspond donc à la rotation d'une fraction connue de la roue dans le sens positif ou négatif.

Interface avec les encodeurs en quadrature

Le dsPIC33FJ128MC804 possède 2 périphériques conçus pour interpréter les signaux d'un encodeur en quadrature : les *Quadrature Encoder Interface* (QEI).

Le rôle du QEI est de surveiller l'évolution de ces deux signaux et d'incrémenter (ou de décrémenter selon le sens de rotation) un compteur. La lecture de ce compteur permet de surveiller l'évolution du déplacement.

Modes de fonctionnement

Le mode de fonctionnement du QEIx (x = 1 ou 2) est défini par le registre de contrôle QEIxCON qui est décrit à la Figure 6.

Les figures 3 et 4 montrent les deux modes de comptage possibles :

- En mode 2X, le compteur POSxCNT est modifié à chaque flanc montant ou descendant du canal A.
- En mode 4X, l'opération est également réalisée à chaque flanc montant ou descendant du canal B.

Le signal QEIxCON.UPDNx permet de connaitre le sens de rotation. Le choix du mode se fait à travers le champ QEIxCON.QEIM, comme le montre le Tableau 1.

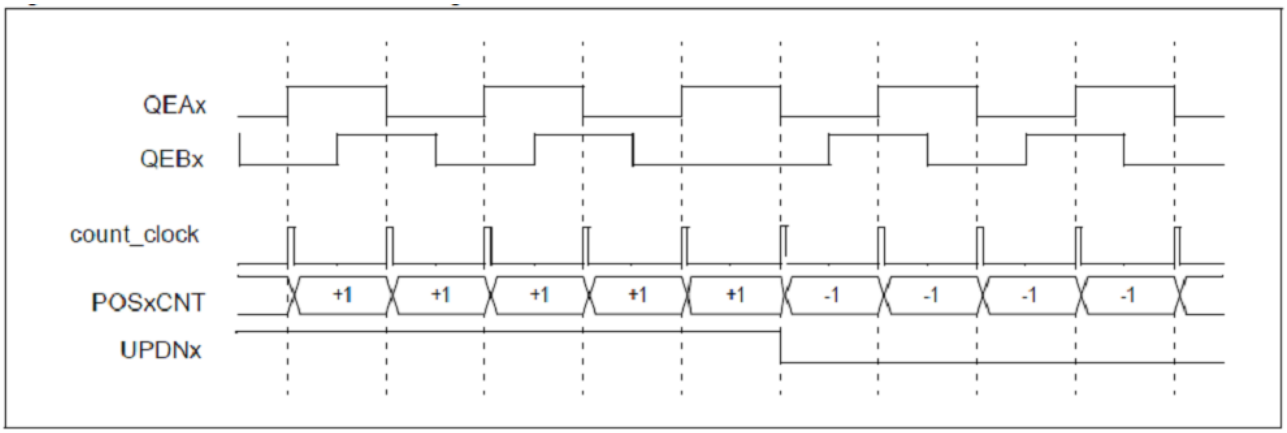


Figure 3 : QEI en mode 2X

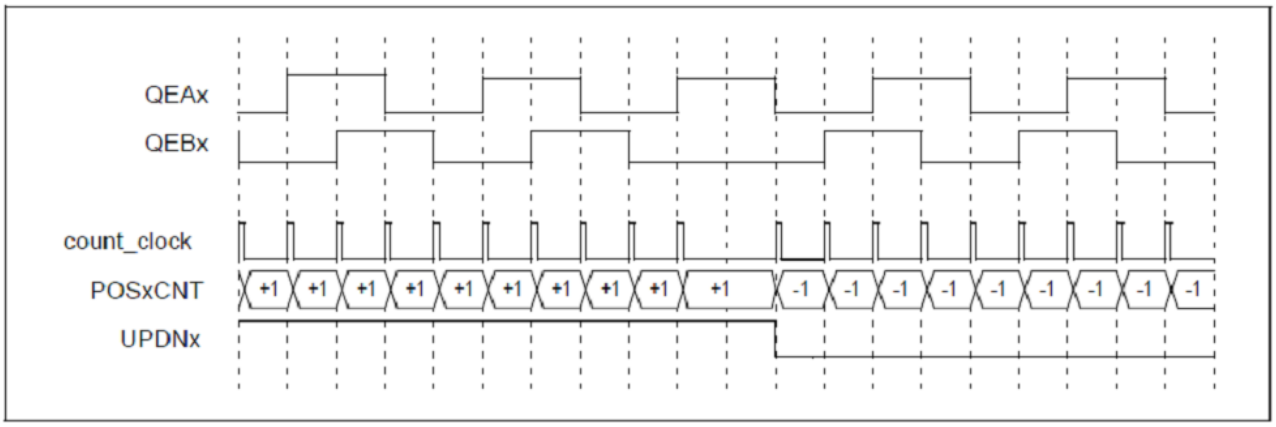


Figure 4 : QEI en mode 4X

QEIxCON.QEIM	Mode de comptage	Remise à zéro par l'index
0b000	OFF	OFF
0b100	2X	OFF
0b101	2X	ON
0b110	4X	OFF
0b111	4X	ON

Tableau 1 : choix du mode

Les bornes A et B peuvent être inversées dans le µC, afin de choisir le sens positif de rotation. Cela se faire via le bit QEIxCONbits.SWPAB (mettre le bit à '1' pour faire l'inversion). Ce fonctionnement de base possède quelques variantes, nous en citons deux ici :

- Dans certains cas, un signal supplémentaire nommé index est généré par l'encodeur au début de chaque de tour de roue. Il est alors possible de configurer le QEI de sorte à ce que le compteur soit automatiquement remis à zéro. L'activation de ce mode est dépendant de la valeur de QEIxCON.QEIM (Figure 3).
- Lorsque l'index est désactivé, il est également possible de provoquer la mise à zéro du compteur à une valeur donnée nommée MAXxCNT comme le montre la Figure 5. Lorsque le compteur s'incrémente alors que sa valeur actuelle est égale à MAXxCNT, il est automatiquement remis à zéro. Inversement, si le compteur se décrémente alors que sa valeur actuelle est zéro, il est automatiquement forcé à la valeur de MAXxCNT. Par défaut, MAXxCNT = 65535, qui est le plus grand nombre entier non signé représentable sur 16bits.

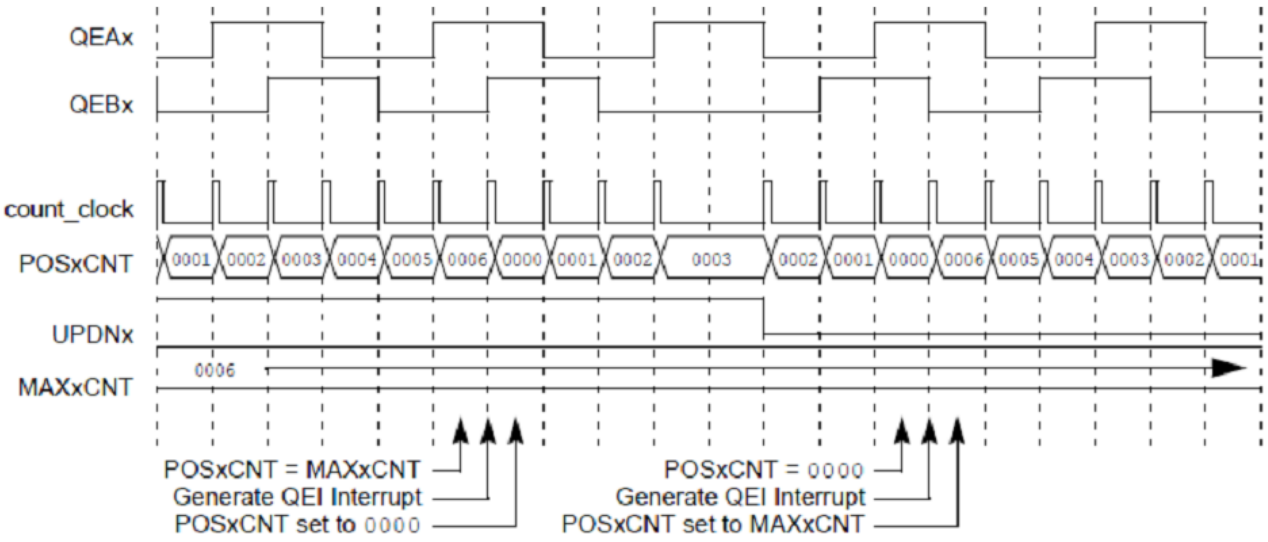


Figure 5 : Débordement du compteur

Comme le montre la Figure 5, le débordement du compteur provoque la levée du flag du QEI : IFS3bits.QEI1IF ou IFS4bits.QEI2IF.

Configuration du QEI

Les étapes nécessaires à la configuration du QEIx sont les suivantes :

- Choix du monde de comptage via QEIxCON.QEIM (figure 32). Le QEI est alors automatiquement activé.
  Lorsque le QEI est actif, les bornes d'IO liées à l'encodeur sont automatiquement configurées en entrées, et les registres TRIS n'ont plus d'effet sur le sens des bornes.

  Remarque : notre dsPIC ayant un *Peripheral Pin Select*, il est néanmoins nécessaire d'utiliser ce dernier pour définir les pattes utilisées par le QEIx.
- Activation éventuelle de l'index via QEIxCON.QEIM
- Activation éventuelle de l'interruption (IEC3bits.QEI1IE ou IEC4bits.QEI2IE) et écriture de la routine d'interruption
- Modification éventuelle de MAXxCNT

Dans le corps du programme, il est alors nécessaire de regarder périodiquement la valeur du compteur POSxCNT afin de surveiller l'évolution de la position.

Application à notre robot

Le fonctionnement des encodeurs est décrit dans le document "*VEX - Quadrature encoder.pdf*". Chaque canal fournit 90 impulsions par tour. En utilisant le mode 4x du périphérique *QEI*, on peut obtenir 360 impulsions par tour. Les encodeurs sont fixés aux axes des roues.

Register 15-1: QEIxCON: QEI Control Register

R/W-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
CNTERR	—	QEISIDL	INDEX	UPDN	QEIM<2:0>		
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SWPAB	PCDOUT	TQGATE(1)	TQCKPS<1:0>(1)		POSRES	TQCS(1)	UDSRC(1)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **CNTERR:** Count Error Status Flag bit
1 = Position count error has occurred
0 = Position count error has not occurred
(CNTERR flag applies only when QEIM<2:0> = 110 or 100)

bit 14 **Unimplemented:** Read as '0'

bit 13 **QEISIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12 **INDEX:** Index Pin State Status bit (read-only)
1 = Index pin is high
0 = Index pin is low

bit 11 **UPDN:** Position Counter Direction Status bit
1 = Position counter direction is positive (+)
0 = Position counter direction is negative (-)
(Read-only bit when QEIM<2:0> = 1xx)
(Read/Write bit when QEIM<2:0> = 001)

bit 10-8 **QEIM<2:0>:** Quadrature Encoder Interface Mode Select bits
111 = Quadrature Encoder Interface enabled (x4 mode) with position counter reset by match (MAXxCNT)
110 = Quadrature Encoder Interface enabled (x4 mode) with index pulse reset of position counter
101 = Quadrature Encoder Interface enabled (x2 mode) with position counter reset by match (MAXxCNT)
100 = Quadrature Encoder Interface enabled (x2 mode) with index pulse reset of position counter
011 = Unused (module disabled)
010 = Unused (module disabled)
001 = Starts 16-bit Timer
000 = Quadrature Encoder Interface/Timer off

bit 7 **SWPAB:** Phase A and Phase B Input Swap Select bit
1 = Phase A and Phase B inputs are swapped
0 = Phase A and Phase B inputs are not swapped

bit 6 **PCDOUT:** Position Counter Direction State Output Enable bit
1 = Position counter direction status output is enabled (QEI logic controls state of I/O pin)
0 = Position counter direction status output is disabled (normal I/O pin operation)

bit 5 **TQGATE:** Timer Gated Time Accumulation Enable bit(1)
1 = Timer gated time accumulation is enabled
0 = Timer gated time accumulation is disabled

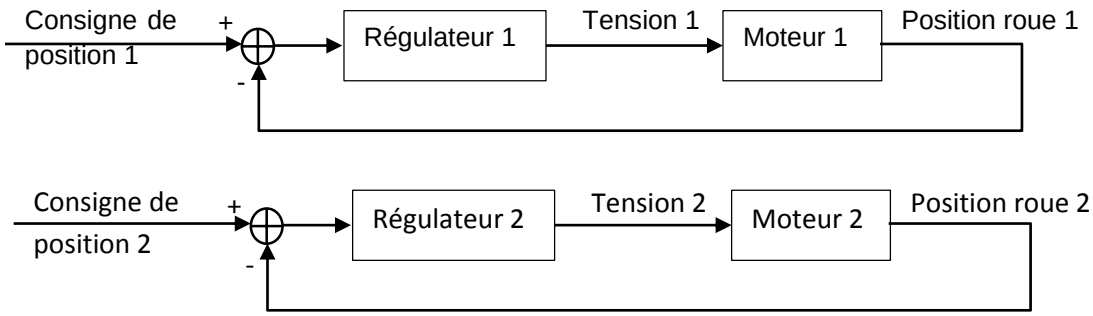
Note 1: When configured for QEI mode, the TQGATE, TQCKPS, TQCS and UDSRC bits are ignored.

Figure 6 : Registre QEIxCON

Développement du régulateur de position

Schéma de régulation

Le schéma ci-dessous décrit la structure du régulateur de position de notre robot :



Chaque moteur est régulé séparément par deux régulateurs identiques. Si le robot doit aller en ligne droite, on fournira la même consigne aux deux moteurs. Si le robot doit tourner sur place, on fournira des consignes égales en valeur absolue mais de signes opposés aux moteurs.

Modélisation du robot

Introduction

Comme nous n'avons pas d'informations sur le fonctionnement interne des moteurs, nous allons utiliser une approche empirique pour créer un modèle de notre robot.

Chaque roue du robot est équipée d'un encodeur en quadrature possédant 90 dents, ce qui nous permet de mesurer la position angulaire des roues avec une précision de 1°. Les roues font 4" de diamètre (101,6mm).

Mesures

Nous savons qu'il existe une relation entre la tension appliquée au moteur (définie par le rapport cyclique) et sa vitesse. Pour obtenir cette relation, nous allons relever la réponse de la vitesse du robot à un échelon de rapport cyclique de 0 à la valeur désirée. Cet échelon est appliqué aux deux moteurs simultanément, pour obtenir un mouvement de translation du robot.

Nous allons travailler avec une fréquence d'échantillonnage de 100Hz, qui est une valeur courante pour le réglage des moteurs en robotique. La vitesse est estimée en comptant le nombre d'impulsions durant une période d'échantillonnage.

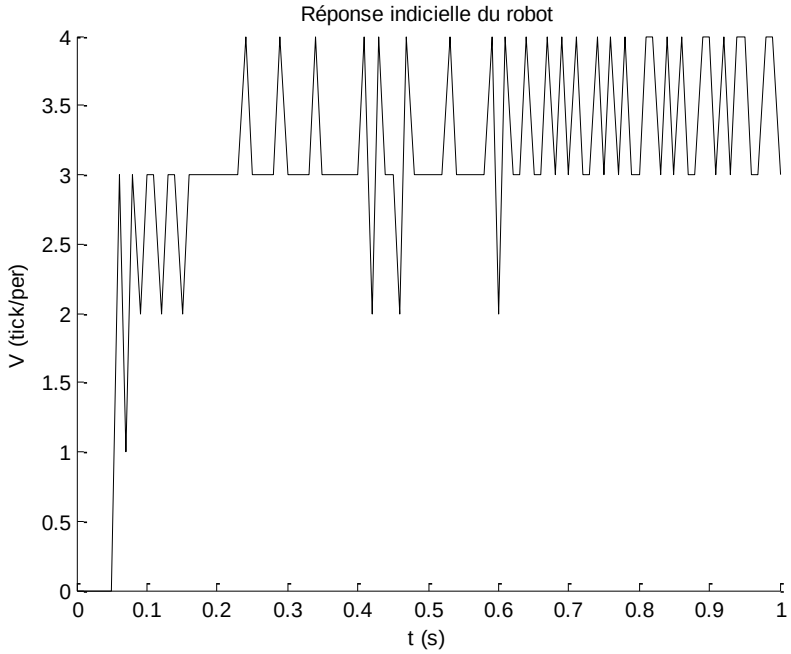


Figure 7 : Réponse indicielle du robot (mesures brutes)

La Figure 7 donne un exemple de la réponse obtenue pour une des roues en utilisant les mesures brutes. On voit qu'elle est fortement bruitée par la quantification.

Pour rendre ces courbes plus lisibles, on fait la moyenne des courbes des deux roues et on la filtre avec un filtre passe-bas d'ordre 2 dont la fréquence de coupure est 20Hz.

La Figure 8 montre les courbes ainsi obtenues pour quelques rapports cycliques. On observe un transitoire qui peut être approché par un système du 1er ordre, suivi d'une phase de régime où la vitesse est relativement constante. L'ondulation périodique qu'on observe est probablement due à une variation dans les frottements en fonction de la position angulaire de la roue ; en effet, la période est compatible avec le diamètre de la roue et la vitesse.

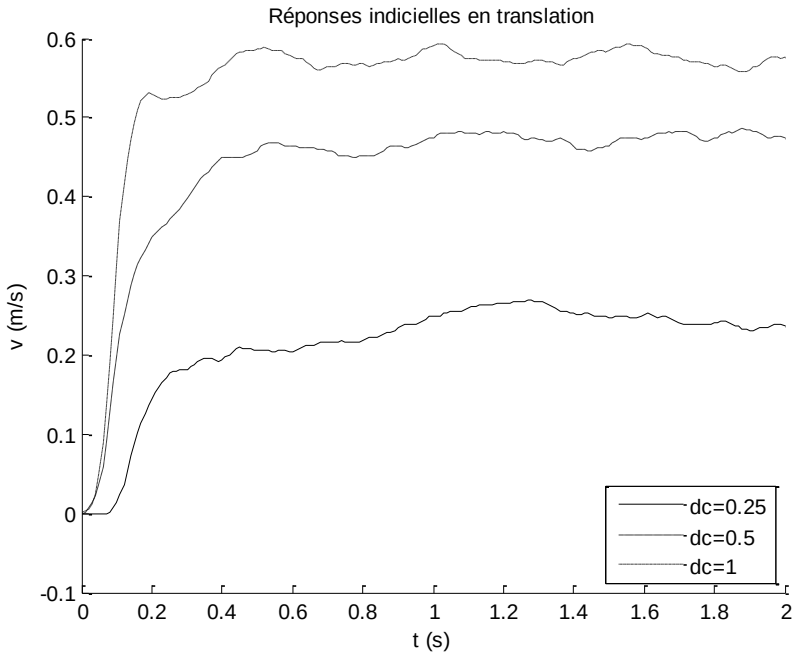


Figure 8 : Réponses indicielles filtrées

Caractéristique statique

Pour obtenir la caractéristique statique de la vitesse du robot en translation, nous avons calculé la moyenne de la vitesse en régime, par pas de 5% de rapport cyclique :

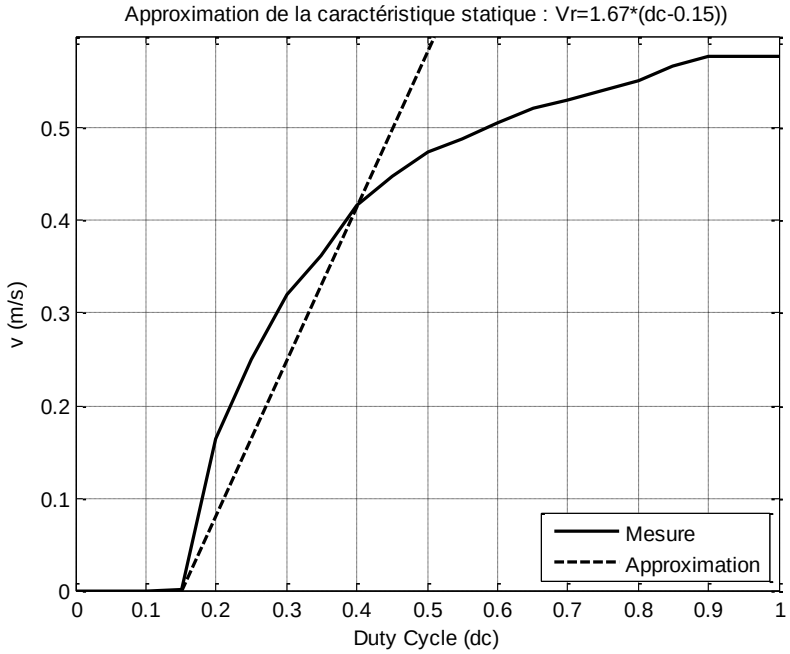


Figure 9 : Caractéristique statique en vitesse

Nous obtenons une courbe (alors que nous nous attendions à avoir une droite). Comme nous voulons un modèle linéaire de notre robot, nous allons approximer cette courbe en choisissant une vitesse nominale pour notre robot. Nous choisissons 0,4m/s pour que notre droite ne s'éloigne pas trop de la courbe.

D'autre part, nous devons tenir compte dans notre modèle de la zone morte pour les rapports cycliques en dessous de 0,15. En effet, dans cette zone, le régulateur n'a aucun effet sur le robot, il faut donc à tout prix l'éviter.

Nous obtenons alors une caractéristique linéaire par morceau :

$$\begin{cases} Vr = \mathrm{k}_v \cdot (\delta + \delta_0) & si\ \delta < -\delta_0 \\ Vr = 0 & si - \delta_0 \leq \delta \leq \delta_0 \\ Vr = \mathrm{k}_v \cdot (\delta - \delta_0) & si\ \delta > \delta_0 \end{cases}, \qquad \text{avec } k_v = 1{,}67\,{}^{m}/_{s} \text{ et } \delta_0 = 0{,}15$$

Cette approximation s'éloigne fortement de la courbe expérimentale pour des vitesses au-delà de la vitesse nominale ; mais nous ne comptons pas utiliser cette partie de la courbe de toute façon.

Modèle dynamique

Pour modéliser le comportement dynamique du robot, nous allons utiliser la réponse indicielle à la vitesse nominale et l'interpoler par une réponse indicielle d'un système du 1er ordre :

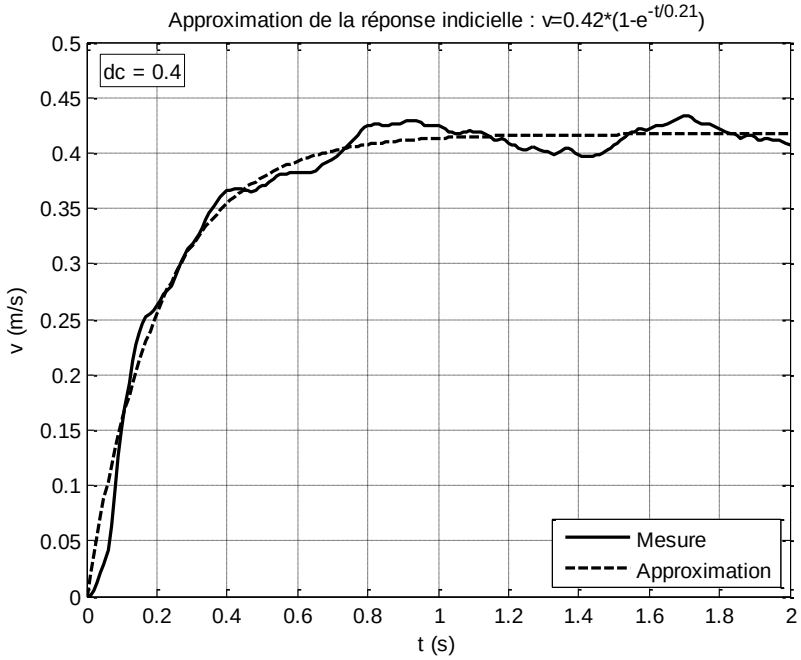
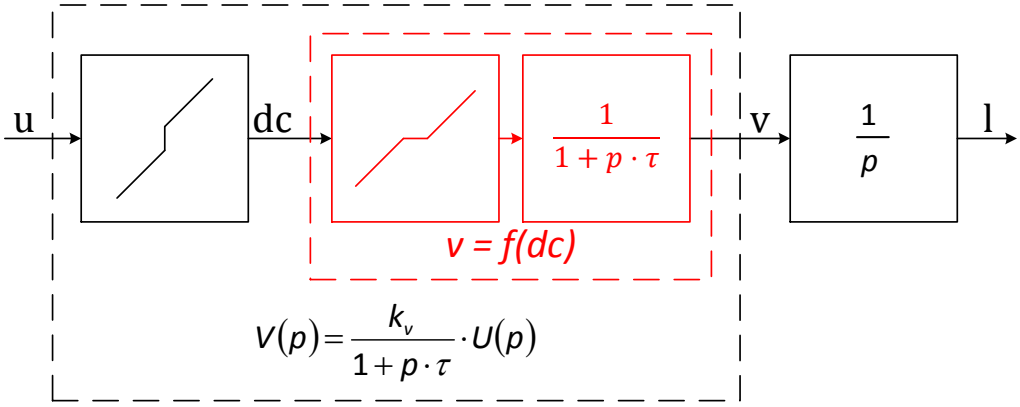


Figure 10 : Approximation de la réponse indicielle à la vitesse nominale

Nous obtenons une constante de temps : $\tau = 210\text{ms}$; le gain correspond évidemment à la vitesse nominale.

Modèle complet

Nous avons maintenant tous les blocs nécessaires pour modéliser le comportement de notre robot en translation :



Les blocs rouges représentent respectivement la caractéristique statique et la réponse indicielle. Ils représentent la fonction de transfert en vitesse de notre robot. Malheureusement, elle n'est pas linéaire à cause de la zone morte du la caractéristique statique.

Pour rendre notre système linéaire, nous allons ajouter une non-linéarité de commande pour transformer notre système $V = f(\delta)$ en un système $V = f'(u)$ linéaire. Ce bloc est décrit par :

$$\delta = \begin{cases} u - \delta_0 & si\ u < 0 \\ 0 & si\ u = 0 \\ u + \delta_0 & si\ u > 0 \end{cases}$$

Comme nous voulons une régulation en position, il nous suffit alors d'ajouter un intégrateur pour obtenir notre fonction de transfert en position :

$$H(p) = \frac{L(p)}{U(p)} = \frac{k_v}{(1 + p \cdot \tau) \cdot p}$$

Schéma de régulation

Notre système possède un pôle intégrateur, nous pouvons donc nous contenter d'un simple régulateur proportionnel :

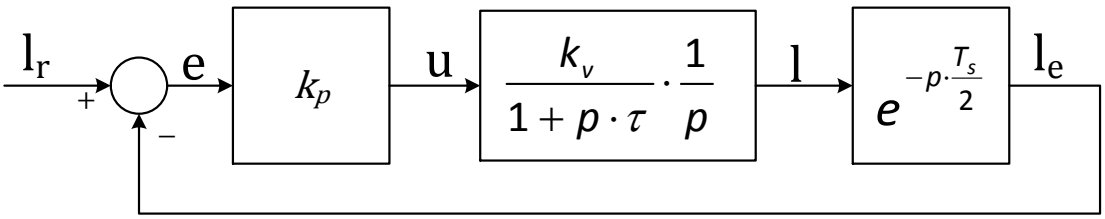


Figure 11 : Schéma de régulation

Le bloc le plus à droite représente l'échantillonnage de la mesure de vitesse. C'est un délai d'une demi-période d'échantillonnage.

Nous obtenons la fonction de transfert en boucle ouverte de notre système :

$$BO(p) = \frac{L_e(p)}{E(p)} = \frac{k_p \cdot k_v \cdot e^{-p \cdot \frac{T_s}{2}}}{(1 + p \cdot \tau) \cdot p}$$

Dimensionnement du régulateur

Nous allons utiliser les critères des marges de gain et de phase pour dimensionner notre régulateur.

Nous avons donc besoin des expressions du gain et de la phase de la réponse harmonique du système en boucle ouverte :

$$\begin{cases} \|BO(\omega)\| = \dfrac{k_p \cdot k_v}{\omega \cdot \sqrt{1 + (\omega \cdot \tau)^2}} \\ \phi\big(BO(\omega)\big) = -\omega \cdot \dfrac{T_s}{2} - arctg(\omega \cdot \tau) - \dfrac{\pi}{2} \end{cases}$$

Nous voulons une marge de gain d'au moins 6dB et une marge de phase d'au moins 30°. Ce sont des valeurs de bonnes pratiques couramment utilisées.

Marge de gain

La marge de gain est définie comme l'inverse du module de la réponse harmonique en boucle ouverte à la pulsation pour laquelle sa phase vaut -180°. Cette marge de gain doit être supérieure à 1 pour que le système soit stable.

Soit ω_1 la pulsation recherchée :

$$\phi\big(BO(j\omega_1)\big) = -\pi \quad \Leftrightarrow \quad arctg(\omega_1 \tau) + \omega_1 \frac{T_s}{2} = \frac{\pi}{2}$$

En résolvant cette relation numériquement, on trouve $\omega_1 = 30{,}7 rad$

Le critère de la marge de gain peut s'écrire :

$$\|BO(j\omega_1)\| = -6\text{dB} = \frac{1}{2} = \frac{k_{p1} \cdot k_v}{\omega_1 \cdot \sqrt{1 + (\omega_1 \tau)^2}}$$

$$\Rightarrow k_{p1} = \frac{\omega_1 \cdot \sqrt{1 + (\omega_1 \tau)^2}}{2 \cdot k_v} = 60{,}2 m^{-1}$$

Marge de phase

La marge de phase est définie comme la différence entre la phase de la réponse harmonique en boucle ouverte à la pulsation pour laquelle son module vaut 1 et −180°. Cette marge de phase doit être positive pour que le système soit stable.

Soit ω_2 la pulsation recherchée, la marge de phase peut s'écrire :

$$M_\phi = \frac{\pi}{6} = \phi\big(BO(j\omega_2)\big) - (-\pi) \;=\; -\omega_2\frac{T_s}{2} - arctg(\omega_2\tau) - \frac{\pi}{2} + \pi$$

$$\Leftrightarrow arctg(\omega_2\tau) \mp \frac{T_s}{2} = \frac{\pi}{3}$$

En résolvant cette relation numériquement, on trouve $\omega_2 = 7{,}54rad$.

ω_2 est définie par l'expression :

$$\|BO(j\omega_2)\| = 1 \;\Leftrightarrow\; \frac{k_{p1}\cdot k_v}{\omega_2\cdot\sqrt{1+(\omega_2\tau)^2}} = 1$$

$$\Rightarrow k_{p2} = \frac{\omega_2\cdot\sqrt{1+(\omega_2\tau)^2}}{k_v} = 8{,}49m^{-1}$$

Conclusion

Nous choisirons évidemment le k_p le plus petit de ceux obtenus par les différents critères de dimensionnement :

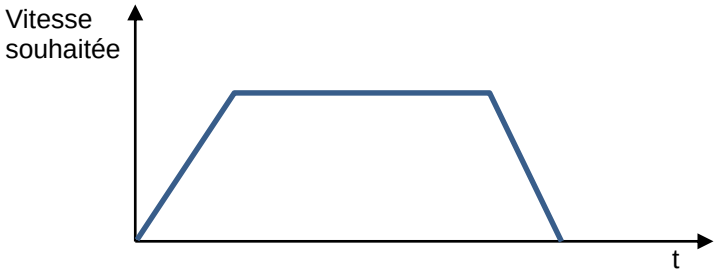
$$k_p = 8{,}49m^{-1}$$

Génération de consigne

Pour que le robot se déplace en ligne droite, il faut que les deux roues parcourent la même distance, en même temps. Pour garantir cela, on doit leur fait suivre une trajectoire physiquement réalisable.

Nous avons choisi une trajectoire pour laquelle l'évolution de la vitesse au cours du temps est un trapèze ; elle est donc divisée en 3 parties :

- Une phase d'accélération, où la vitesse, nulle au départ, croît linéairement jusqu'à atteindre la vitesse nominale
- Une phase à vitesse constante
- Une phase de décélération, semblable à la phase de décélération.



En intégrant ce profil de vitesse, nous obtenons une trajectoire de position, qui sera la consigne de nos régulateurs : pour que le robot se déplace en ligne droite, on applique la même consigne aux moteurs ; pour qu'il tourne, on inverse le signe de la consigne appliquée à l'un des moteurs.

On peut estimer l'accélération maximale du robot en se basant sur la pente à l'origine de la réponse indicielle de la Figure 10 ; on trouve $2\,{}^{m}/_{s^2}$.

Après avoir essayé plusieurs accélérations, nous décidons de limiter l'accélération de notre consigne à $0{,}5{}^{m}/_{s^2}$ pour que notre robot puisse la suivre sans difficulté.