UNIVERSITÉ LIBRE DE BRUXELLES

ECOLE POLYTECHNIQUE DE BRUXELLES

ELEC-H-309 - Rapport

Projet Intégré

BA3 - Electronique et télécommunications

Théo Lepoutte
Thibault Dessart

24 mai 2019

Table des matières

1 Introduction

Dans le carde du projet intégré de l'unité d'enseignement ELEC-H309, il est demandé aux étudiants de programmer le déplacement d'un robot commandé par le biais d'un son modulé en fréquence. L'objectif principal de ce projet est de mettre en pratique les nombreuses connaissances acquises jusqu'à aujourd'hui.

Ce rapport est constitué de deux grandes parties. La première comprend le travail de programmation nécessaire à la régulation du déplacement du robot. La suivante, quant à elle, se concentre essentiellement sur le dimensionnement de la chaîne d'acquisition du signal sonore.

2 Régulation du mouvement du robot

Afin de pouvoir contrôler au mieux les déplacements du robot, une régulation doit être appliquée sur les roues. En effet, on ne peut pas se contenter de donner une consigne en tension pour que les moteurs tournent à une certaine vitesse donnée. Ceci vient du fait que ces derniers ne se comportent pas de manière similaire, des variables telles que l'usure et les frottements affectent les caractéristiques de rotation. Il faut donc appliquer une boucle de régulation sur notre système. La régulation choisie est la proportionelle car elle demande moins de temps de calcul que les régulations proportionnelle-dérivée (PD) et proportionnelle-dérivée-intégrale (PID). Ce régulateur consiste à calculer l'erreur, à la multiplier par une constante k_p et à ajouter ce résultat à la consigne.

2.1 Initialisation des moteurs

Il faut tout d'abord définir un *timer* qui permettra de commander les servomoteurs s'occupant de la rotation. Ceux-ci fonctionnent en appliquant une tension continue selon un certain rapport cyclique [1]. Toutes les T secondes, un signal haut est envoyé pendant T_{on} secondes.

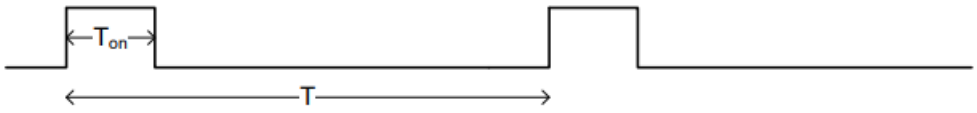


FIGURE *1 – Signal de commande des servomoteurs*

La période maximale sur laquelle le signal peut s'étendre n'ayant que pour contrainte de se trouver entre 3ms et 20ms, on choisit la borne inférieure de cette condition afin d'avoir une commande plus rapide. Le *timer* est donc défini de manière à respecter au mieux cette consigne. Il faut donc régler sa période avec le registre PR2 pour qu'il respecte l'équation suivante :

$$\frac{\text{PR2}}{F_{cy}} = T \iff \frac{\text{PR2}}{40 \cdot 10^6 \text{Hz}} = 3 \cdot 10^{-3}\text{s} \iff \text{PR2} = 120\,000 \tag{1}$$

1. Le rapport cyclique correspond au rapport entre la durée d'un phénomène et une période T.

où F_{cy} correspond à la fréquence d'horloge du dsPic qui est de 40MHz. On ne peut cependant pas avoir une valeur de 120 000 pour le registre PR2 car celui-ci est codé sur 16 bits et est donc limité à une valeur ne dépassant pas $2^{16} - 1 = 65535$. On lui donne alors la valeur de 15 000 à laquelle on impose un *prescaler*[2] de 8 (car $8 \cdot 15\,000 = 120\,000$).

Pour calculer les valeurs de la durée des états hauts T_{on} à imposer à nos moteurs, il faut respecter les conditions suivantes :

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

En suivant la logique de l'équation (1), on peut définir les valeurs à donner à $OC1R$ et $OC2R$[3] :

$$\begin{cases} T_{on} = 1,5ms \iff V_{mot} = 0 \iff OCxR = 7500 \\ T_{on} = 2ms \iff V_{mot} = V_{alim} \iff OCxR = 10000 \\ T_{on} = 1ms \iff V_{mot} = -V_{alim} \iff OcxR = 5000 \end{cases} \tag{3}$$

2.2 Calcul des k_p de translation et de rotation

Pour effectuer les calculs des deux coefficients k_p des régulateurs, il est nécessaire d'étudier les caractéristiques statique et dynamique des moteurs[4].

2.2.1 Caractéristique statique

La caractéristique statique est obtenue en traçant la courbe de la vitesse moyenne en régime en fonction du rapport cyclique.

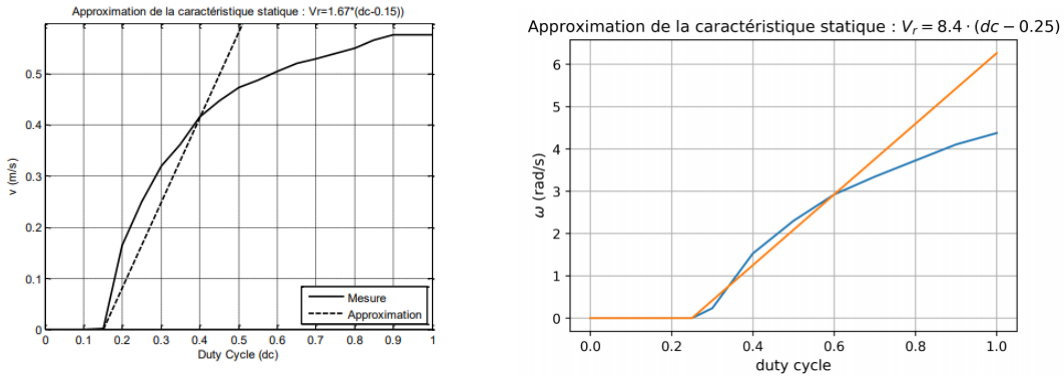


FIGURE 2 – *Caractéristique statique en translation et en rotation*

2. Le *prescaler* correspond à un coefficient diviseur de la fréquence d'horloge du timer permettant d'augmenter la période maximale.
3. $OCxR$ est un registre de comparaison dont la valeur définit le rapport cyclique.
4. Cette section est fortement inspirée du document *"Étude du déplacement du robot"* fourni dans le cadre de ce projet.

Étant donné que notre modèle est linéaire, il convient d'approximer la courbe par une droite et de choisir une vitesse nominale la plus proche possible des valeurs réelles, soit l'intersection entre la droite et la courbe. Cette vitesse est choisie pour la translation à 0.4 m/s et pour la rotation à 3 rad/s.

On constate également que le modèle possède une zone morte (que l'on note δ_0) dans laquelle le robot ne réagit pas, bien qu'une tension lui soit appliquée. Cette dernière correspond à 0.15 dans le cas de la translation et à 0.25 dans celui de la rotation.

On peut déduire de tout ça une caractéristique linéaire par morceaux :

$$\begin{cases} V_r = k_v(\delta + \delta_0) & si \quad \delta < -\delta_0 \\ V_r = 0 & si \quad -\delta_0 \leq \delta \leq \delta_0 \\ v_r = k_v(\delta - \delta_0) & si \quad \delta > \delta_0 \end{cases} \tag{4}$$

dans laquelle k_v vaut 1.67 m/s dans le cas de la translation et 8.4 rad/s pour la rotation.

2.2.2 Caractéristique dynamique

La caractéristique dynamique s'obtient en interpolant la réponse indicielle obtenue à la vitesse nominale par une réponse indicielle d'un système du premier ordre.

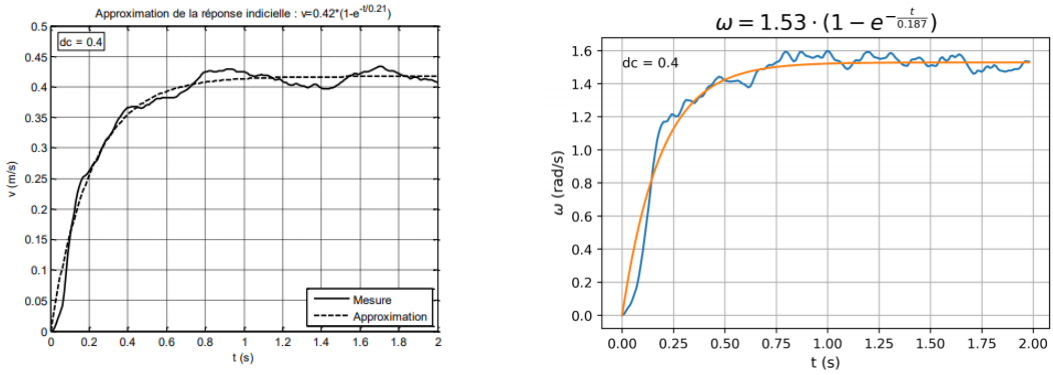


FIGURE *3 – Caractéristique dynamique en translation et en rotation*

Elle nous permet de définir un constante de temps τ qui vaut 210 ms pour la translation et 184 ms pour la rotation.

2.2.3 Modèle complet

Toutes les données nécessaires à l'élaboration du modèle du déplacement du robot sont désormais connues. Celui-ci est représenté dans la figure 4.

Les caractéristiques statique et indicielle sont représentées par les blocs rouges. Ceux-ci correspondent à la fonction de transfert en vitesse du robot. Cette dernière n'étant pas linéaire à cause de la présence d'une zone morte, il faut la linéariser en ajoutant une non-linéarité de commande, ce qui transforme le système $V = f(\delta)$ en $V = f'(u)$. Le bloc ainsi créé est décrit par :

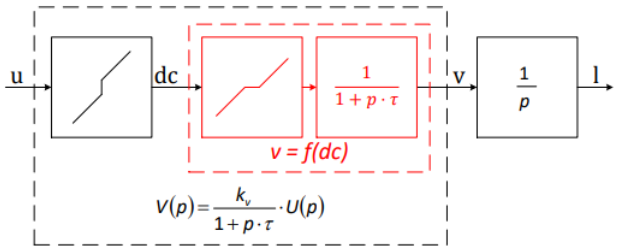


FIGURE 4 – *Modèle du déplacement du robot*

$$\delta = \begin{cases} u - \delta_0 & si \quad u < 0 \\ 0 & si \quad u = 0 \\ u + \delta_0 & si \quad u > 0 \end{cases} \tag{5}$$

Afin d'obtenir une régulation de position, il suffit d'ajouter un intégrateur. La fonction de transfert suivante est alors obtenue :

$$H(p) = \frac{L(p)}{U(p)} = \frac{k_v}{(1 + p\tau)p} \tag{6}$$

2.2.4 Schéma de régulation

Le régulateur proportionnel dont notre système peut se satisfaire est schématisé ci-dessous :

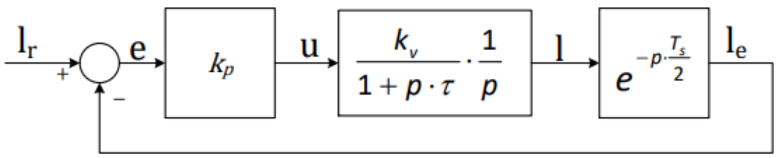


FIGURE 5 – *Schéma de régulation*

Le bloc $e^{p\frac{T_s}{2}}$ est une représentation de l'échantillonnage effectué pour la mesure de vitesse et correspond à un délai d'une demi-période d'échantillonage. La période d'échantillonnage choisie est $T_s = 0.01$s. La fonction de transfert en boucle ouverte est donc donnée par :

$$BO(p) = \frac{L_e(p)}{E(p)} = \frac{k_p k_v e^{p\frac{T_s}{2}}}{(1 + p\tau)p} \tag{7}$$

2.2.5 Dimensionnement des régulateurs

Pour le dimensionnement des régulateurs, les critères de marge de phase et de marge de gain sont utilisés. Une fois calculés, les k_p les plus adéquats seront choisis. Les expressions de ces deux marges sont données par :

$$\begin{cases} ||BO(\omega)|| = \frac{k_p k_v}{\omega\sqrt{1+(\omega\tau)^2}} \\ \phi(BO(\omega)) = -\omega\frac{T_s}{2} - arctan(\omega\tau) - \frac{\pi}{2} \end{cases} \tag{8}$$

Les conditions sont fixées à une marge de gain d'au moins 6 dB et une de phase d'au moins 30° qui sont des valeurs de bonne pratique.

Marge de gain

Elle correspond à l'inverse du module de la réponse harmonique en boucle ouverte pour une pulsation de $-180°$ et doit être supérieure à 1 afin d'avoir un système stable. Si ω_1 correspond à la pulsation recherchée elle doit obéir à cette équation :

$$\phi(BO(\omega)) = -\pi \iff arctan(\omega_1\tau) + \omega_1\frac{T_s}{2} = \frac{\pi}{2} \tag{9}$$

qui une fois résolue donne $\omega_{1,trans} = 30.7\,\text{rad/s}$ dans le cas de la translation et $\omega_{1,rot} = 32.8\,\text{rad/s}$ pour la rotation.

Il est désormais possible de trouver une valeur pour le premier k_p en considérant le critère de marge de gain :

$$||BO(j\omega_1)|| = -6\text{dB} = \frac{1}{2} = \frac{k_{p1}k_v}{\omega_1\sqrt{1+(\omega_1\tau)^2}} \Rightarrow k_{p1} = \frac{\omega_1\sqrt{1+(\omega_1\tau)^2}}{2k_v} \tag{10}$$

Il en résulte un $k_{p1,trans} = 60.2\,\text{m}^{-1}$ pour la translation et un $k_{p1,rot} = 11.95\,\text{rad}^{-1}$ pour la rotation.

Marge de phase

La marge de phase correspond à la différence entre la phase de la réponse harmonique dont le module vaut 1 et $-180°$. Celle-ci doit être positive afin d'obtenir un système stable.

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

$$\begin{aligned} M_\phi = \frac{\pi}{6} = \phi(BO(j\omega_2)) - (-\pi) = -\omega_2\frac{T_s}{2} - arctan(\omega_2\tau) - \frac{\pi}{2} + \pi \\ \iff arctan(\omega_2\tau) \mp \omega_2\frac{T_s}{2} = \frac{\pi}{3} \end{aligned} \tag{11}$$

Dont le résultat donne deux valeurs de ω_2 possibles pour chacune des régulations : $\omega_{2,trans} = \{7.57\,\text{rad/s}\,;\,9.2\,\text{rad/s}\}$ en translation et $\omega_{2,rot} = \{10.9\,\text{rad/s}\,;\,8.5\,\text{rad/s}\}$ en rotation. Les plus petites valeurs de ces couples sont choisies afin de calculer k_{p2} :

$$||BO(j\omega_2)|| = 1 \iff \frac{k_{p2}k_v}{\omega_2\sqrt{1+(\omega_2\tau)^2}} = 1 \Rightarrow k_{p2} = \frac{\omega_2\sqrt{1+(\omega_2\tau)^2}}{k_v} \tag{12}$$

Ce qui donne $k_{p2,trans} = 8.49\,\mathrm{m}^{-1}$ en translation et $k_{p2,rot} = 1.897\,\mathrm{rad}^{-1}$ en rotation. Ces k_p étant tous deux plus petits que ceux calculés pour le critère de la marge de gain, ce sont ceux-ci qui vont être gardés pour la régulation des mouvements.

2.3 Consigne de vitesse

Afin que le robot puisse se déplacer de manière fluide, ce dernier doit suivre un profil de vitesse (Figure 6). Il consiste en une accélération permettant d'atteindre une certaine vitesse qui, elle, restera constante. Une fois une certaine distance parcourue, une consigne de décélération semblable à la consigne d'accélération est respectée afin de permettre au robot de s'arrêter à la position voulue.

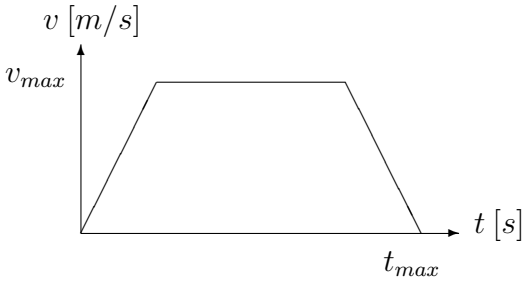


FIGURE *6 – Profil de vitesse*

Ce modèle ne peut malheureusement pas permettre le déplacement dans tous les cas de figure. En effet, dans le cas où le temps qui sera calculé par le code est inférieur à la somme du temps de l'accélération et de celui de décélération (qui en pratique est le même pour les deux), le palier de vitesse constante ne sera pas atteint. Il faut donc respecter un profil de vitesse triangulaire (Figure 7) et non plus trapézoïdal.

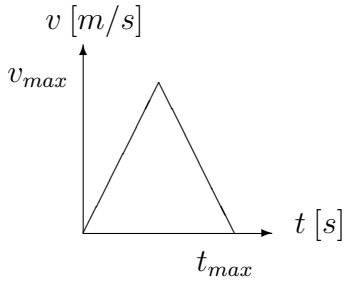


FIGURE *7 – Profil de vitesse triangulaire*

2.3.1 Cas trapézoïdal

En premier lieu, il faut calculer le temps qui doit théoriquement être pris pour effectuer le déplacement. Ce dernier se calcule comme suit et via la fonction *theoricMaxTime* (Annexe A) :

$$t_{max} = 2t_a + \frac{x_{max} - a.t_a^2}{v_{max}} \tag{13}$$

où x_{max} correspond à la consigne de déplacement, a à la consigne d'accélération (et de décélération), v_{max} à la vitesse correspondant au palier et $t_a = \frac{v_{max}}{a}$ au temps d'accélération (qui est égal à celui de décélération).

Une fois cette donnée connue, il est possible de générer en tout instant t la valeur de la position instantanée que devra respecter le robot pour chacune des phases du déplacement :

— **phase d'accélération** : cette phase a lieu dans le cas où $t \leq t_a$. La position x est alors donnée par un MRUA aux conditions initiales nulles :

$$x = \frac{a}{2}t^2 \tag{14}$$

— **phase à vitesse constante** : on se trouve ici dans le cas d'un MRU pour lequel la position initiale correspond au déplacement effectué lors de la première phase. La valeur de x est donc calculée comme suit :

$$x = \frac{a}{2}t_a^2 + v(t - t_a) \tag{15}$$

— **phase de décélération** : quand $t \geq t_{max} - t_a$ on se trouve à nouveau dans le cas d'un MRUA mais avec une accélération négative. Le calcul se fait sur base de la consigne x_{max} et de t_{max} :

$$x = x_{max} - \frac{a}{2}t_a^2 + v(t - (t_{max} - t_a)) - \frac{a}{2}(t - (t_{max} - t_a))^2 \tag{16}$$

On soustrait donc de x_{max} le parcours déjà effectué.

2.3.2 Cas triangulaire

On entre dans le cas triangulaire si $t_a v > x_{max}$ car si cette condition est vérifiée, il est impossible pour le robot d'atteindre la vitesse de palier v. Dans ce cas de figure, le temps maximal t_{max} se calcule différemment du cas trapézoïdal et correspond simplement à deux fois le temps de l'accélération effectuée sur $\frac{x_{max}}{2}$. Ce temps est donc :

$$\frac{x_{max}}{2} = a\frac{t_{max}^2}{2} \iff t_{max} = 2\sqrt{\frac{2x_{max}/2}{a}} \iff t_{max} = 2\sqrt{\frac{x_{max}}{a}} \tag{17}$$

Le temps maximal étant connu, tout est présent pour pouvoir analyser les deux phases du mouvement :

— **phase d'accélération** : si $t \leq \frac{t_{max}}{2}$, le robot accélère suivant la même équation que dans le cas trapézoïdal à savoir l'équation (14).

— **phase de décélération** : dans ce cas-ci, le robot suit aussi la même équation que dans le cas trapézoïdal à la différence près qu'il est nécessaire de calculer la vitesse finale v obtenue à la fin de la phase précédente. Il suit donc l'équation (16) dans laquelle le terme v est remplacé par :

$$v = a\frac{t_{max}}{2} \tag{18}$$

qui correspond à la vitesse atteinte à la moité de la distance à parcourir (et donc en $\frac{t_{max}}{2}$).

2.4 Translation

La consigne de translation permet le déplacement du robot suivant une ligne droite et ce sur une distance donnée. Ce déplacement se fait à une vitesse de croisière de 0.4 m/s et une accélération de 0.5 m/s^2. Pour ce faire, il a été choisi dans un souci d'efficacité de réguler non pas sur la moyenne du déplacement des deux roues mais sur le déplacement de chacune de celles-ci. En effet, l'efficacité est augmentée dans le sens où, si chacune des roues respecte la consigne de vitesse, la rotation du robot sera déjà limitée et ce, sans la présence d'une régulation de rotation (qui sera malgré tout ajoutée). Le schéma de régulation de chacune des roues est représenté par la figure 8.

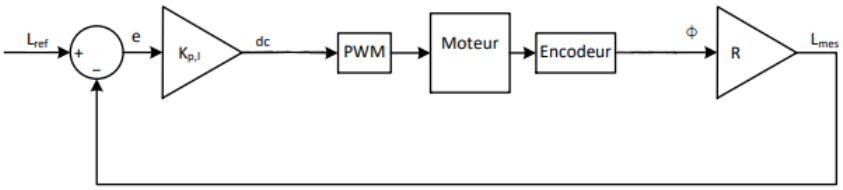


FIGURE 8 – *Boucle de régulation en translation pour un moteur*

La boucle de régulation mesure toutes les 0.01 s (ce qui correspond à une fréquence de 100Hz) l'angle ϕ de chacune des roues via la commande POSxCNT (où x correspond à l'indice du moteur correspondant). Pour avoir la distance parcourue par chacune des roues, il faut multiplier ϕ par le rayon des roues R. Cette donnée étant connue, l'erreur peut être calculée à l'aide de la fonction *posCalc* (Annexe B) qui renvoie la distance que devrait avoir parcouru le robot en fonction du temps écoulé. Il suffit alors de faire la différence entre la distance réelle et la distance théorique afin de connaître l'erreur sur le déplacement.

On peut désormais calculer les rapports cycliques R_{cycl} qui devront être ajoutés ou soustraits (ceci est défini en fonction du sens de montage des moteurs) à la valeur 7500 qui correspond à une vitesse de rotation des moteurs nulle (voir section 2.1) :

$$R_{cycl} = (k_p(x_{th} - x) + U_0)) \cdot 2500 \tag{19}$$

où x_{th} est la distance théorique calculée avec la fonction *posCalc*, x est la distance réelle et U_0 correspond à la zone morte dont il est question dans la section 2.2.1. En ce qui concerne le terme multiplicateur de 2500, il correspond à la différence entre la valeur de l'$OCxR$ de la vitesse maximale et celui d'une vitesse nulle. Il reste juste à ajouter un rapport cyclique correspondant à un angle de rotation de 0° (voir section 2.5) afin d'empêcher la rotation du robot.

2.5 Rotation

Pour le cas de la rotation, la régulation se fait sur l'angle que le robot parcourt par rapport à sa position initiale. Le schéma de la régulation est représenté par le figure 9.

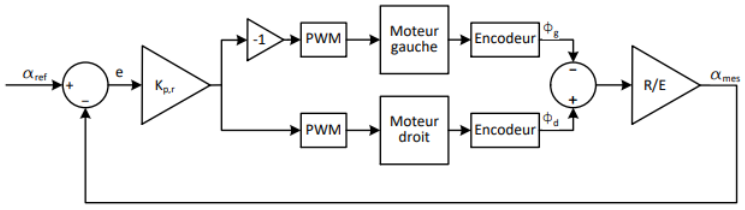


FIGURE 9 – *Boucle de régulation en rotation*

L'angle ϕ parcouru par le robot est donné en radians par :

$$\phi = \frac{3,14}{180} \cdot \frac{R}{E}(y - x) \tag{20}$$

où R est le rayon de chaque roue, E est l'empattement, x et y sont les angles parcourus respectivement par la roue droite et la roue gauche.

Afin de connaître l'angle théorique en fonction du temps, la fonction *posCalc* est aussi utilisée avec comme données d'entrées : la vitesse de croisière de 3 rad/s et l'accélération de 1.5 rad/s^2. Cette donnée théorique permet de connaître l'erreur et, par conséquent, le rapport cyclique à soustraire à la valeur de 7500.

$$R_{cycl} = (k_p(\phi_{th} - \phi) + U_0)) \cdot 2500 \tag{21}$$

où ϕ_{th} est l'angle théorique, ϕ l'angle réel et U_0 correspond à la correction de la zone morte. Un rapport cyclique correspondant à un déplacement de 0m est ajouté à celui-ci afin de s'assurer que le robot ne se déplace pas en translation.

2.6 Tests de la régulation

Une fois que toute la partie théorique développée ci-dessus est incorporée dans le script du code de commande du robot, des tests sur la précision des déplacements ont été effectués. De ceux-ci, il en ressort que ce dernier obéit bien aux instructions qui lui sont imposées. Il reste néanmoins des imperfections lors du mouvement de translation qui pourraient s'expliquer par la présence d'un jeu dans le mouvement des moteurs. Ce jeu inégal entre les deux moteurs entraîne une petite rotation initiale du robot lors de son déplacement en translation (qui après cette petite rotation reste rectiligne) mais n'impacte pas (de manière notable) la consigne de rotation.

3 Dimensionnement de la chaîne d'acquisition

3.1 Microphone

Pour pouvoir capter le signal audio comprenant les commandes à transmettre au robot, un microphone électrostatique à électret est utilisé. L'électret est un matériau diélectrique possédant une charge permanente (l'électret est à l'électrostatique ce que l'aimant est au magnétisme, électret étant un mot-valise formé par les deux mots anglais

electrostatic et *magnet*). Ainsi, en utilisant des électrets pour la membrane et l'armature, on crée un condensateur dont la différence de potentiel varie en fonction de la vibration de la membrane.

Ce condensateur est ensuite relié au *gate* d'un transistor afin d'amplifier le signal reçu et d'adapter l'impédance. Ce transistor doit cependant être alimenté en tension continue (3.3V dans notre cas), c'est pourquoi il est nécessaire d'utiliser une capacité afin de bloquer sa tension d'alimentation. Une résistance de 2.2kΩ est également utilisée pour pouvoir le polariser.

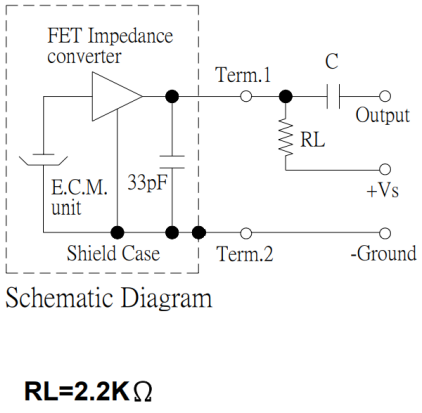


FIGURE *10 – Câblage du microphone à électret*

Ce schéma n'étant pas intuitif, il est préférable de modéliser le circuit via son équivalent de Thévenin, ce qui permettra, par la suite, de faciliter les calculs de R_1 et R_2.

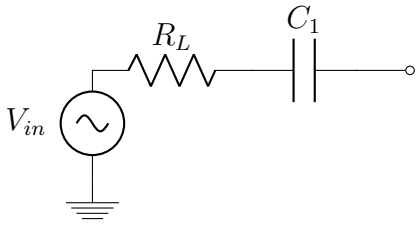


FIGURE *11 – Equivalent de Thévenin du circuit du microphone*

3.2 Amplification

D'après le cahier des charges, le microphone fournit un signal alternatif de 1mV d'amplitude au maximum et l'ADC accepte des tensions comprises entre 0 et 3.3V. De ce fait, l'amplitude maximale du signal sortant doit être de 1.65V et une tension continue d'offset de 1.65V est nécessaire pour pouvoir capter l'entièreté du signal. Afin d'avoir une marge de sécurité, un signal de 1.5V d'amplitude est choisi. Le gain total du circuit amplificateur devra ainsi être de $\frac{1.5\text{V}}{1\text{mV}} = 1500$.

Les fréquences utiles du signal transmis étant de 900 et 1100Hz, un produit gain bande passante d'au moins $1500 \cdot 1100\text{Hz} = 1.65\text{MHz}$ est nécessaire. Par facilité, il faut également que celui-ci puisse être alimenté assymétriquement avec des tensions de 3.3V et 0V. L'amplificateur MCP6281, ayant un produit gain bande passante de 5MHz et répondant aux critères de tension, a été choisi. La valeur du gain maximum atteignable avec cet amplificateur opérationnel est de 4545.

Enfin, le montage inverseur a été préféré au non-inverseur. Celui-ci permet de faciliter la polarisation du signal en utilisant un simple diviseur résistif sur la borne positive de l'amplificateur opérationnel mais également d'économiser une capacité. L'inversion de phase du signal n'est pas problématique dans le cadre de notre projet.

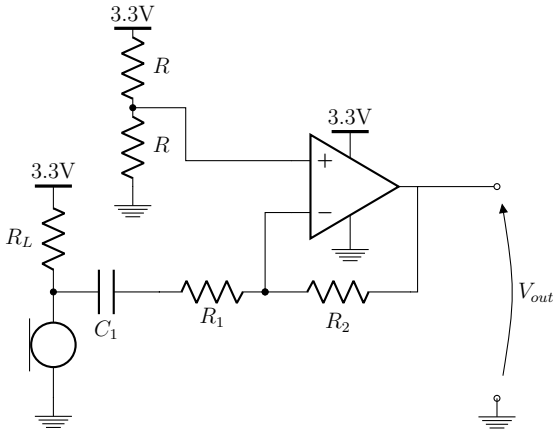


FIGURE *12 – Circuit amplificateur inverseur*

Des résistances de 10kΩ ont été choisies pour le diviseur résistif (R). En effet, des résistances trop faibles vont engendrer un courant trop important et provoquer une perte de puissance inutile tandis que des résistances trop grandes vont générer du bruit.

En remplaçant le montage du micro par son équivalent de Thévenin, on trouve un gain pour l'amplificateur inverseur de $-\frac{R_2}{R_{in}}$ où $R_{in} = R_L + R_1$. De ce fait, on peut, en réalité, choisir une valeur de R_1 nulle afin de diminuer le plus possible la valeur nécessaire pour $R_2 \cdot R_L$ valant 2.2kΩ, la résistance R_2 doit valoir $1500 \cdot 2.2\text{k}\Omega = 3.3\text{M}\Omega$.

3.3 Filtre passe-haut

La capacité C_1 ayant pour but de bloquer la tension d'alimentation du transistor, elle crée un filtre passe-haut. Il ne faut donc pas prendre n'importe quelle valeur pour celle-ci afin de ne pas trop atténuer les signaux utiles. En utilisant le zéro virtuel, on obtient le schéma du circuit qui nous permet ensuite de calculer la réponse en fréquence du signal :

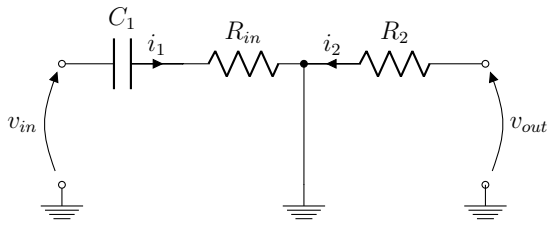


FIGURE *13 – Circuit après simplification avec le zéro absolu*

$$\begin{cases} \underline{V_{in}} = (\frac{1}{j\omega C_1} + R_{in})\underline{I_1} \\ \underline{V_{out}} = R_{in}\underline{I_2} \\ \underline{I_2} = \underline{I_1} \end{cases} \implies H_{HP}(j\omega) = \frac{\underline{V_{out}}}{\underline{V_{in}}} = -\frac{R_2}{R_{in} + \frac{1}{j\omega C_1}} = -\frac{R_2 j\omega C_1}{1 + R_{in} j\omega C_1}$$

Or, on veut que pour $\omega \geq 2\pi 900$ rad/s le gain soit de 1500. On a donc :

$$|H_{HP}(j\omega)| = \frac{R_2 C_1 \omega}{\sqrt{1 + R_{in}^2 \omega^2 C_1^2}} \to 1500 \quad \text{pour} \quad \omega \to 2\pi 900 \text{rad/s}$$

Étant donné que le gain n'atteindra jamais 1500, on prend une valeur qui tend vers 1500. En isolant C_1 et en prenant une fréquence limite de $\omega = 2\pi \cdot 900$ rad/s et un gain de 1485 (99% de 1500), on trouve :

$$C_1 \geq \sqrt{\frac{1485^2}{\omega^2(R_2^2 - 1485^2 R_{in}^2)}} = 564.1\text{nF}$$

On ne peut cependant pas prendre une capacité trop élevée au risque d'avoir des parasites. C'est pourquoi afin d'avoir le meilleur équilibre notre choix s'est porté sur une capacité de 470nF. La légère perte de gain pour les fréquences utiles plus basses n'est pas problématique.

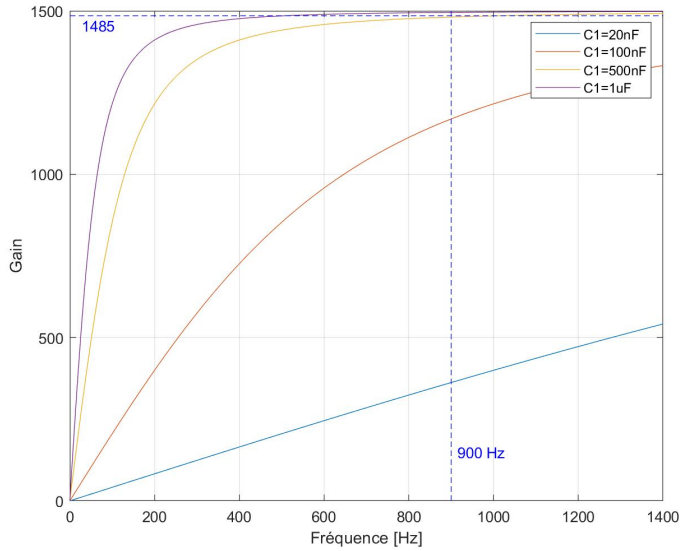


FIGURE *14 – Gain en fonction de la fréquence et de la valeur de C_1*

3.4 Filtre de garde

Afin d'empêcher le repliement spectral lors de l'échantillonnage du signal, il est nécessaire de placer un filtre de garde après notre étage amplificateur afin de pouvoir filtrer les hautes fréquences. Ce filtre devra donc agir comme un passe-bas. D'après le cahier des charges, un filtre de Butterworth d'ordre 2 doit être utilisé. Sachant que l'amplitude de la réponse en fréquence d'un filtre de Butteworth d'ordre 2 est :

$$|H_{LP}(j\omega)| = \sqrt{\frac{1}{1+(\frac{\omega}{\omega_c})^4}}$$

Et sachant que l'atténuation maximale des fréquences utiles doit être de $|H_{LP}| = 0.99$, on peut isoler la fréquence de coupure dans l'équation :

$$0.99 \leq \sqrt{\frac{1}{1+(\frac{2\pi*1100}{2*\pi*f_c})^4}} \iff f_c \geq (\frac{1100^4 \cdot 0.99^2}{1-0.99^2})^{\frac{1}{4}} = 2914\text{Hz}$$

L'implémentation d'un filtre de Butterworth passif s'effectue avec un circuit RLC. Cependant, afin de ne pas avoir à utiliser d'inductance, il est préférable d'utiliser un filtre actif. Le filtre répondant au mieux à notre demande est le filtre passe-bas de Sallen et Key. De plus, celui-ci est très facile à réaliser.

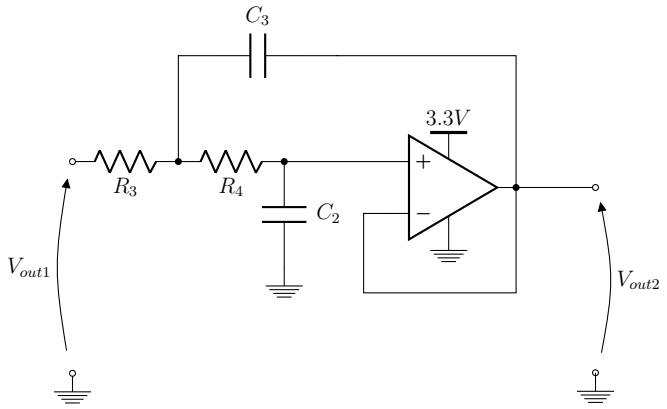


FIGURE *15 – Filtre Sallen et Key passe-bas d'ordre 2*

La fonction de transfert de ce filtre est

$$H_{SK}(j\omega) = \frac{\frac{1}{(j\omega)^2 C_2 C_3}}{R_3R_4 + \frac{1}{j\omega C_3}(R_3+R_4) + \frac{1}{C_2C_3(j\omega)^2}}$$

Pour faciliter la résolution, on choisit ces valeurs pour les composants : $R_3 = R_4$ et $C_3 = 2C_2$. On obtient alors pour l'amplitude de la fonction de transfert :

$$H_{SK}(j\omega) = \frac{\frac{1}{2C_2^2(j\omega)^2}}{R_3^2 + \frac{R_3}{C_2 j\omega} + \frac{1}{2C_2^2(j\omega)^2}} = \frac{1}{2C_2^2R_3^2(j\omega)^2 + 2R_3C_2j\omega + 1}$$

En posant $\omega_0 = \frac{1}{\sqrt{2}R_3C_2}$ on obtient :

$$H_{SK}(j\omega) = \frac{1}{1 + \frac{j\omega}{(\frac{\omega_0}{\sqrt{2}})} + (\frac{j\omega}{\omega_0})^2}$$

On a donc bien deux pôles en $f_1 = \frac{\omega_0}{2\sqrt{2}\pi}$ et $f_0 = \frac{\omega_0}{2\pi}$. La fréquence de coupure du filtre correspond donc au deuxième pôle $f_0 = \frac{\sqrt{2}}{4\pi R_3 C_2} = f_c$. L'amplitude de la fonction de transfert est :

$$|H_{SK}(j\omega)| = \frac{\omega_0^2}{\sqrt{(\omega_0^2 - \omega^2)^2 + (\sqrt{2}\omega_0\omega)^2}} = \sqrt{\frac{1}{1 + (\frac{\omega}{\omega_0})^4}}$$

Le filtre de Sallen et Key possède donc bel et bien une réponse en fréquence dont l'amplitude correspond à celle d'un filtre de Butterworth (Figure 19).

En faisant égaler ce pôle à la fréquence de coupure calculée précédemment on obtient donc :

$$f_c = \frac{\sqrt{2}}{4\pi R_3 C_2} \geq 2914\text{Hz}$$

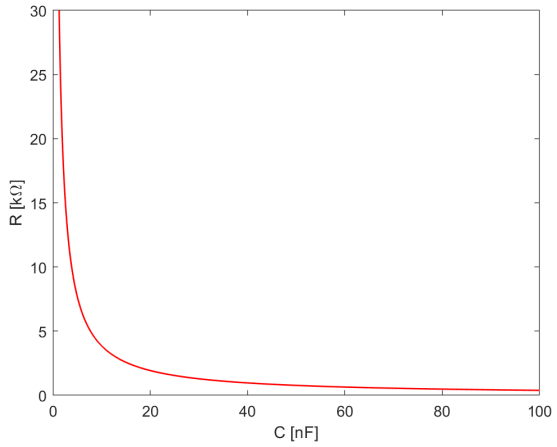


FIGURE *16 – Graphe de R_3 en fonction de C_2*

On choisit alors arbitrairement $C_2 = 10\text{nF}$ et on obtient comme valeur pour R_3 :

$$R_3 \leq \frac{1}{25893\, C_2} = 3.86\text{k}\Omega$$

3.5 Chaîne d'acquisition

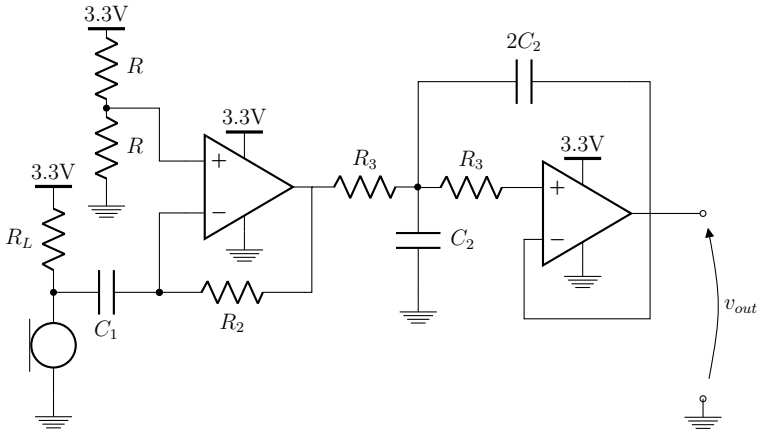


FIGURE *17 – Chaîne d'acquisition*

Après avoir testé le circuit expérimentalement, quelques petits ajustements ont été effectués. En effet, le gain de 1500 étant en réalité beaucoup trop grand et le signal saturant fortement, une résistance de 2.7MΩ, une résistance de 2.7kΩ et une capacité de 470nF ont été choisies à la place des valeurs théoriques.

	Valeurs calculées	Valeurs réelles
R_L	2.2kΩ	2.2kΩ
R	10kΩ	10kΩ
R_2	3.3MΩ	2.7MΩ
R_3	3.86kΩ	2.7kΩ
C_1	564.1nF	470nF
C_2	10nF	10nF

TABLE *1 – Valeurs des composants*

De ce fait, l'amplitude de la réponse en fréquence garde une forme similaire à celle désirée mais avec un gain moindre. (Figure 18).

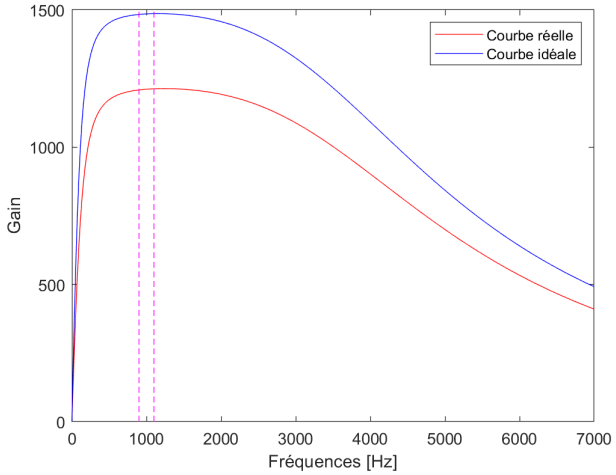


FIGURE *18 – Comparaison des réponses en fréquence réelle et théorique*

4 Démodulation du signal

4.1 Fréquence d'échantillonnage

Il est spécifié dans le cahier des charges que l'atténuation minimale des fréquences repliées est de $H_{LP} = 0.05$. Il faut donc trouver la fréquence à partir de laquelle l'atténuation correspond à cette valeur.

$$0.05 \geq \sqrt{\frac{1}{1 + (\frac{2\pi f_{min}}{2\pi 2914})^4}} \iff f_{min} \geq (\frac{1 - 0.05^2}{0.05^2})^{\frac{1}{4}} \cdot 2914 = 13\,024 Hz$$

A cette fréquence on ajoute la fréquence utile maximale. On obtient donc une fréquence d'échantillonnage minimale :

$$f_s \geq 13\,024 + 1100 = 14\,024 Hz$$

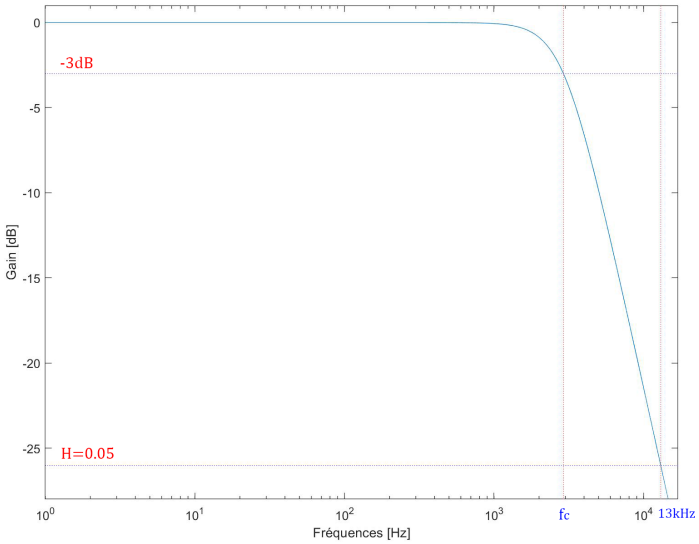


FIGURE *19 – Gain du filtre passe-bas en fonction de la fréquence*

La calcul de la valeur à donner au *timer* donne alors $\frac{F_{cy}}{f_s} = \frac{40\text{MHz}}{14.1\text{kHz}} = 2836.88$. On arrondit ce dernier à 2500 afin de faciliter les calculs. On obtient ainsi une fréquence d'échantillonnage de 16kHz.

4.2 Filtres numériques et démodulation

L'information binaire étant transmise grâce à une modulation FSK de fréquence porteuse $f_p = 1\text{kHz}$ et de déplacement de fréquence $\Delta f = 100\text{Hz}$, il est nécessaire de réaliser deux filtres passe-bande centrés chacun sur les fréquences de 900Hz et 1100Hz afin de démoduler le signal. Leur réalisation est effectuée grâce aux transformées en z et aux coefficients de la fonction de transfert.

$$H(z) = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

En effet, il est possible d'écrire la sortie $y[n]$ du filtre comme une équation aux différences avec ces coefficients :

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$$

Dans notre cas, un filtre à réponse impulsionnelle infinie (IIR) dont l'allure correspond à un filtre de Butterworth est choisi. Les filtres IIR ont comme particularité d'avoir une sortie dépendant à la fois des signaux d'entrée et des signaux de sorties précédents, contrairement aux filtres à réponse impulsionnelle finie (FIR) qui eux ne dépendent que des signaux d'entrées. Ils demandent également moins de temps de calcul et de mémoire que les FIR. Plusieurs implémentations de ces filtres existent, celle utilisée dans le cadre du projet est la *Direct Form II*. Celle-ci est en effet assez courante et facile à réaliser. Elle s'écrit :

$$\begin{cases} y[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2] \\ w[n] = x[n] - a_1 w[n-1] - a_2 w[n-2] \end{cases}$$

Un seul étage ne suffit cependant pas à filtrer correctement le signal, il faut donc combiner plusieurs filtres à la suite afin d'obtenir le signal désiré.

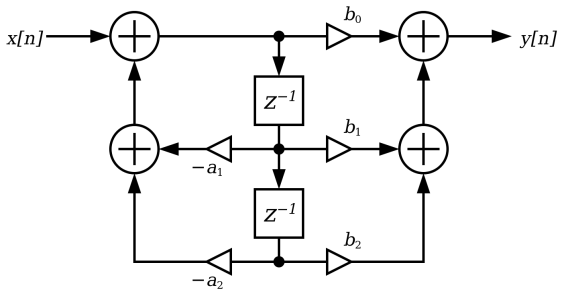


FIGURE *20 – Direct Form II* [wikipedia]

Trouver les coefficients correspondants peut se faire via de nombreux programmes. L'utilisation de l'outil *Filter Designer* de Matlab se révèle être fort efficace. En effet, il suffit de choisir le type de filtre, d'insérer les différentes fréquences de coupure et celui-ci nous renvoie une matrice SOS (*Second-Order Section*) contenant les coefficients de chaque étage et une matrice colonne G contenant les gains correspondants. Le cahier des charges nous indique que les fréquences utiles se situent dans un intervalle de $[f - 0.015 \cdot f, f + 0.015 \cdot f]$ et que les fréquences se situant au delà de $[f - 0.035 \cdot f, f + 0.035 \cdot f]$ doivent être coupées. De plus, l'atténuation minimale des fréquences coupées doit être de $H = 0.1$, ce qui correspond donc à 20dB. En entrant ces données dans *Filter Designer*, quatre étages de filtrage sont obtenus pour chacun des filtres.

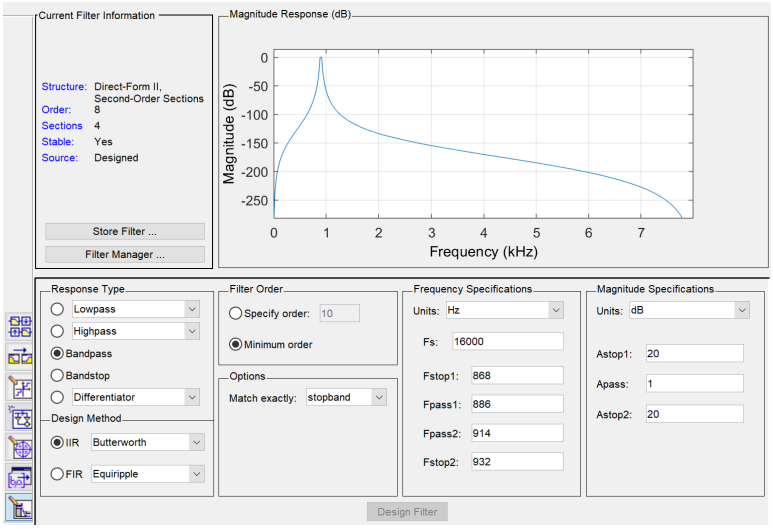
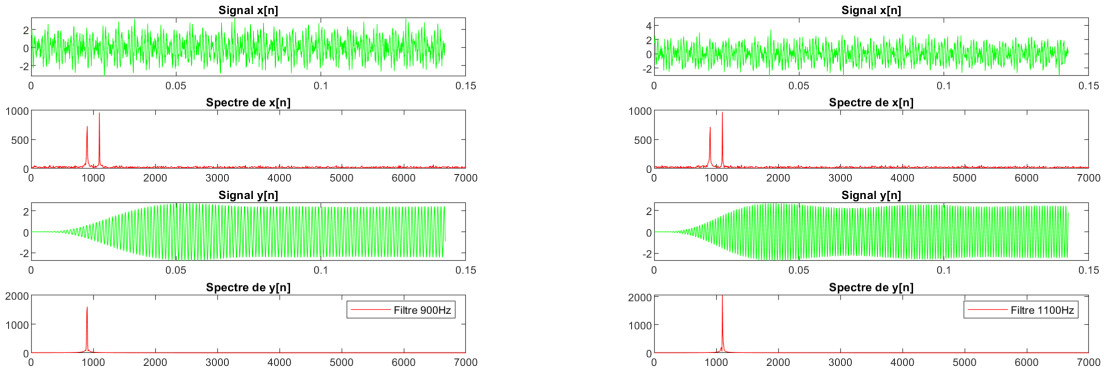


FIGURE *21 – Dimensionnement du filtre avec FilterDesigner*

Des simulations sont effectuées sur Matlab afin de vérifier l'efficacité des filtres sur un signal bruité contenant les deux fréquences (Annexe D). Ceux-ci fonctionnent bien selon nos attentes.



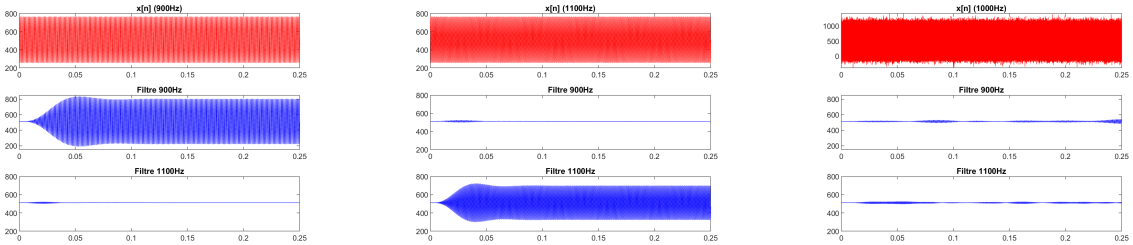
(a) Filtre 900Hz

(b) Filtre 1100Hz

FIGURE *22 – Simulation Matlab des filtres numériques*

L'implémentation de ces filtres sur le dsPic demande cependant quelques ajustements. En effet, ce dernier ne comprenant pas d'ALU dédiée aux calculs en virgule flottante (float), il est nécessaire de réaliser l'ensemble des opérations en nombre entiers (int ou long) afin de diminuer au maximum les temps de calculs. De ce fait, les coefficients et les signaux entrants sont multipliés par $2^{10} = 1024$ avant les calculs et les variables sont stockées sur 32 bits afin de s'assurer de ne pas avoir d'overflow. Il est également à noter que, afin de diminuer encore au maximum les temps de calculs, le multiplicateur ne doit pas être stocké dans une variable mais doit être écrit explicitement à chaque opération ou défini au début du code via un $\#define$.

Le code est ensuite testé grâce à la fonction Python de l'oscilloscope fournie et le résultat correspond bien aux simulation Matlab de la figure 22.



(a) Signal 900Hz (b) Signal 1100Hz (c) Signal 1000Hz bruité

FIGURE *23 – Test du code implémenté sur le dsPic*

4.3 Interprétation

Après avoir filtré le signal, il faut interpréter les réponses obtenues. En effet, si la réponse à un signal n'a pas été atténuée c'est que le signal comprend la fréquence correspondante. On vérifie donc qu'à chaque période du signal, l'amplitude de la réponse dépasse un certain seuil. L'échantillonnage et les calculs se faisant sur 12 bits, la valeur de la sortie est comprise entre 0 et 4095. De plus, un offset de 2048 est appliqué afin de capter l'entièreté du signal sortant. L'amplitude du signal atténué atteignant généralement 2080, les seuils pour le filtre de 900Hz et de 1100Hz sont placés respectivement à 2200 et 2170. Ces valeurs ont été trouvées expérimentalement.

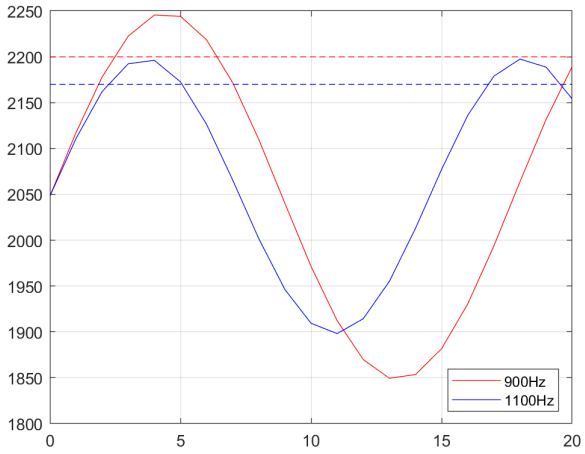


FIGURE *24 – Choix des seuils pour les signaux filtrés*

Il faut ensuite analyser chaque période des signaux filtrés. Sachant que l'échantillonnage se fait à 16kHz on peut calculer le nombre d'échantillons minimum :

$$\begin{cases} T_s = \frac{1}{16000\text{Hz}} = 62.5\mu\text{s} \\ T_{1100} = \frac{1}{1100\text{Hz}} = 0.9\text{ms} \Longrightarrow N_e \geq \frac{T_{1100}}{T_s} = 14.4 \\ T_{900} = \frac{1}{900\text{Hz}} = 1.11\text{ms} \Longrightarrow N_e \geq \frac{T_{900}}{T_s} = 17.7 \end{cases}$$

On choisit ainsi un nombre d'échantillons $N_e = 20$ afin de s'assurer de toujours avoir des périodes complètes pour les deux signaux mais également pour faciliter les calculs (Figure 24).

Le cahier des charges nous précise par la suite que chaque bit de donnée est émis pendant une durée de 100ms, on a donc une fréquence de transmission de 10Hz. Le nombre d'échantillons par bit est donc de $\frac{16000\text{Hz}}{10\text{Hz}} = 1600$. Il faut par après calculer l'*over sampling ration* (OSR). Étant donné qu'on vérifie tous les 20 échantillons si le seuil a bien été dépassé, le nombre de période à analyser par bit est $N_p = \frac{1600}{20} = 80$, ce qui correspond à l'OSR. Ces données sont à insérer dans la fonction FskDetector. Le dernier paramètre a régler est le nombre minimal d'échantillons similaires pour considérer que le bit est valide. Celui-ci est laissé sur son paramètre initial de 3/4 de l'OSR.

5 Analyse des résultats

La dernière étape consiste à faire le lien entre les deux micro-contrôleurs et ainsi de combiner les parties régulation et traitement sonore. Le baud rate [5] de chaque micro-contrôleur a tout d'abord dû être réglé sur la même valeur de 115200. Le code final fonctionne de cette façon : chaque fois qu'un message est entièrement détecté par la fonction fskDetector, le programme sort de la boucle principale et envoie ce message stocké dans 2 bytes au micro-contrôleur gérant le déplacement du robot. Le premier byte comprend la valeur du déplacement et le second contient l'ordre.

Les tests se sont avérés positif et le système fonctionnel. Le robot réagit la plupart du temps immédiatement aux signaux sonores envoyés, les quelques erreurs de réception dépendant du son ambiant et de la distance séparant l'émetteur sonore du robot. Quelques ajustements ont également dû être faits sur certains paramètres de la régulation tels que l'empattement et la marge d'erreur afin d'augmenter la précision.

6 Conclusion

Les objectifs du cahier des charges ont été atteints. Le son modulé est amplifié par la chaîne d'acquisition, filtré numériquement de manière à isoler les fréquences utiles, démodulé puis interprété par le robot qui va alors se déplacer selon la consigne.

Ce projet nous a permis de mettre en oeuvre de nombreuses connaissances informatiques, mathématiques et électroniques acquises ces dernières années. Nous avons ainsi pu appliquer certains principes théoriques tels que la régulation ou encore le dimensionnement d'une chaîne d'acquisition.

5. Le baud rate correspond à la vitesse de transmission des données

Annexe A

Fonction theoricMaxTime

```
float theoricMaxTime(float max, float a, float v, float time){
    float res;
    max = abs(max);
    if (time*v > max){
        res = sqrt(max/a)*2; //cas triangulaire
    }
    else {
        res = (2*time) +(max - (time*time*a))/v;
    }
    return res;
}
```

Annexe B

Fonction posCalc

```
float posCalc(float t, float max, float v, float a){
    max = abs(max);
    float time = v/a;   //temps de l'acceleration
    float tMax = theoricMaxTime(max, a , v, time);
    float res;
     if (time*v > max){ //cas triangulaire
        if (t <= tMax/2.0){
            res = (a/2.0)*t*t;
        }
        else{
            v = a*tMax/2.0;
            res = max-(a/2.0)*(tMax/2.0)*(tMax/2.0) + v *(t-(tMax-(tMax
    /2.0)))-(a/2.0)*(t-(tMax-(tMax/2.0)))*(t-(tMax-(tMax/2.0)));
        }
    }
     else{  //cas trapezoidale
        if (t<=time){
            res = (a/2.0)*t*t;
        }
        else if (t>time && t<(tMax-time)){
            res = (time*time*(a/2.0)) + v*(t-time);
        }

        else if (t>=(tMax-time)){
            res = max-(time*time*(a/2.0)) + v *(t-(tMax-time))-(a/2.0)*(t-(
    tMax-time))*(t-(tMax-time));
        }
        else {
            res = max;
        }
    }
    return res;
}
```

Annexe C

Code du spectre de la chaîne d'acquisition

```
%Valeurs théoriques
Rint= 2200
R2t = 3.3E6
C1t = 564E-9
R3t = 2730
C2t = 10E-9
%Valeurs réelles
Rin= 2200
R2= 3E6
R3= 2700
C1= 470E-9
C2= 10E-9

f = 0:15000;   %Plage de fréquences
w = 2*pi*f;

wo =1/(sqrt(2)*R3*C2);
wot=1/(sqrt(2)*R3t*C2t);

H1 = j*R2*C1*w./(1+Rin*w*C1*j);
H1t= j*R2t*C1t*w./(1+Rint*w*C1t*j);

H2=1./(1+sqrt(2)*j*w./wo+(j*w).^2./(wo)^2);
H2t=1./(1+sqrt(2)*j*w./wot+(j*w).^2./(wot)^2);

H=H1.*H2;    Ht=H1t.*H2t;

plot(f,abs(Hmod),'-r')
hold on
plot(f,abs(Hmodt),'-b')
xlabel("Fréquences [Hz]")
ylabel("Gain")
legend("Courbe idéale","Courbe réelle")
```

Annexe D

Code de test des filtres numériques

```
format long

fs = 16000;          %frequence d'echantillonnage
Ne = 1600;           %nombre d'échantillons

Ttot= Ne/fs;         %période totale d'écoute
t =  (1:Ne)*(1/fs);

mult = 1024;
multsquare = 1048576;

a91=[-1912,1018];
a92=[-1921,1019];
a93=[-1907,1011];
a94=[-1911,1011];
b91=[1024,0,-1024];
b92=[1024,0,-1024];
b93=[1024,0,-1024];
b94=[1024,0,-1024];
a09=7;
W91 = [0, 0, 0];
W92 = [0, 0, 0];
W93 = [0, 0, 0];
W94 = [0, 0, 0];

a111=[-1847,1017];
a112=[-1861,1018];
a113=[-1843,1008];
a114=[-1849,1008];
b111=[1024,0,-1024];
b112=[1024,0,-1024];
b113=[1024,0,-1024];
b114=[1024,0,-1024];
a011=8;

W111 = [0, 0, 0];
W112 = [0, 0, 0];
W113 = [0, 0, 0];
W114 = [0, 0, 0];

```

```
x = 512+ 256*sin(2*pi*900*t)+256*randn(size(t));   % on a alors x[n]
y9(1:Ne)=0;
y11(1:Ne)=0;

for i=1:Ne
    W91(1) = (x(i)*mult - a91(1)*W91(2) - a91(2)*W91(3))/mult;
    output1 = a09*(b91(1)*W91(1) + b91(2)*W91(2) + b91(3)*W91(3))/
   multsquare;

    W91(3)=W91(2);
    W91(2)=W91(1);

    W92(1) = (output1*mult - a92(1)*W92(2) - a92(2)*W92(3))/mult;
    output2 = a09*(b92(1)*W92(1) + b92(2)*W92(2) + b92(3)*W92(3))/
   multsquare;

    W92(3)=W92(2);
    W92(2)=W92(1);

    W93(1) = (output2*mult - a93(1)*W93(2) - a93(2)*W93(3))/mult;
    output3 = a09*(b93(1)*W93(1) + b93(2)*W93(2) + b93(3)*W93(3))/
   multsquare;

    W93(3)=W93(2);
    W93(2)=W93(1);

    W94(1) = (output3*mult - a94(1)*W94(2) - a94(2)*W94(3))/mult;
    output4 = a09*(b94(1)*W94(1) + b94(2)*W94(2) + b94(3)*W94(3))/
   multsquare;

    W94(3)=W94(2);
    W94(2)=W94(1);
    y9(i) = 512+output4;
end
for i=1:Ne
    W111(1) = (x(i)*mult - a111(1)*W111(2) - a111(2)*W111(3))/mult;
    output1 = a011*(b111(1)*W111(1) + b111(2)*W111(2) + b111(3)*W111(3))/
   multsquare;

    W111(3)=W111(2);
    W111(2)=W111(1);

    W112(1) = (output1*mult - a112(1)*W112(2) - a112(2)*W112(3))/mult;
    output2 = a011*(b112(1)*W112(1) + b112(2)*W112(2) + b112(3)*W112(3))/
   multsquare;

    W112(3)=W112(2);
    W112(2)=W112(1);

    W113(1) = (output2*mult - a113(1)*W113(2) - a113(2)*W113(3))/mult;
    output3 = a011*(b113(1)*W113(1) + b113(2)*W113(2) + b113(3)*W113(3))/
   multsquare;

    W113(3)=W113(2);
    W113(2)=W113(1);
```

```
89
90      W114(1) = (output3*mult - a114(1)*W114(2) - a114(2)*W114(3))/mult;
91      output4 = a011*(b114(1)*W114(1) + b114(2)*W114(2) + b114(3)*W114(3))/
        multsquare;
92
93      W114(3)=W114(2);
94      W114(2)=W114(1);
95
96      y11(i) = 512+output4;
97  end
98
99
100
101 ax1=subplot(3,1,1)
102 plot(ax1,t,x,'-r')
103 title("x[n] (900Hz+bruit)")
104 ax2=subplot(3,1,2)
105 plot(ax2,t,y9,'-b')
106 title("Filtre 900Hz")
107 axis([0 t(Ne) 150 850]);
108 ax3=subplot(3,1,3)
109 plot(ax3,t,y11,'-b')
110 title("Filtre 1100Hz")
111 axis([0 t(Ne) 200 800]);
```

Bibliographie

[1] Storey, Neil. 2017. *Electronics : A Systems Approach, 6th Edition*

[2] http://www.f-legrand.fr/scidoc/docimg/sciphys/electro/sallenkey/sallenkey.html

[3] https://uv.ulb.ac.be/pluginfile.php/1169222/mod_resource/content/0/KECG2740PBJ.pdf

[4] https://en.wikipedia.org/wiki/Electret_microphone

[5] https://fr.wikipedia.org/wiki/Microphone#Microphone_%C3%A9lectrostatique

[6] https://en.wikipedia.orgg/wiki/Sallen%E2%80%93Key_topology

[7] https://www.sciencedirect.com/topics/engineering/sallen-key

[8] http://res-nlp.univ-lemans.fr/NLP_C_M15_G03/co/Contenu_63.html

[9] http://www.eas.uccs.edu/~mwickert/ece5655/lecture_notes/ece5655_chap8.pdf

[10] http://www.t-es-t.hu/download/microchip/an852a.pdf