RAPPORT DE MODAL

PHY43M06EP: Robotique

FRAMBOURT MATEÏS

TABLE DES MATIÈRES

INTRODUCTION

Je suis arrivé dans ce modal un peu par hasard. Au départ je m'étais inscrit pour le ModAL FPGA car c'était quelque chose qui me semble très versatile mais un peu obscur pour le découvrir en auto-didacte. C'est donc avec surprise que j'apprend la disparition de ce ModAL dans une fusion avec le ModAL de Robotique. Au final, ça ne m'a pas trop géné car c'était mon deuxième voeu. J'avais une légère expérience en robotique. En effet, j'ai participé de 2012 à 2021 aux compétitions de robotique junior (robot radioguidés) de planêtes sciences et j'aide encore marginalement dans l'équipe familiale. J'avais donc une certaine connaissance des composants et de leurs utilisations mais je ne connaissais presque rien dans le contrôle et la commande haut niveau. Mon objectifs dans ce cours était donc d'apprendre cette partie pour pouvoir ensuite l'appliquer au Binet X-Robot qui est en manque crucial de RH et de compétence.
D'un point de vue plus académiques, l'objectif final est de réaliser un robot « tank » pour pouvoir travailler sur l'anticipation de mouvement de l'adversaire. l'objectif était donc de réussir à pointer une cible sur l'adversaire tout en se déplaçant dans un environnement encombré. Pour cela nous avions à notre disposition :

- Roues holonomes
- Moteurs avec encodeurs
- Raspberry pi 4
- Cartes moteurs I²C Groves
- Caméra grand angle 5MP
- Servomoteurs

Cependant, avant de réaliser le projet final il est nécessaire de réaliser un suivi de ligne fonctionnel. Dans ce sens j'ai passé (malheureusement) la majeur partie des séances (8/10) sur ce projet initial.

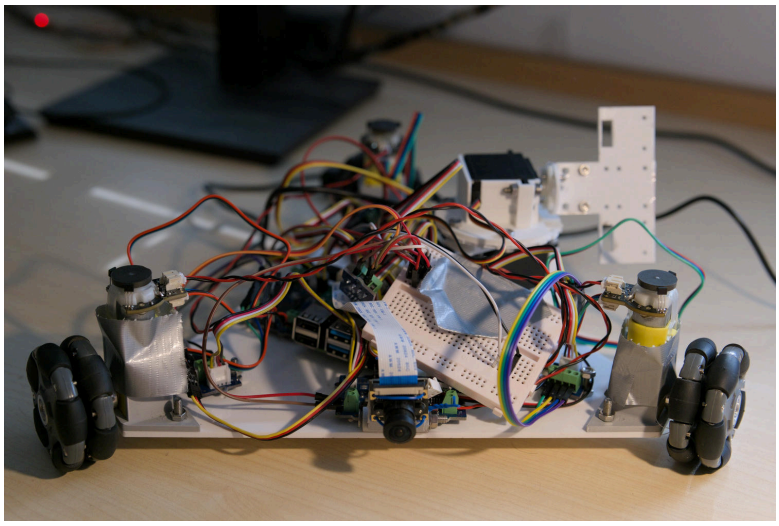

Fig. 1. – Vue d'ensemble du robot final

1
BASE COMMUNE

1.1 - Base mécanique

Dans l'optique de pouvoir réutiliser un maximum de composants (tant mécaniques que logiciels), J'ai commencé par modéliser précisément les composants à ma disposition. Ensuite j'ai dessiné les supports des moteurs pour avoir une emprise au sol minimale (ils ont ensuite été imprimés au DrahiX). Enfin j'ai dessiné la plaque principale qui devait permettre d'avoir tous les composants à plats avec l'ensemble des trous déjà percés (elle a été découpée au laser du DrahiX). Cependant, la breadboard et la batterie ont été oubliées ce qui a mené à un montage relativement complexe à suivre et à modifier. L'objectifs n'a donc pas complètement été rempli.
J'ai choisi d'utiliser 3 moteurs en triangles pour 2 raisons :

- Il y a 3 degrés de liberté sur le robot donc avoir 3 moteurs pourrait donc nous permettre de facilement faire de la cinématique inverse exacte (ce n'est pas la méthode qui a finalement été utilisée)
- Les placers à 120° permets d'avoir une symétrie et donc on peut avoir plusieurs « face avant » du robot et il suffit juste de faire un décalage au niveau des consignes pour utiliser par exemple un autre outil

D'autre part, la caméra est montée sur un pivot car elle possède un grand angle de vue. Il est donc possible de régler l'étendue du champ de vision en inclinant la caméra pour qu'elle ne soit pas perturbée par des objets extérieurs.

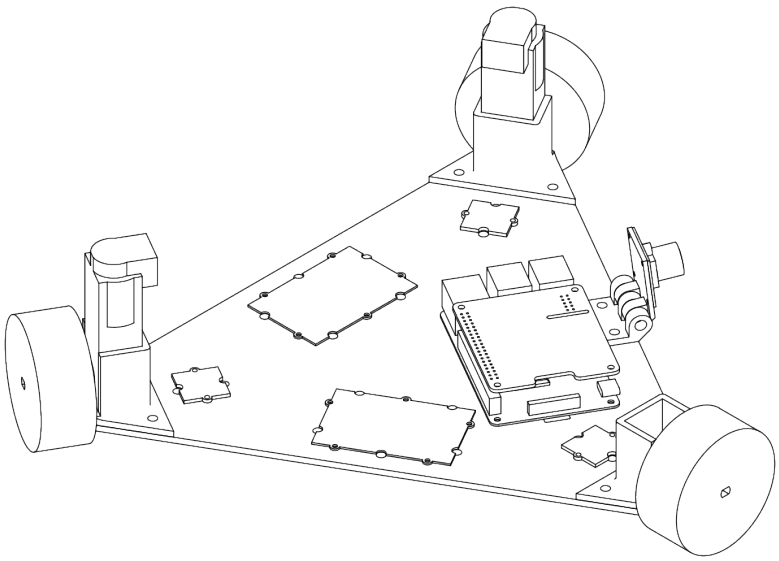

Fig. 2. – Vue isométrique de l'assemblage basique

1.2 - Encodeurs

Rapidement, les encodeurs sont des paires de capteur à effet hall en quadrature de phase autour d'un aimant à 14 pôles radiaux de manières à ce qu'ils donnent 7 périodes de la Fig. 3 par rotation. Pour interpréter le signal, j'ai fait une librarie CMake qui attache une interruption aux 2 pins de l'encodeur et appelle une fonction très simple :

```
uint8_t phase = identifier_phase();
if (phase == (lastPhase + 1) % 4)
{
    pos++;
}
else if (phase == (lastPhase - 1) % 4)
{
    pos--;
}
lastPhase = phase;
```

et pour identifier la phase on fait appel à une table de correspondance :

```
uint8_t const LUT[] = {0, 1, 3, 2};
return LUT[2 * digitalRead(pinA) + digitalRead(pinB)];
```

J'ai essayé pendant plusieurs séances d'implémenter cette librairie de manière conforme à la POO mais je n'arrivais pas à faire une fonction statique qui appelle une méthode. Finalement, j'ai réussi à l'implémenter pour une autre librairie dont je parlerais plus loin.
Globalement, le problème principal de cette mesure est qu'elle donne précisément la position mais pas la vitesse. Il faut ensuite dériver numériquement ce qui ne donne pas de très bon résultat car la position est une fonction en escalier. Or je souhaitais asservir les moteurs en vitesse ce qui rend l'asservissement un peu instable.

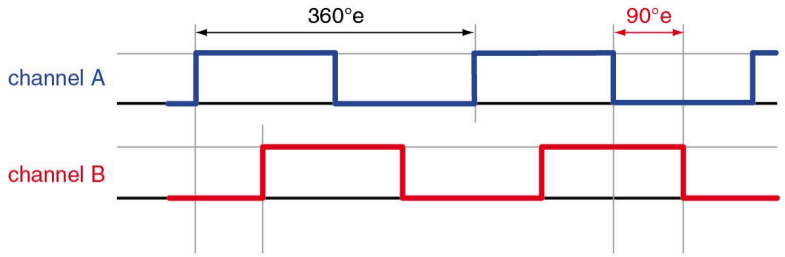

Fig. 3. – Signal en sortie des encodeurs (© Maxon)

1.3 - PID

J'ai ensuite créé une librairie CMake générique pour les PID, c'est-à-dire un contrôleur générique à trois composante (proportionnelle, intégrale, dérivée). Celui-ci devait permettre d'avoir un contrôle précis et réactif mais les premiers moteurs était trop peu puissant et ne reposait presque que sur la composante intégrale. En effet, même en rêglant la composante proportionnelle à la limite de l'instabilité, la valeur atteinte

en vitesse était très éloignée de la valeur désirée. D'autre part, l'entrée du moteur étant proportionnelle à sa vitesse, il fallait « intégrer » la commande envoyée au moteur. Dans le cas contraire, dès que l'erreur est nulle, la consigne envoyée au moteur est nulle et donc le moteur s'arrête. Ce n'est bien sûr pas le comportement voulu. Dans ce sens, l'ajout d'une commande constante au moteur permet au PID de compenser l'erreur que celle ci aurait seule. (On approxime donc le moteur à une réponse linéaire à la consigne et on corrige l'erreur d'approximation par PID). Cependant, même avec toute ces modifications, le moteur ne répondait pas correctement à faible commande. En effet, les effets de frictions sèches étaient trop important. Le modèle a donc été adapté pour les prendres en comptes (en augmentant la consigne artificiellement à faible commande). Pour limiter l'inertie de la composante intégrale, l'intégrale sature à partir d'une limite arbitraire.

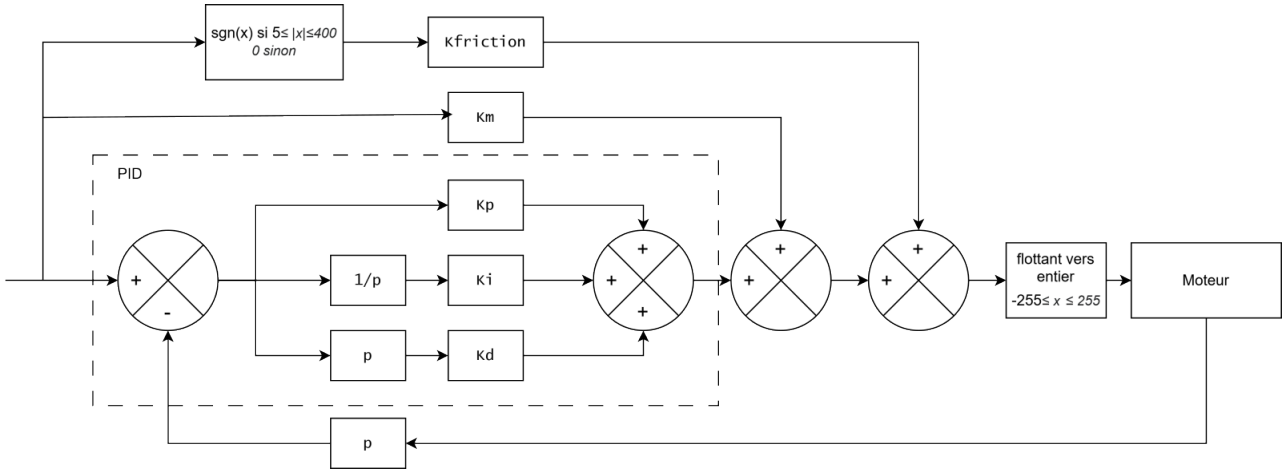

Fig. 4. – Asservissement de chaque moteur

1.4 - Roues holonomes

Les roues holonomes sont des roues qui n'imposent pas de contrainte normale à la roue. C'est dire qu'elle glisse sur la direction normale à la roue. On peut donc considérer uniquement comme un ajout de vecteur (il faut cependant prendre en compte que chaque roue adhère sur la direction tangentielle). Le robot est en liaison équivalente à une liaison plane, il y a donc 3 dégrés de libertés or on a 3 moteurs donc on pourrait écrire les équations directe du mouvement pour ensuite tenter des les inverser. Cependant ici, vu le faible nombre de moteur, j'ai choisi d'exhiber 3 vecteurs de bases du mouvement (Fig. 5). Par la suite, il est donc possible de partir directement des vecteurs du mouvement désirés. Pour autant, les moteurs ont une valeur maximale de vitesse de rotation possible. Il faut donc limiter le vecteur de commande de sorte à ce que chacune de ses composantes reste inférieures à la limite. En pratique, j'ai réalisé la transformation avec des flottants $\in [-1, 1]$. j'ai exploré 3 méthodes :

- Normaliser : $\boldsymbol{t_{\text{out}}} = \frac{\boldsymbol{t_{\text{in}}}}{\|\boldsymbol{t_{\text{in}}}\|}$ on a donc $\|\boldsymbol{t_{\text{out}}}\| = 1$ donc toute les composantes $\in [-1, 1]$
- Tronquer : pour tout composante $t_{\text{in}} \in \boldsymbol{t_{\text{in}}}$, $t_{\text{out}} = \min(\max(t_{\text{in}}, -1), 1)$ donc $-1 \leq t_{\text{out}} \leq 1$
- Mettre à l'échelle : soit $t_{\max} \in \boldsymbol{t_{\text{in}}}$ tel que $\forall t \in t_{\text{in}}, |t_{\max}| \geq |t|$ alors $\boldsymbol{t_{\text{out}}} = \frac{\boldsymbol{t_{\text{in}}}}{|t_{\max}|}$

L'avantage de tronquer, c'est que le calcul est extrêmement rapide mais on perd de la direction originelle du vecteur. La normalisation permet de conserver la direction mais réduit plus les composantes que nécessaire

(on est dans la boule unitaire) alors que mettre à l'échelle permet de conserver la direction (en étant dans le cube unité complet). Au final, l'algorithme choisi est donc :

```
float avant[3] = {0, -1, 1};
float droite[3] = {-1, 1 / sqrt2, 1 / sqrt2};
float rotation[3] = {1, 1, 1};
scale(input);
for (int i = 0; i < 3; i++)
{
    output[i] = avant[i] * input[0] + droite[i] * input[1] + rotation[i] * input[2];
}
scale(output);
```

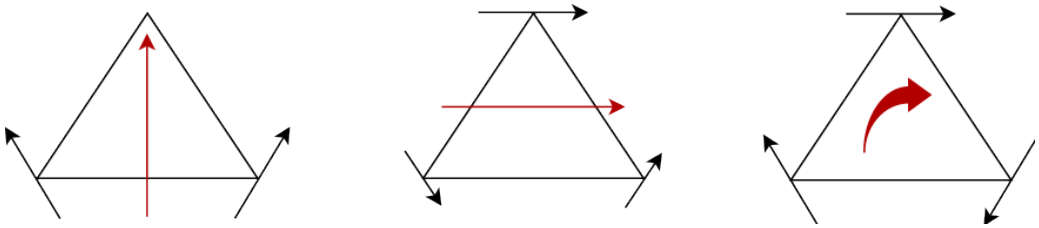

Fig. 5. – Vecteurs considérés pour les roues holonomes

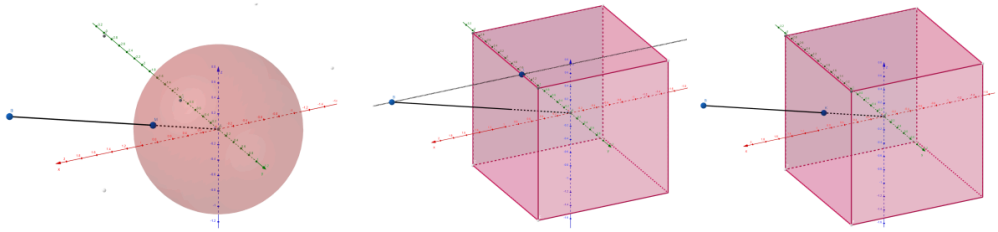

Fig. 6. – Les différentes méthodes de projections explorées

1.5 - Carte Moteur

Cette section est courte car l'utilisation est relativement simple mais a demandé du temps de calibration. En effet, les cartes recoivents 2 bytes pour les vitesses et 1 bytes pour les directions. Cependant, les valeurs semblent aléatoires :

```
uint8_t DirLut[4] = {0x0a,0x06,0x09,0x05};
wiringPiI2CWriteReg16(this->m_fd,0xaa,DirLut[(dirA+2*dirB)]);
```

et pour la vitesse :

```
this->setDirection(this->_M1_direction,this->_M2_direction);
wiringPiI2CWriteReg16(this->m_fd,0x82,((uint16_t)this->_speed1)<<8|this->_speed2);
```

On peut donc remarquer que c'est ici qu'intervient la limitation à 8 bit par moteur qui était affiché sur la Fig. 4

2
ROBOT SUIVEUR

2.1 - Estimation du chemin

Le chemin est un scotch rouge ou bleu faisant un cycle sans embranchement. On peut donc essayer d'estimer le chemin en ne prennant que le barycentre des pixels de la bonne couleur. Dans un premiers temps j'ai essayé de prendre le barycentre bleu et rouge sur toute l'image mais le résultat était trop influencé par les pixels proches de la caméra et le robot oscillait. J'ai donc séparé l'image en 2, une partie supérieure et inférieure sur lesquels on calcule le barycentre des 2 couleurs. Pour cela, on retourne l'image et on convertit l'image de RGB vers HSV pour avoir accès à la teinte qui est plus facile à interprêter. Ensuite, on filtre en ne gardant que les pixels entre 2 bornes. Enfin on calcule les moments pour enfin avoir les barycentres. les moments sont $M_{\text{ij}} = \sum_{p\in \text{ image}}(x^i y^j p)$ donc $\mathbf{bary} = \begin{pmatrix} \frac{M_{10}}{M_{00}} \\ \frac{M_{01}}{M_{00}} \end{pmatrix}$

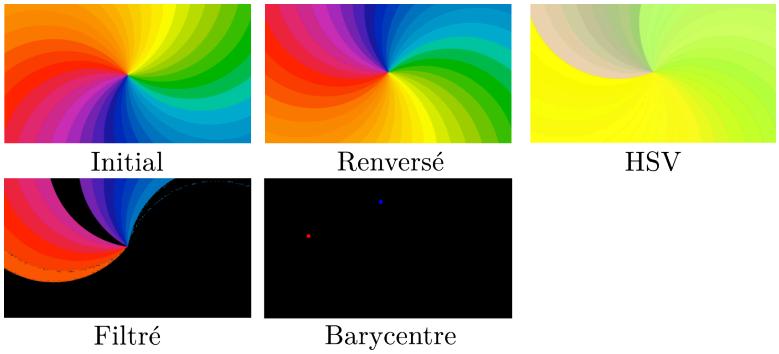

Fig. 7. – Filtre appliqué

2.2 - Choix entre bleu et rouge

Il faut ensuite choisir s'il faut suivre les barycentres bleus et rouges. Pour cela, j'ai considéré qu'on pouvait estimer comme fiable s'il y a plus de 2% de pixel de l'image sont de cette couleurs. Il est nécessaire d'avoir un hysterésis pour ne pas être sensible à un verrouillage bref d'un obstacle. Au final, j'ai utilisé une machine à 3 états : Soit on suis le rouge, soit le bleu soit le robot est perdu.

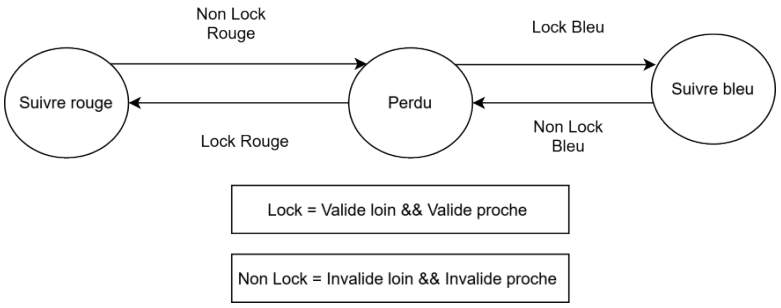

Fig. 8. – Logique de choix des couleurs

2.3 - Loi de commande

Le robot a maintenant 2 barycentres sélectionnés qu'il doit suivre. Il y a 3 degrés de libertés à commander. j'ai choisi de mettre directement la commande « avancer » à 0.5 pour donner une direction globale. Si le robot est perdu il tourne en alternant le sens de rotation pour parcourir un arc de plus en plus grand pour trouver le chemin le plus proche de la direction initiale. Dans le cas contraire, on vérifie si on un verrouillage proche alors on prend en compte ce barycentre et on applique un PID pour maintenir le barycentre au centre horizontalement. D'autre part, si il y a un verrouillage loin valide, alors on le prend en compte et on calcule l'angle entre le centre du bord bas de l'image et le barycentre avec l'horizontale. On applique alors un PID sur l'angle pour le maintenir à 90°. Pour résumer en un schéma :

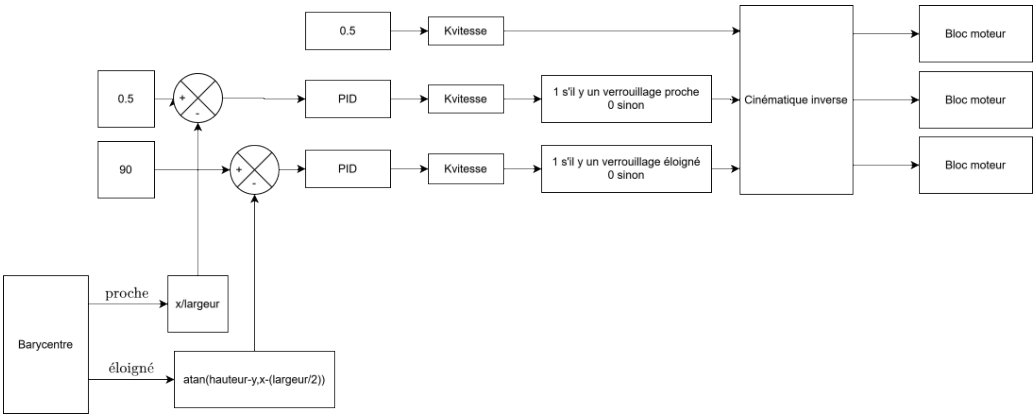

Fig. 9. – Schéma de la loi de commande

3
ROBOT TANK

3.1 - Ajout d'une tourelle

3.2 - état de l'art

[1]

CONCLUSION

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

BIBLIOGRAPHIE

[1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, et P. Wang, « Path planning techniques for mobile robots: Review and prospect », *Expert Systems with Applications*, vol. 227, p. 120254, 2023, doi: https://doi.org/10.1016/j.eswa.2023.120254.

ANNEXE A
FICHE D'ÉVALUATION DU STAGIAIRE

Yeah j'ai eu que des A partout trop bien, je suis un.e super stagiaire.