[**Test des Moteurs 1**](#_w6p2jndkyu3)

[Principe de fonctionnement 1](#_8zty3i1fcaio)

[Caractéristique Moteur 3](#_nf9ly4lfocyn)

[Présentation 4](#_txxhoadz68zj)

[Spécification: 4](#_ctncp4ja9avj)

[Mode de câblage de l'encodeur: 4](#_4vyihv1q8hh3)

[Elimination du couple de frottement 5](#_lhdqaub3e4a2)

[**Test Encodeur 5**](#_dyxvxmrg9z6l)

[Principe de fonctionnement des encodeur 5](#_310z2h5nfez3)

[**Mise en place du correcteur proportionnelle 7**](#_4xwe6xofbcr1)

[**Mise en place du correcteur proportionnelle dérivée 8**](#_j733nn421iki)

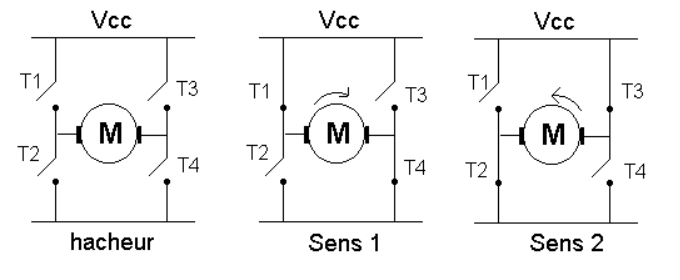
## ***Test des Moteurs***

### Principe de fonctionnement

La méthode de contrôle des moteur est effectuer avec une commande unipolaire dont le principe de fonctionnement est le suivant :

* En ajustant la tension fournie au moteur à l'aide de la commande unipolaire, on peut contrôler la vitesse du moteur. Une tension plus élevée accélère le moteur, tandis qu'une tension plus basse le ralentit.
* Elle se base sur le principe largement utilisé dans le monde de l’industrie qui est la commande par pont de transistors : Le Hacheur 4 quadrants.
* Pour faire tourner la roue dans un sens, on fait varier la commande entre 0 et 0.5.
* Pour faire tourner la roue dans le sens contraire au sens indiqué précédemment, on fait varier la commande entre 0.5 et 1.

Voici un schéma permettant de visualiser le principe d’un hacheur 4 quadrant :

Quatre transistors sont ici représentés par des interrupteurs T1, T2, T3 et T4, qui sont montés en pont et permettent de commander le sens de rotation du moteur : lorsque T1 et T4 sont fermés (saturés), le moteur tourne dans un sens (sens 1) . Lorsque T2 et T3 sont fermés, le moteur va tourner dans l'autre sens (sens 2).

Dans mon cas à moi pour pouvoir utiliser ses moteur je passe par l’utilisation d’un L298 mini qui à pour rôle de hacher la tension/courant en fonction de la MLI envoyé.

Avec pour unité de traitement l’ESP32, il faut dans un premier temps initialiser les broches en des sorties MLI, avec la fréquence souhaitée et la résolution de la MLI souhaiter.

// Config Moteur

ledcSetup(canal\_moteur\_droit, frequence, resolution);

ledcSetup(canal\_moteur\_gauche, frequence, resolution);

// Connexion entre les pins et les canaux

ledcAttachPin(pinPWM\_A\_MoteurDroit, canal\_moteur\_droit);

ledcAttachPin(pinPWM\_B\_MoteurDroit, canal\_moteur\_droit\_bis);

// Connexion entre les pins et les canaux

ledcAttachPin(pinPWM\_A\_MoteurGauche, canal\_moteur\_gauche);

ledcAttachPin(pinPWM\_B\_MoteurGauche, canal\_moteur\_gauche\_bis);

L’ESP32 dispose de 16 canaux de 0 à 15, pour les PWM. Il est possible de

configurer la fréquence et la résolution des canaux (de 1 à 16 bits). Les canaux

sont associés par 2, c’est-à-dire que cannal\_moteur\_droit et cannal\_moteur\_droit\_bis auront obligatoirement la même fréquence et la même résolution.

Dans mon cas, je travaille avec une résolution de 12 bits, ce qui donne une valeur de rapport cyclique comprise entre 0 et 4095 et une fréquence de 19 500 Hz.

En utilisant un potentiomètre créé à partir de l’application Bluetooth electronics, je fais varier sa valeur entre 0 et 4095.

Lorsque le slider émet la valeur 2048, le moteur ne tourne dans aucun des deux sens.

Lorsque le slider émet la valeur 4000, le moteur tourne dans le sens des aiguilles d’une montre car la variable pwm vaut 4000 et reverse\_pwm vaut 95.

Lorsque le slider émet la valeur 95, le moteur tourne dans le sens contraire des aiguilles d’une montre car la variable pwm vaut 95 et reverse\_pwm vaut 4000.

pwm = slider;

reverse\_pwm = 4095 - pwm;

ledcWrite(canal\_moteur\_droit, pwm);

ledcWrite(canal\_moteur\_droit\_bis, reverse\_pwm);

Cela permet de bien le fonctionnement citer précédemment “En ajustant la tension fournie au moteur à l'aide de la commande unipolaire, on peut contrôler la vitesse du moteur. Une tension plus élevée accélère le moteur, tandis qu'une tension plus basse le ralentit.”

Datasheet hacheur :

<https://www.datasheethub.com/mx1508-mini-dual-channel-1-5a-dc-motor-driver-module/#google_vignette>

### Caractéristique Moteur

Une image contenant Pièce auto, outil

Description générée automatiquementUne image contenant connecteur, fils électriques, câble

Description générée automatiquement

#### Présentation

##### Spécification:

| **Modèle** | **Encodeur de moteur à engrenages JGA25-370** |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| **Diamètre Roue** | **65** | **mm** |  |  |  |  |  |  |  |  |
| **Diamètre de l'arbre** | **4** | **mm** |  |  |  |  |  |  |  |  |
| **arbre D** | **3,5** | **mm** |  |  |  |  |  |  |  |  |
| **Tension nominal** | **12** | **V** |  |  |  |  |  |  |  |  |
| **Vitesse à vide (tr/min)** | **12** | **26** | **35** | **60** | **77** | **130** | **170** | **280** | **620** | **1360** |
| **Spécification du terminal** | **PH 2,0** |  |  |  |  |  |  |  |  |  |
| **Masse** | **90** | **g** |  |  |  |  |  |  |  |  |

##### Mode de câblage de l'encodeur:

Rouge: alimentation du moteur + (l'échange avec le blanc permet une rotation avant et arrière)

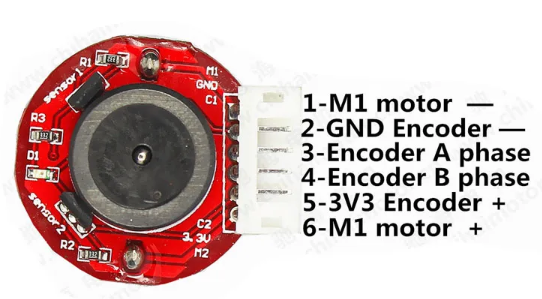
Noir: alimentation de l'encodeur-(les pôles positif et négatif du moteur ne doivent pas être mal connectés) 3.3-5V

Jaune: retour de signal (11 signaux lorsque le moteur tourne pour un cercle)

Vert: retour de signal (11 signaux lorsque le moteur tourne pour un cercle)

Bleu: alimentation de l'encodeur + (les pôles positif et négatif du moteur ne doivent pas être mal connectés) 3.3-5V

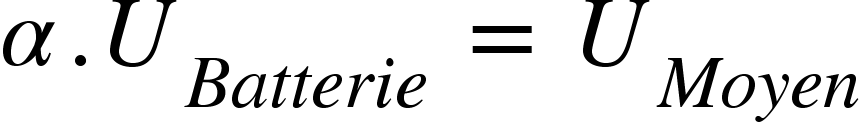
Blanc: alimentation du moteur-(l'échange avec le rouge permet une rotation avant et arrière)



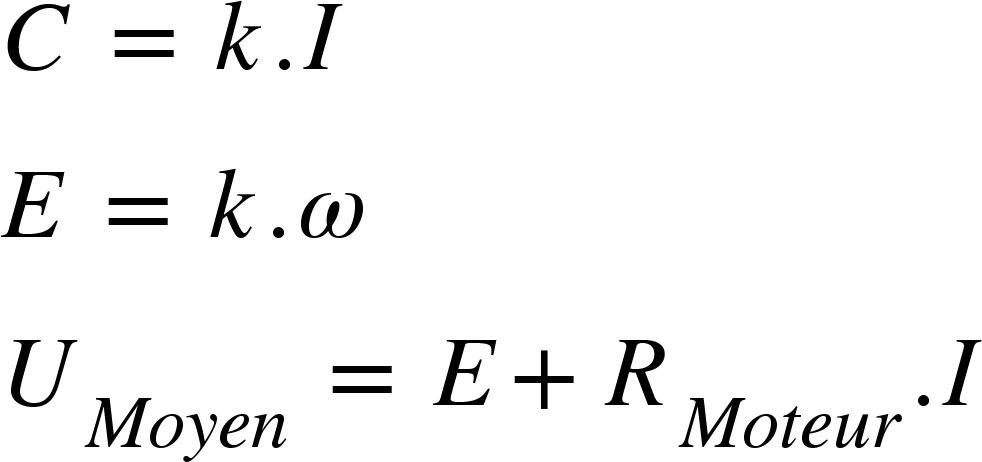
#### Elimination du couple de frottement

Comme tout système mécanique, des frottements sont présents au sein du système mais plus précisément des frottements sec et visqueux.

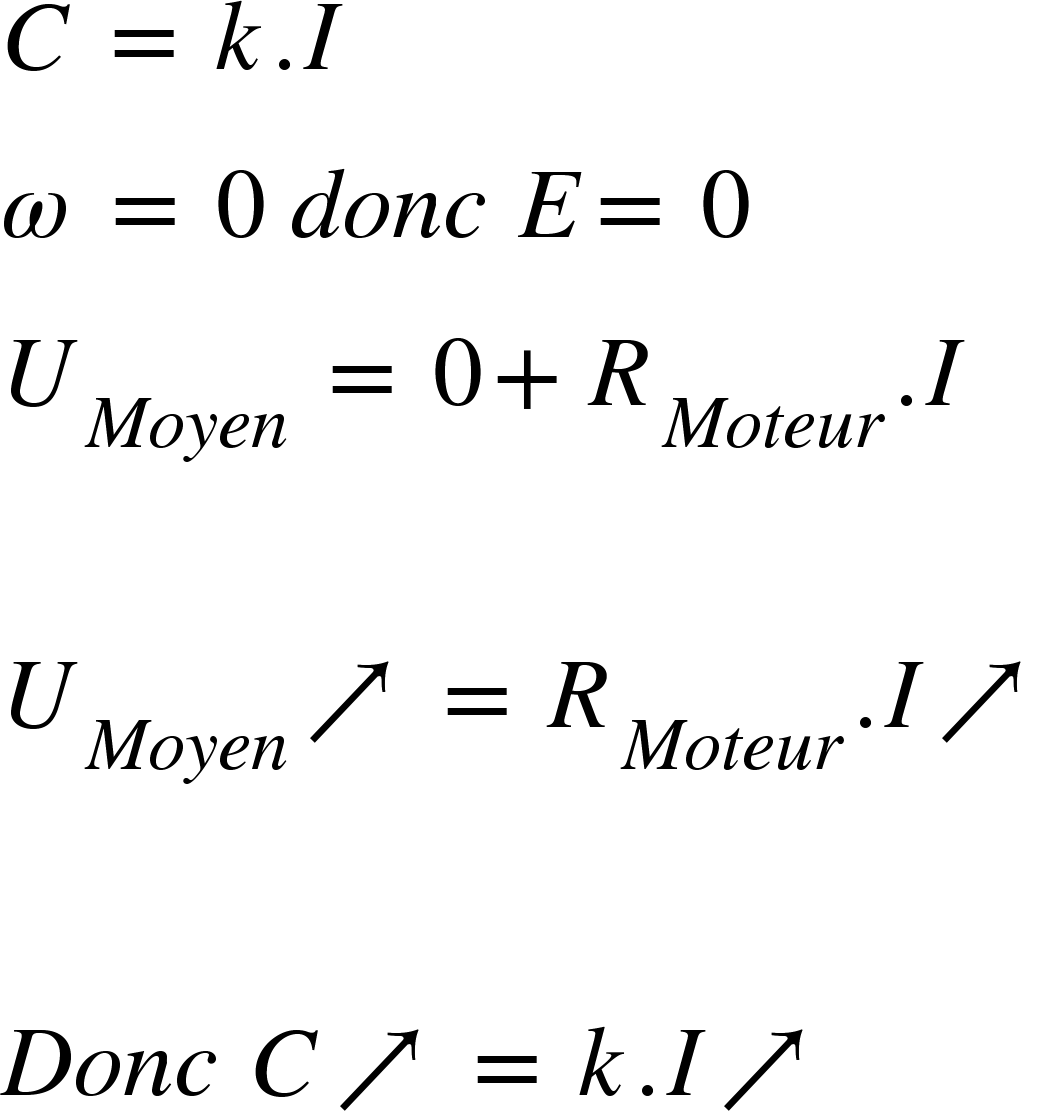
Etant donnée que je suis équipé d’une batterie 7.2V et que la tension nominale du moteur est de 12V, on travaille donc avec seulement 60% des capacité des moteurs. Lorsque je vais faire varier la MLI, on varie donc la tension moyenne car



De ce fait étant donnée que les équations de base d’un moteur DC sont les suivantes :



On peut sentir que lorsqu’on augmente la tension moyenne, le courant va augmenter, du fait que la vitesse de rotation vaut 0 à t=0. On a donc :



Cela paraît LOGIQUE, j’augmente la tension moyenne, j’augmente le courant bande de trou du Q.

Pour pouvoir limiter ce problème, on crée un potentiomètre sur le terminalMécatro, je fais varier la MLI dans un sens, pour une MLI de 2964 en partant du point neutre qui dans mon cas est de 2048 j’obtiens une commande de compensation d’environ 0.22.

## Test Encodeur

### Principe de fonctionnement des encodeur

Prenons l’exemple d’un encodeur incrémental, un encodeur incrémental est un dispositif qui mesure les mouvements à l’aide de capteur optique et fournit une série de signaux pulsés en sortie. Ces signaux sont généralement deux, A et B, déphasés de 90 degrés qu’on nomme quadrature. Les signaux A et B sont générés par un disque rotatif généralement un codeur optique. Ces signaux sont déphasés de 90 degrés l'un par rapport à l'autre. Lorsque l'axe tourne, les transitions de ces signaux sont utilisées pour détecter la direction et la direction de la rotation.

Les encodeurs incrémentaux mesurent les changements de position par rapport à une

référence, souvent à partir d'une position de départ connue, ou initialisée. En comptant les transitions des signaux A et B, le système peut déterminer le nombre d'impulsions et donc la rotation de l'axe. En observant la séquence des transitions entre les signaux A et B, le système peut également déterminer la direction dans laquelle l'axe tourne. Par exemple, si A passé de 0 à 1 avant B, l'axe se tourne dans le sens des aiguilles d'une montre, et inversement.

Au niveau de la programmation on inclura la bibliotheque ESP32Encoder :

Dans un premiers temps, on initialise avec les pin et variable des encodeurs :

ESP32Encoder::useInternalWeakPullResistors = UP;

unsigned long encoderDroitlastToggled;

bool encoderDroitPaused = false;

float valeurcodeurdroit, valeurcodeurgauche;

ESP32Encoder encoderGauche;

ESP32Encoder encoderDroit;

encoderDroit.attachHalfQuad(32, 33);

encoderGauche.attachHalfQuad(25, 26);

encoderDroit.clearCount();

Après avoir initialisé on peut récupérer les valeurs des encodeurs :

valeurcodeurgauche = encoderGauche.getCount();

valeurcodeurdroit = encoderDroit.getCount();

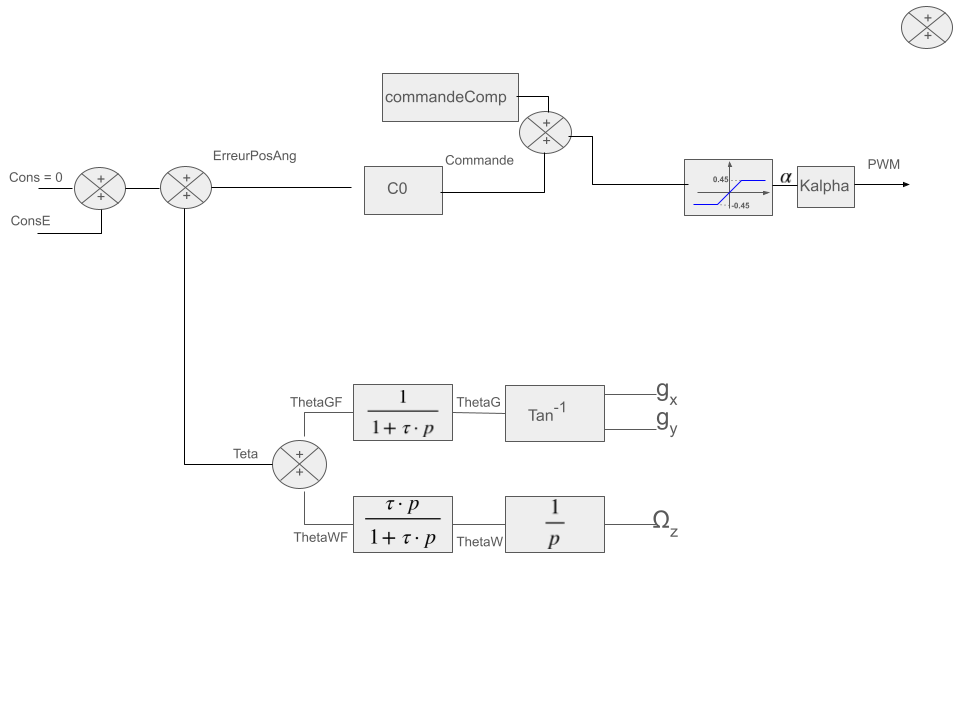
Lorsqu’on fait tourner les roues dans un sens, on peut lire des valeurs qui évoluent mais qui est toujours incrémenter de +1 ou -1 en fonction du sens de fonctionnement.

## 

## Mise en place du correcteur proportionnelle

Pour mettre en place le correcteur proportionnel, on crée des variables qu’on place en globale pour qu’il soit lu peu importe où on est dans le programme.

Les variables crée sont ceux présent dans le schéma fonctionnel ci-dessous :



Les étapes du correcteur proportionnel sont :

Calcul de l’erreur donc (cons+consE)-Teta,

On amplifi cette erreur à l’aide du correcteur proportionnelle

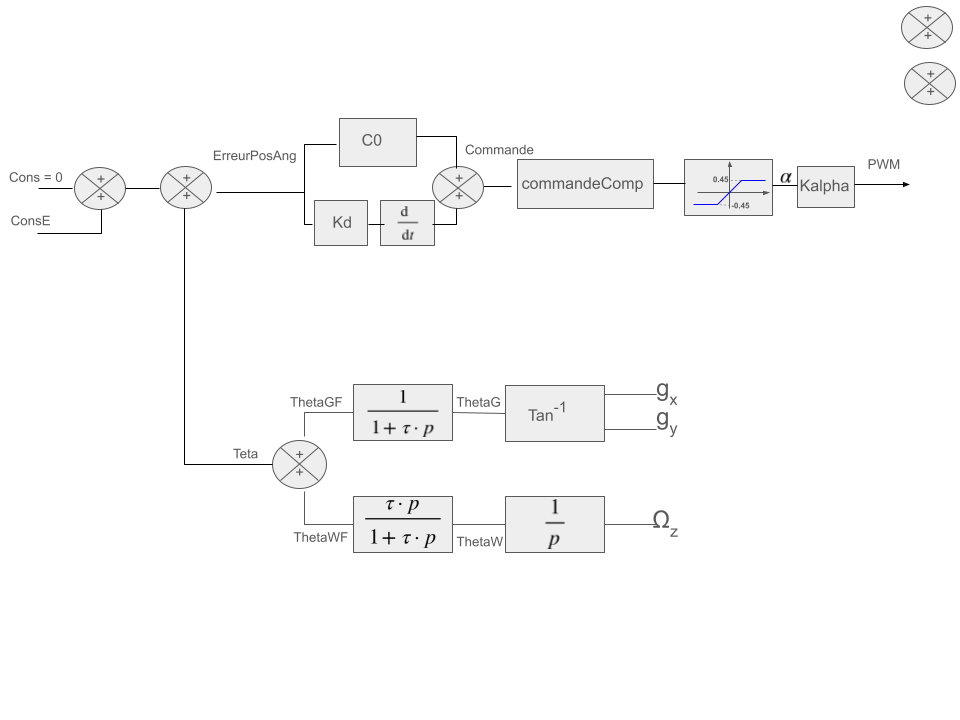
A la commande on ajoute la compensation permettant aux moteurs poue pouvoir se mettre en route

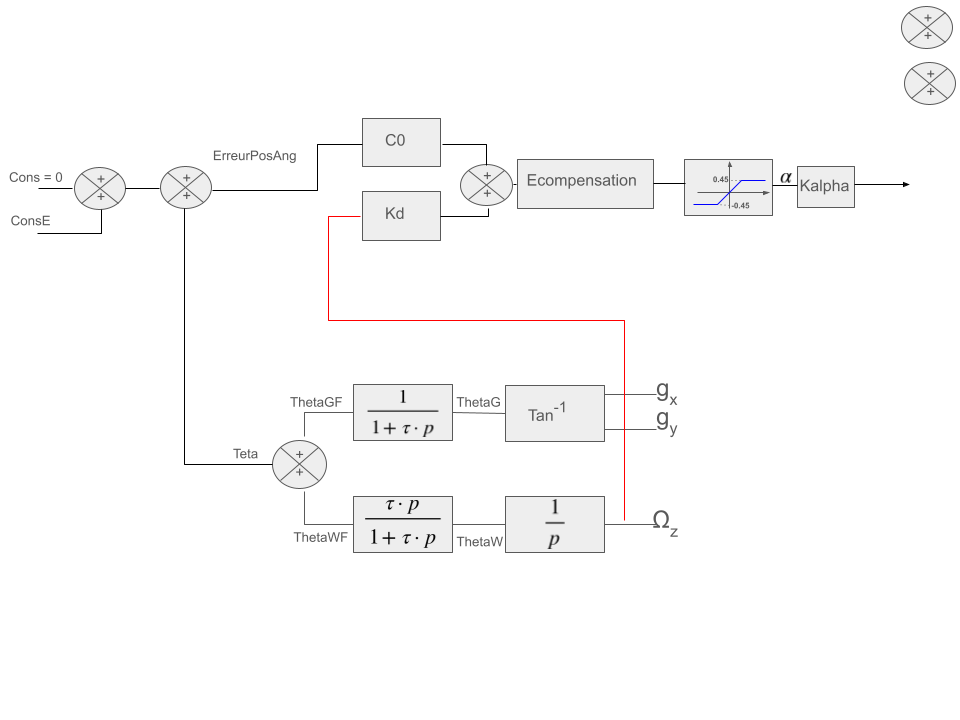
Si la commande est saturée, au-dessus d’une valeur prédéfinie qui est de 0,45 ou -0,45, on lui alloue une valeur fixe.

## 

## Mise en place du correcteur proportionnelle dérivée élémentaire

Lorsqu’on ajoute le correcteur proportionnelle dérivée élémentaire en parallèle du correcteur proportionnelle, on obtient ceci. La valeur qu’on va dérivée est l’erreur de l’angle, au lieu de recalculer d/dt, on va directement prendre la valeur d’omega z. On modifiera le programme de telle manière à ce qu’il ressemble au schéma fonctionnelle numéro 3.





L’ajout qu’on effectue au niveau programmation est la suivante :

commande = ErreurPosAng \* Co + Kd \* (-1) \* g.gyro.z;

J’effectue un \*-1 car si on laisse la valeur brute l’alimentation sécurisée va devenir un transistor du fait que je demanderais au robot d’aller dans le sens opposé à celui que je souhaite, de ce fait il ne tiendra pas debout.

Après l’ajout du PDE :

* Donner à Co une valeur grande
* Augmenter Kd
* Jouer avec les deux variables jusqu’à voir le résultat escompté