TRAVAUX PRATIQUES
COMMANDE NUMERIQUE

Polytech Lille : SE3

Intervenants :

M. Pekpe (Midzodzi.Pekpe@univ-Lille.fr)
R. Merzouki (rochdi.merzouki@polytech-lille.fr)

Polytech Lille
Cité Scientifique, Avenue Langevin 59655 Villeneuve d'Ascq Cedex.
Tél. 33 (0) 3 28 76 73 00
Fax 33 (0) 3 28 76 73 01

Polytech Lille-Département SE-3ère année

TRAVAUX PRATIQUES DE COMMANDE NUMERIQUE

Les travaux pratiques d'automatique linéaire sont organisés en 4 séances de 4 heures chacune, auxquelles s'ajoute une séance de contrôle de 2 heures.

Les TP de la commande numérique ont pour but d'illustrer l'enseignement portant sur les systèmes continus linéaires. Les méthodes d'analyse et de synthèse seront utilisées pour observer, analyser des systèmes réels ou simulés, ainsi que pour les corriger et améliorer leurs performances.

A la fin de chaque séance, chaque binôme remet à l'enseignent un compte-rendu écrit. La moyenne des quatre notes de compte-rendu représente la moitié de la note finale, dont l'autre moitié est la note de contrôle.

La liste des TP est la suivante :
TP1 : Vers la commande d'un Robot Mobile omni-directionnel 'Robotino'
TP2 : Régulation de vitesse d'un moteur à courant continu;
TP3 : Modélisation et commande d'un robot mobile.
TP4 : Régulation numérique : Simulation avec une étude de cas sur un système électromécanique.

Chaque binôme effectue les 3 premiers TP. Les rotations sont organisées de la manière suivante :

Binômes	Séance 1 (4 heures)	Séance 2 (4 heures)	Séance 3 (4 heures)	Séance 4 ◊
B1	TP1	TP2	TP3	TP4
B2	TP1	TP2	TP3	TP4
B3	TP2	TP3	TP1	TP4
B4	TP2	TP3	TP1	TP4
B5	TP3	TP1	TP2	TP4
B6	TP3	TP1	TP2	TP4

La page suivante donne la répartition nominative des séances 1 à 3.
Lors de la 4ème séance, tous les étudiants réaliseront un même TP en simulation, il s'agit du TP4.

Intervenants :
M. Pekpe (Midzodzi.Pekpe@univ-Lille.fr);
R. Merzouki (rochdi.merzouki@polytech-lille.fr);

TP N°1

Vers la commande d'un Robot Mobile Omni-directionnel 'Robotino'
Version : RobotinoView

Objectifs du TP

- Identification les systèmes de mesures embarqués sur le robot mobile ;
- Etude de la régulation en vitesse d'un robot mobile omnidirectionnel.

Description générale du robot

Le Robotino® est un système robotique mobile moderne de haute qualité. Trois roues omnidirectionnelles lui permettent de se déplacer dans toutes les directions du plan ainsi que de tourner sur lui-même. Le choix de roue omnidirectionnelle permet de réduire les forces de contacts lors de déplacement complexe. Le Robotino® est autonome, ces nombreux capteurs, sa webcam ainsi qu'une puissante unité de commande confèrent au système l'intelligence nécessaire pour résoudre de façon autonome les problèmes qui lui sont posés.

L'accès à l'unité de commande peut se faire soit en connectant un écran et un clavier directement sur le robot, soit en prenant le contrôle à distance via une connexion réseau.

L'ajout de capteurs ou d'actionneurs supplémentaires peut être envisagé grâce à l'interface d'entrée-sortie.

Le robot dispose d'un socle afin de le poser quand il n'est pas utilisé ou quand il est en charge. Ce socle est conçu de façon à pouvoir vérifier le fonctionnement des capteurs et des actionneurs sans risque de collisions. En effet les roues restent libres et n'entre pas en contact avec le sol.

Figure 1.1 : Robotino®

Description des composants du robot

Châssis

Le châssis est une plateforme d'acier soudé au laser. Les batteries, les unités de déplacement et la webcam sont fixés sur le châssis ainsi que les capteurs de mesure de la distance et d'anticollision. Le châssis offre de l'espace supplémentaire pour l'ajout éventuel d'autres capteurs et/ou actionneurs. Le déplacement manuel du Robotino® se fait par ses poignées de transports (1) de la Figure 1.2.

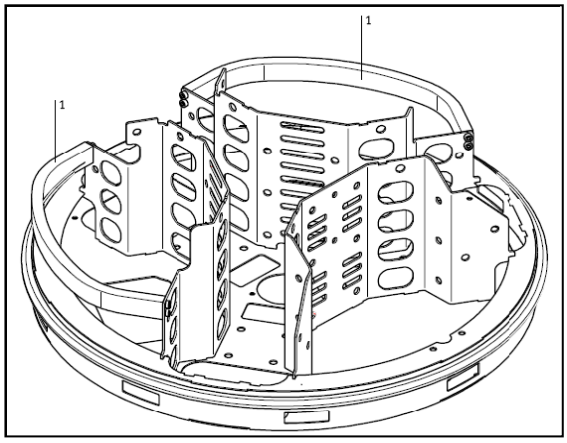


Figure 1.2 : Châssis du robot®

Unité de déplacement

Le robot est muni de 3 unités de déplacement (Figure 1.3), implantés à 120° l'une de l'autre sur un châssis circulaire, qui lui permet de se déplacer selon 3 axes (3 degrés de liberté) :

- Mouvement selon l'axe longitudinal ;
- Mouvement selon l'axe latéral ;
- Mouvement de lacet.

La vitesse réelle du moteur peut être comparée à la vitesse désirée grâce au codeur incrémental et peut être régulée avec un contrôleur PID. Le déplacement du robot s'effectue avec l'asservissement en parallèle des 3 unités.

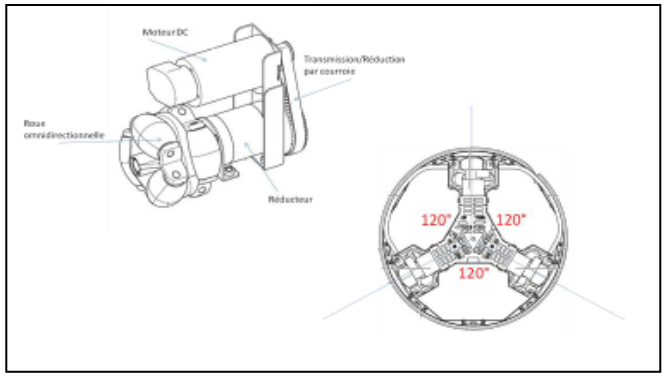


Figure 1.3 : Roues omnidirectionnelles

Système de vision

Le Robotino est équipé d'une webcam (Figure 1.4), dont la hauteur et l'inclinaison sont ajustables. La webcam est connectée à l'unité centrale par connexion USB1. Il est possible d'effectuer du traitement d'image dans le but par exemple de suivre un objet ou de faire de l'asservissement visuel.

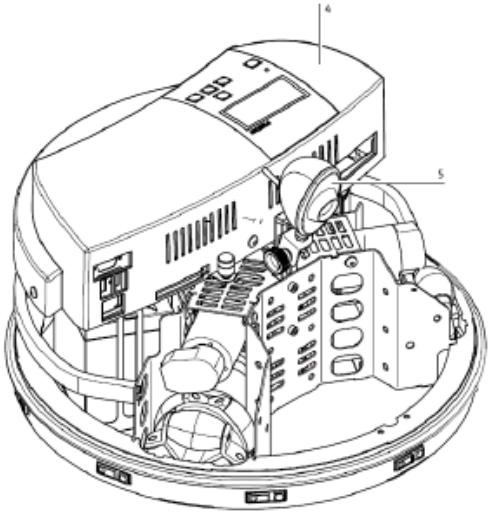


Figure 1.4 : Système de vision du robot

Contrôleur

Le contrôleur est constitué des composants suivants :

- un processeur 300Mhz, compatible avec Linux et 128MB de SDRAM ;
- une carte mémoire compact flash 256 MB ;
- une carte Ethernet ;
- un module wifi ;
- 2 ports USB1 ;
- un port VGA .

Ces différents éléments servent à la connexion d'un PC, un clavier et un écran.

Circuit d'entrée-sortie

Le circuit d'entrée-sortie établit la communication entre le contrôleur et les capteurs, les unités de déplacement et l'interface entrée-sortie.

Alimentation

La puissance est fournie par 2 batteries rechargeables de 12V débitant un courant maximal de 4Ah. Les deux batteries sont montées sur le châssis. Egalement disponible, deux batteries supplémentaires et un chargeur qui permet de charger deux, pendant que les deux autres sont utilisées.

Capteurs infrarouges

Le Robotino® est équipé de 9 capteurs infrarouges de mesure de distance qui sont montés sur la périphérie du châssis circulaire avec un angle de 40° l'un de l'autre. Le Robotino® peut donc détecter tous les objets aux alentours si leur hauteurs sont le permettent. Chacun des capteurs peut être interrogé individuellement afin d'éviter un obstacle ou de maintenir une distance par rapport à un objet. Les capteurs sont capables de mesurer des distances allant de 4 à 30 centimètres. Chaque capteur est connecté à l'alimentation et possède une sortie analogique.

Capteur anticollision

Le capteur anticollision (Figure 1.5) est situé sur la circonférence du châssis, la moindre pression sur le capteur crée un court-circuit entre ses deux surfaces et un signal est envoyé au contrôleur. Ainsi, un arrêt d'urgence des moteurs peut être programmé lors d'une collision.

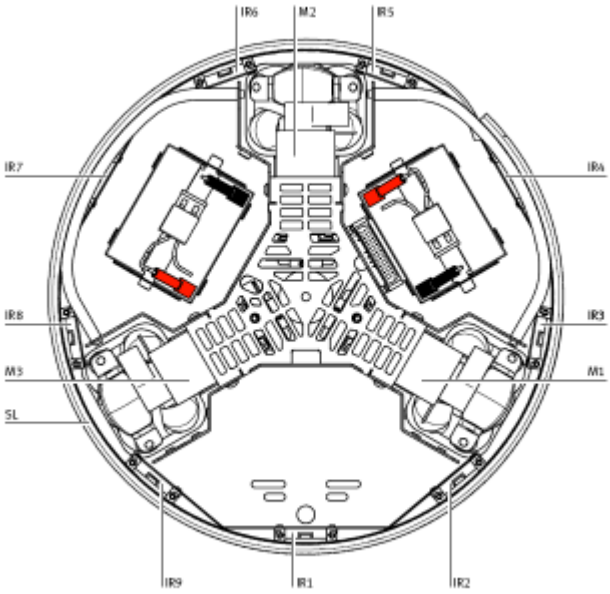


Figure 1.5 : Disposition des capteurs infrarouges (IR), des moteurs (M) et du capteur anticollision (SL)

Codeur incrémental

Un codeur incrémental est implanté sur chaque moteur, chaque codeur retourne une information sur la position angulaire de l'arbre. A partir de là, il est possible de déduire la vitesse réelle du moteur en tr/mn. Si cette vitesse dévie de la consigne, elle peut être régulée avec un régulateur PID dont les paramètres sont ajustables.

Interface entrées/sorties

L'interface E/S (Figure 1.6) permet de brancher des capteurs et/ou actionneurs et/ou voyants supplémentaires :

- 8 entrées analogiques (0 à 10 V) (AIN0 à AIN7)
- 8 entrées numériques (DI0 à DI7)
- 8 sorties numériques (DO0 à DO7)
- 2 relais pour les actionneurs supplémentaires (REL0 et REL1).

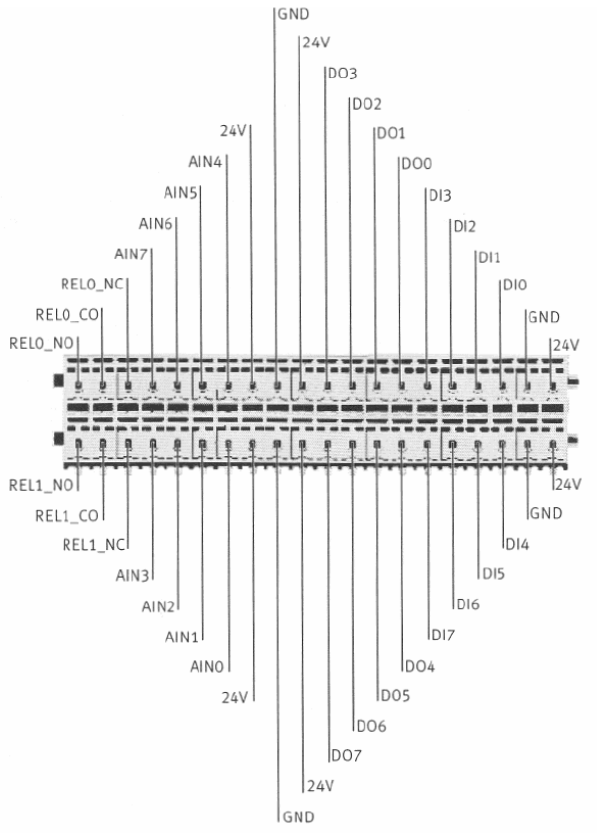


Figure 1.6 : Interface E-S du Robotino

Clavier et afficheur

Un clavier et un afficheur sont installés au dessus de l'unité centrale et permettent de sélectionner des options, d'afficher des informations ou bien de lancer des programmes en mémoire.

Mise en marche

Allumer Robotino®

Pour cela, pressez le bouton ON/OFF jusqu'à ce que le voyant vert s'allume. Pendant le démarrage, deux barres apparaissent sur l'afficheur, puis quand le robot est prêt (après environ 30 secondes) l'adresse IP du robot s'affiche (ex. 172.26.94.18) ainsi que l'état de charge des batteries.

Eteindre Robotino®

Une pression sur le bouton ON/OFF jusqu'à ce le voyant vert s'éteint permet d'éteindre le robot.

Menu principal

La navigation dans les différents menus s'effectue avec les touches fléchées et la sélection avec la touche ENTER. Les différents choix possibles sont les suivants :

- *Langages :* Il est possible de définir la langue entre l'anglais et l'allemand.
- *State of charge (Etat de la charge) :* Cette option informe de l'état réel de charge des accumulateurs : Tension et courant
- *Démos :* Lancer le programme de démonstration en appuyant sur la touche ENTER. Initialement, 6 programmes sont sauvegardés dans le robot (Circle, Forward, Quadrangle, Roaming, Follow line). Pour stopper le programme, appuyer brièvement sur n'importe qu'elle touche ou sur le détecteur d'anticollision.
- *Réseau :* Cette option affiche l'adresse IP du robot et permet de la changer. Bouger le curseur vers la droite en appuyant sur la touche ENTER. Augmenter la valeur du curseur ou bien la diminuer avec les flèches HAUT et BAS jusqu'à la valeur souhaiter.

Le logiciel RobotinoView®

Robotino® View est le logiciel de programmation graphique sous Windows du Robotino. Ce logiciel permet la création et l'exécution des programmes de commande pour Robotino® et offre également la possibilité de séquencer plusieurs programmes.

Familiarisation avec l'espace de travail

Le logiciel s'exécute sous un système d'exploitation Windows où l'écran principal est décris par la Figure 1.7.

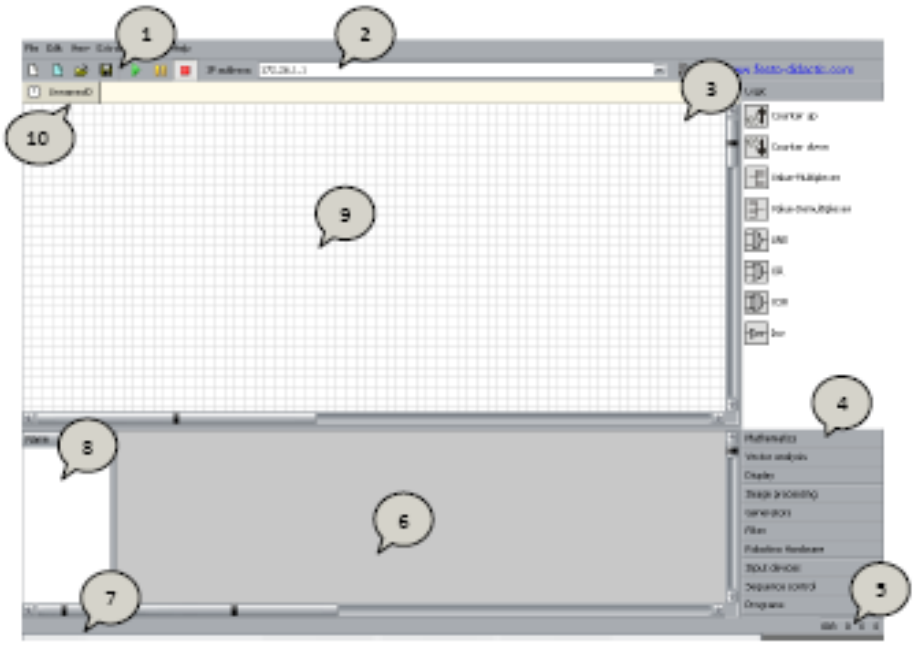


Figure 1.7. Ecran Principal de Robotinoview

(1) Barre d'outils : Elle permet de créer, d'ouvrir et de sauvegarder les programmes. Cette barre d'outils permet également de lancer, mettre en pause ou bien de stopper l'exécution d'un programme. Un double clic sur le bouton arrêt cause l'arrêt de tous les programmes actifs ;

(2) Champ de saisie de l'adresse IP du robot ;

(3) Bouton de connexion qui établi ou termine la connexion avec le robot ;

(4) Affichage de la bibliothèque avec tous les blocs de fonctions disponibles ;

(5) Affichage des performances de données pendant le contrôle à distance du robot par le programme ;

(6) Un double clic sur un bloc fonction permet l'affichage des paramètres interne dans cette zone ;

(7) Une barre d'état présentant diverses informations ;

(8) Zone affichant le nom et les valeurs de toutes les constantes du programme ;

(9) Espace de travail permettant le développement du programme ;

(10) Onglets permettant le passage d'un programme à un autre.

Présentation des différentes familles de bloc

Logic : Cette catégorie regroupe les fonctions logiques classiques tels que les compteurs, les multiplexeurs et les portes ET, OU, etc.

Mathematics : Contient les opérations mathématiques simples, tels que le produit, la somme, les comparaisons, etc.

Vector Analysis : Regroupe les outils permettant les conversions et opérations sur les vecteurs plans.

Display : Contient les outils d'affichages tels que l'oscilloscope, l'affichage des vecteurs dans le plan.

Image Processing : Regroupe les outils de base de traitement d'image.

Generators : Contient divers générateurs de signaux, impulsion, sinus, triangulaire, horloge et contant.

Filter : La fonction smoothing permet le lissage d'un signal.

Robotino Hardware : Contient les différent actionneurs et capteurs de Robotino®.

Input Devices : Périphérique de commande du Robot tels que le joystick et le panneau de contrôle.

Sequence Control : Divers éléments permettant de démarrer, de quitter ou de synchroniser plusieurs programmes.

Programs : Liste des programmes ouverts afin de les insérer dans une séquence

Exercice1 :

Après avoir découvert le principe de fonctionnement du logiciel **Robotinoview 2** et en se basant sur l'aide à partir de la touche F1, identifier les blocs permettant de lire:

1. les capteurs optiques en dessous du châssis. Quelles sont les entrées branchées sur ces capteurs ;
2. le capteur inductif en précisant le numéro de l'entrée branchée dessus ;
3. la caméra et tester son fonctionnement ;
4. le capteur de choc ;
5. Les codeurs optiques ;
6. Les capteurs de distance. Trouver la référence des trois capteurs frontaux.

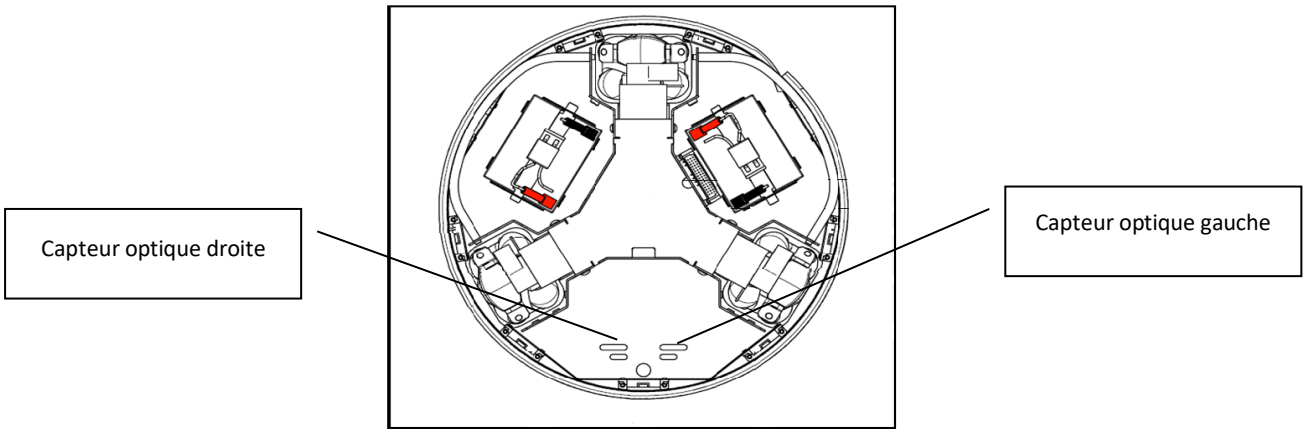


Figure 1.8 : Capteurs optiques du Robotino

Exercice 2 :

Ecrire un programme permettant au robot de suivre d'un trajet étiqueté par une couleur noir sur un sol clair grâce aux capteurs optiques (Figure 1.8), en utilisant le mode omni-drive de la Figure 1.9. Pour chaque mouvement du robot, mesurer sa trajectoire linéaire grâce aux fonctions de navigation. Enregistrer le temps de simulation, les vitesses désirées et mesurées de chaque moteur, grâce au bloc d'échange de données. Analyser les données par rapport à l'asservissement en vitesse. Tracer les courbes correspondantes. Le système composé du régulateur PID et le moteur–roue est-il stable, précis et rapide ? Commenter ?

Pour accéder à l'aide de chaque bloc de fonction, sélectionner ce dernier et appuyer sur F1.

Pour afficher les valeurs entrantes et sortantes à chaque bloc de fonction, appuyer sur CTRL+D.

Penser à utiliser les fonctions logiques et mathématiques pour gérer les quatre cas d'orientation possible sur un suivi de ligne.

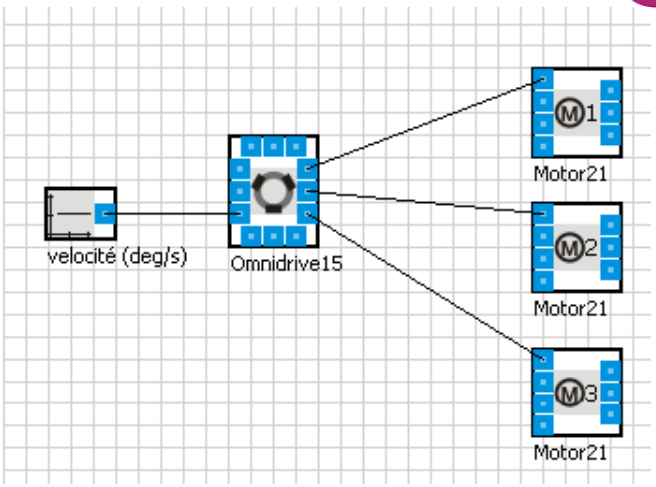


Figure 1.9 : Exemple d'un schéma de commande

Exercice 3 :

Ecrire un programme permettant de freiner le robot lors de la détection d'un obstacle frontale, détecté par les capteurs de distance, pendant son suivi d'une ligne noir au sol. Lorsqu'une collision apparaît à partir du parechoc (Bumper), le robot doit s'arrêter. Mémoriser le profil des vitesses des roues avant et après le freinage, puis effectuer une analyse.

Exercice 4 :

Ecrire un programme permettant au robot de s'arrêter pendant 3 secondes dès la détection d'une ligne métallique au sol. Il reprend son suivi d'une trajectoire contrasté en couleur après cette pause. Penser à utiliser le capteur inductif du robot (Figure 1.10).

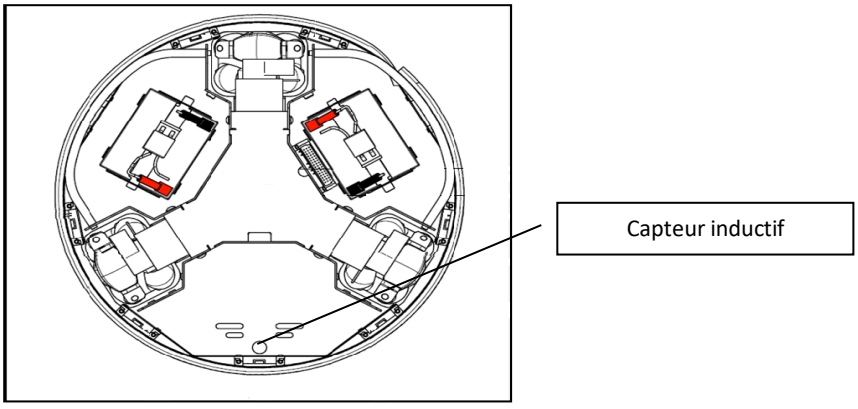


Figure 1.10 : Capteur inductif du Robotino

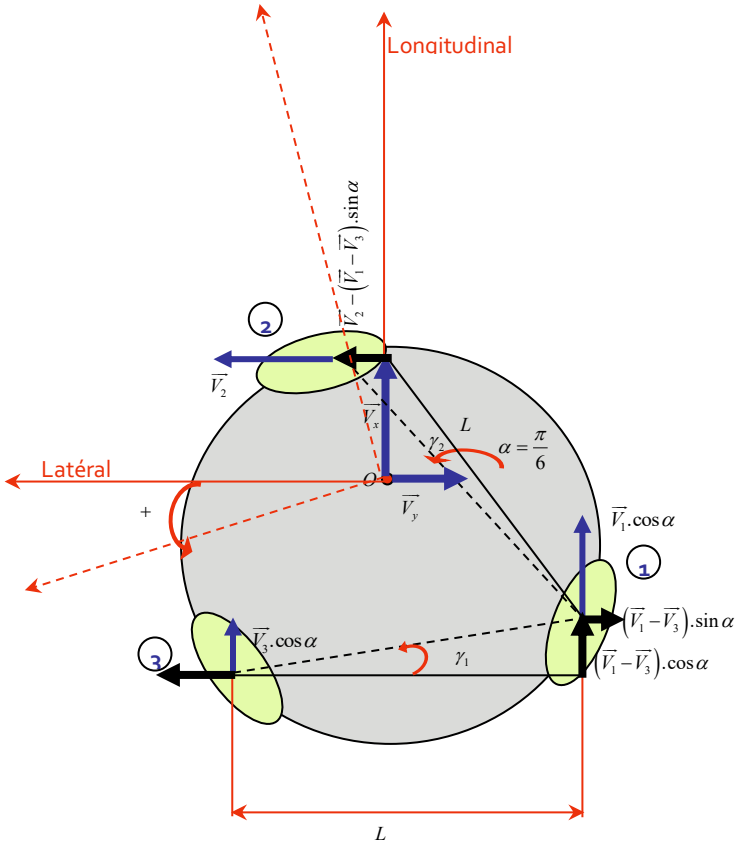
Exercice 5 :

Ecrire un programme permettant d'éviter un obstacle statique sur sa trajectoire tout en reprenant aussitôt sa trajectoire. Penser à utiliser la propriété omnidirectionnelle du robot. Enregistrer les profils des vitesses réalisées.

Annexes :

I. Modélisation Cinématique du Robotino

Calcul de $\overrightarrow{V_x}$ et $\overrightarrow{V_y}$ du centre O projetées sur les axes longitudinal, vertical et latéral:



Axe Longitudinal:

$$V_x = \frac{V_1+V_3}{3}.\cos\alpha = \frac{R}{3}.\left[(\dot{\theta}_1+\dot{\theta}_3).\cos\left(\frac{\pi}{6}\right)\right]$$

Axe Latéral:

$$V_y = \frac{1}{3}.\left(V_2+(V_3-V_1).\sin\alpha\right) = \frac{R}{3}.\left[(\dot{\theta}_3-\dot{\theta}_1).\sin\left(\frac{\pi}{6}\right)+\dot{\theta}_2\right]$$

Axe vertical:

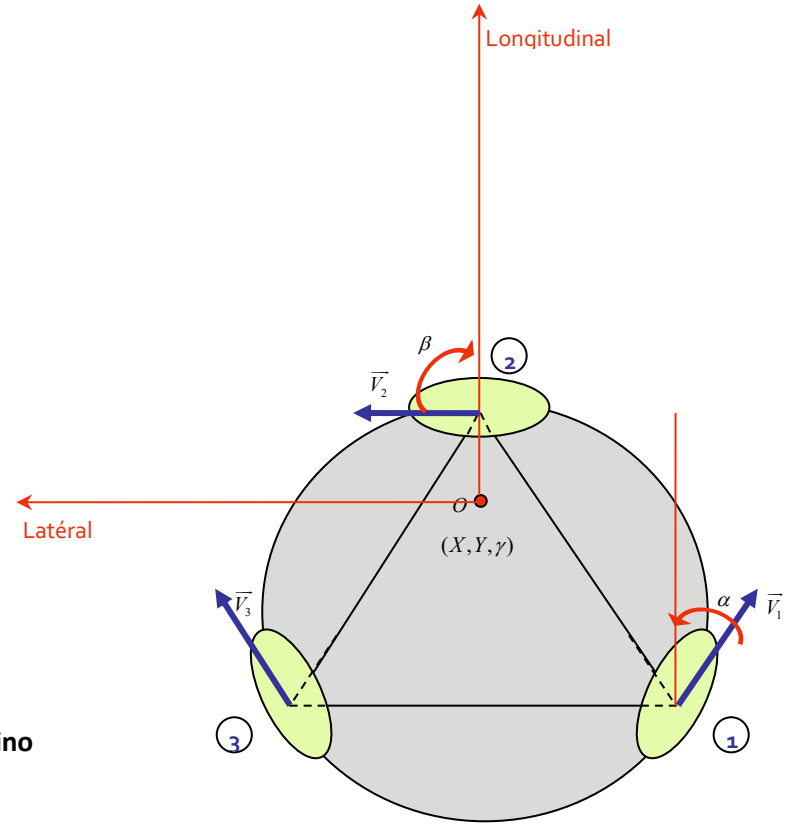
$$\dot{\gamma}=\dot{\gamma}_1+\dot{\gamma}_2 = \frac{R}{L}.\left[(\dot{\theta}_1-\dot{\theta}_3).\cos\left(\frac{\pi}{6}\right)\right]+\frac{R}{L}.\left[\dot{\theta}_2-(\dot{\theta}_1-\dot{\theta}_3).\sin\left(\frac{\pi}{6}\right)\right]$$

Modèle Cinématique Direct:

$$\begin{bmatrix} V_x \\ V_y \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} \frac{R}{3}\cos(\pi/6) & 0 & \frac{R}{3}\cos(\pi/6) \\ -\frac{R}{3}\sin(\pi/6) & \frac{R}{3} & \frac{R}{3}\sin(\pi/6) \\ \frac{R}{L}\cos(\pi/6)-\frac{R}{L}\sin(\pi/6) & \frac{R}{L} & -\frac{R}{L}\cos(\pi/6)+\frac{R}{L}\sin(\pi/6) \end{bmatrix} . \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

Modèle Cinématique Inverse:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{R} & -\frac{\sqrt{3}}{R} & \frac{\sqrt{3}.L}{3.R} \\ 0 & \frac{\sqrt{3}.\left(\sqrt{3}-1\right)}{R} & \frac{\sqrt{3}.L}{3.R} \\ \frac{\sqrt{3}}{R} & \frac{\sqrt{3}}{R} & -\frac{\sqrt{3}.L}{3.R} \end{bmatrix} . \begin{bmatrix} V_x \\ V_y \\ \dot{\gamma} \end{bmatrix}$$



II. Modélisation Géométrique du Robotino

(X,Y,γ) Coordonnées du CG.

$$X = \int_{t_1}^{t_2} V_x.dt = \frac{R}{3}.\int_{t_1}^{t_2}\left[(\dot{\theta}_1+\dot{\theta}_3).\cos\left(\frac{\pi}{6}\right)\right].dt$$

$$Y = \int_{t_1}^{t_2} V_y.dt = \frac{R}{3}.\int_{t_1}^{t_2}\left[\dot{\theta}_2-(\dot{\theta}_3-\dot{\theta}_1).\sin\left(\frac{\pi}{6}\right)\right].dt$$

$$\gamma = \int_{t_1}^{t_2} \dot{\gamma}.dt = \frac{R}{L}.\int_{t_1}^{t_2}\left[(\dot{\theta}_1-\dot{\theta}_3).\cos\left(\frac{\pi}{6}\right)\right].dt + \frac{R}{L}\int_{t_1}^{t_2}\left[\dot{\theta}_2-(\dot{\theta}_1-\dot{\theta}_3).\sin\left(\frac{\pi}{6}\right)\right].dt$$

TP N°2

Régulation de vitesse d'un moteur à courant continu

Objectifs du TP

- Identification des paramètres de la fonction de transfert du système en boucle ouverte (ou validation du modèle).
- Etude et comparaison des performances de deux régulateurs : Proportionnel (P) et Proportionnel et Intégral (PI).

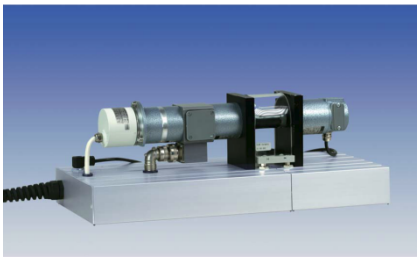
Présentation

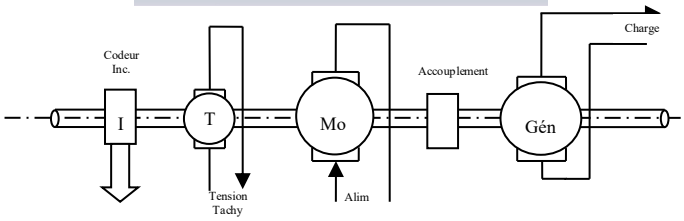
Le banc de TP se divise en deux parties :

La première (DR300) est constituée du moteur, la charge, le capteur de vitesse angulaire et le régulateur analogique (P et PI).

La deuxième est composée d'un PC muni d'une carte d'acquisition industrielle et du logiciel Matlab®. Celui-ci équipé des boîtes à outils nécessaires, jouera les rôles de générateur de signaux (GBF) et d'oscilloscope).

Le DR300





Ce dispositif est composé de deux moteurs à courant continu identiques. Le premier (Mot.) à excitation permanente est utilisé comme actionneur. Le deuxième (Géné.) fonctionne en génératrice et joue le rôle d'une charge variable.

Le dispositif sera utilisé dans sa configuration la plus simple.

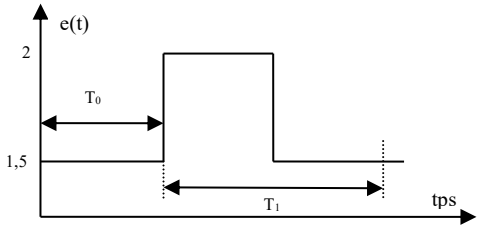
Manipulation

Mettre le commutateur sur la position Time1.

Étude en boucle ouverte.

Il est vivement conseillé de revoir les méthodes graphiques pour la détermination du gain et de la constante de temps d'un système de premier ordre.

On désire identifier la fonction de transfert du système en boucle ouverte. Pour cela, on s'intéresse à la réponse indicielle du système. Cependant, pour éviter la zone morte (problème de seuil) et pour vérifier la symétrie du moteur par rapport aux échelons ascendant et descendant, on applique un signal de la forme suivante :



Ce signal est obtenu à partir 

N.B. Faites vérifier impérativement votre câblage avant de lancer le système !!!

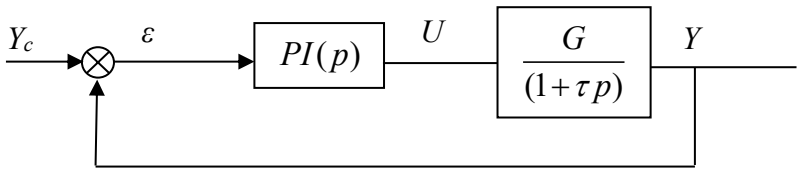
Le temps T_0 permet d'avoir comme point de départ un régime permanent !

- Choisir les valeurs de T_0 et T_1 qui permettent à la sortie du système d'atteindre le régime permanent. Veiller à ce que ces durées ne soient pas trop longues.
- Injecter le signal, fabriqué à l'aide de "signal builder" dans l'entrée I-moteur.
- Observer d'abord les signaux à l'oscilloscope (simulink) puis les récupérer dans "worksapace" à l'aide de boîtes "simout".
- Sous Matlab, écrire un script qui permet de tracer les deux courbes sur le même graphe.
- Déterminer alors le gain et la constante de temps du système en boucle ouverte.
- Comparer avec les valeurs théoriques.

Effet de la charge

- Connecter la sortie AO1 (boîtier)à l'entrée I-Generator (**7**) (zone external).
- Envoyer un signal constant d'amplitude 0.2V. Commenter les résultats observés.

Étude en boucle fermée



La fonction de transfert du régulateur analogique (PI) intégré à la maquette est de la forme :

$PI(p)=\frac{k_{PI}\left(1+T_{PI}p\right)}{p}$. Mettre cette expression sous la forme standard $k_p\left(1+\frac{1}{T_i p}\right)$

En déduire les valeurs de k_p et T_i en fonction de k_{PI} et T_{PI}

Action proportionnelle (*P*)

- Mettre le commutateur (12) sur "*off*" pour désactiver l'action intégrale.
- Mettre les deux potentiomètres sur "1" et déterminer la valeur de l'erreur finale d'ordre "0" $\left(\varepsilon(\infty)\right)$.
- Augmenter progressivement les valeurs des deux potentiomètres, qu'observe-t-on ?
- Peut-on annuler l'erreur $\left(\varepsilon(\infty)\right)$?

Effet de la charge

- Connecter la sortie AO1 (boîtier)à l'entrée I-Generator (**9**) (zone external).
- Envoyer un signal constant d'amplitude 0.2V. Commenter les résultats observés.

Action proportionnelle et Intégrale (PI)

On désire maintenant annuler l'erreur statique et améliorer les performances dynamiques. On peut utiliser pour cela une correction proportionnelle et intégrale (PI).

- Montrer que ce correcteur permet bien d'annuler l'erreur statique.

Nous voulons que l'ensemble (moteur-charge) se comporte comme un système de deuxième ordre de la forme $\frac{N(p)}{p^2+2\xi\omega_n p+\omega_n^2}$ avec les caractéristiques suivantes :

Remarque : le temps de réponse à 5% le plus court est obtenu pour un dépassement relatif de 5%. Ce qui correspond à un coefficient d'amortissement $\xi = \frac{\sqrt{2}}{2}$. C'est un compromis "rapidité-amortissement" généralement admis.

Pour ω_n, prendre la valeur qui correspond à un temps de réponse (à 5%) $T_r = \frac{3}{\xi \omega_n} = 0{,}1s$

Donner l'expression de la fonction de transfert du système étudié en boucle fermée, la mettre sous la forme $\frac{Y(p)}{Y_c(p)} = \frac{N(p)}{p^2+\alpha p+\beta}$ α et β étant des paramètres qui dépendent des paramètres du correcteur PI.

En déduire les expressions des paramètres du régulateur.
Transcrire le calcul des paramètres du régulateur sous forme d'un script Matlab (progPI.m). Cette partie doit être préparée à l'avance !

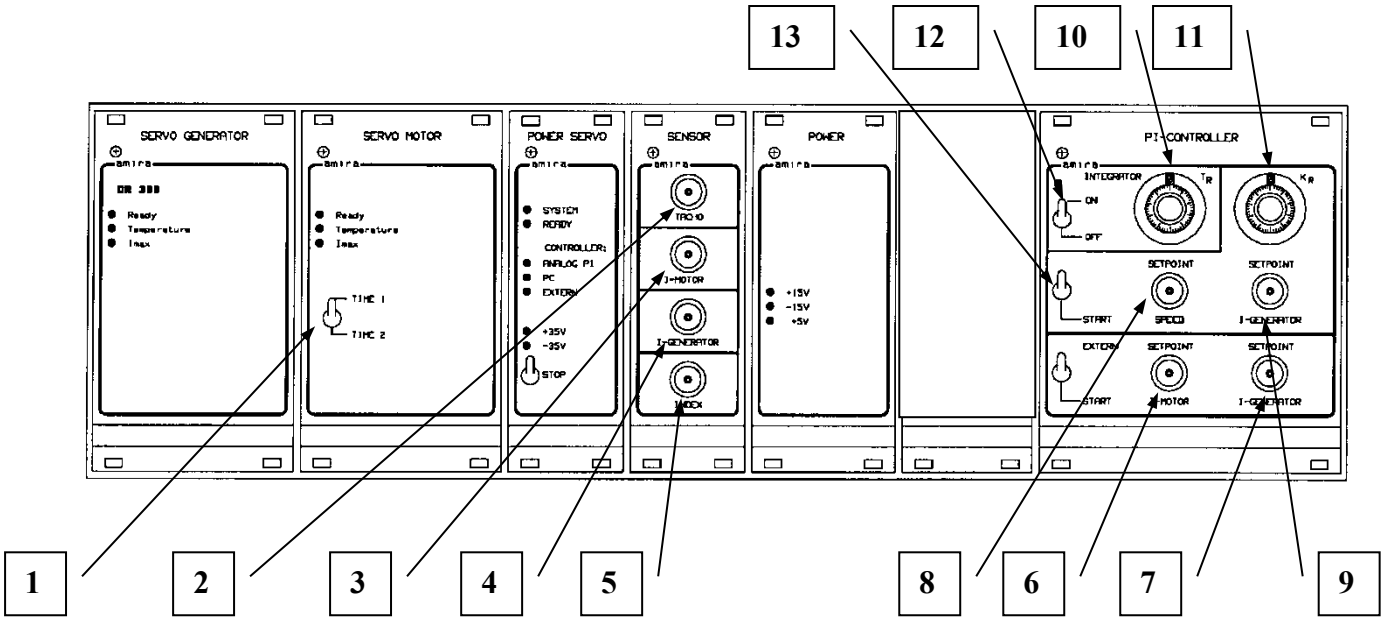
- Après l'avoir testé sous Simulink, vérifier sa validité sur le système.

Effet de la charge

Refaire la même manipulation que dans le cas précédent

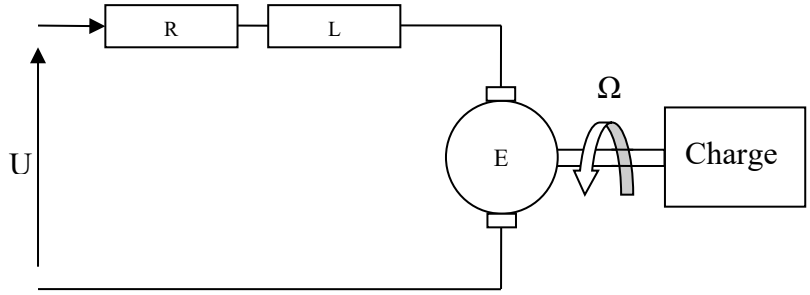
- Annexes

Annexe 1 **Face avant du module d'interface et de puissance.**



1	Choix entre les 2 boucles de régulation du courant absorbé.
2	Tension délivrée par la génératrice tachymétrique. Calibrage : $2{,}5\,\mathrm{mV/tr\cdot mn^{-1}}$
3	Courant absorbée par le moteur M1 Calibrage : $0{,}4\ \mathrm{A/V}$;
4	Courant absorbée par la génératrice M2 Calibrage : $0{,}2\ \mathrm{A/V}$;
5	Signal d'index fourni par le codeur incrémental. Calibrage : $1\,\mathrm{TTL\ pulse/tr}$
6	Entrée de consigne en courant pour essai en boucle ouverte ou dans le cas de l'utilisation du correcteur de vitesse externe. Calibrage : $0{,}4\ \mathrm{A/V}$; $\pm 10\ \mathrm{V}$
7	Entrée de consigne pour le courant débité par la génératrice M2 dans le cas de l'utilisation d'un correcteur de vitesse externe. Calibrage : $0{,}138\ \mathrm{Ncm/V}$; $\pm 10\ \mathrm{V}$
8	Entrée de consigne en vitesse dans le cas de l'utilisation du correcteur PI fourni avec le module .Calibrage : $2{,}5\,\mathrm{mV/tr\cdot mn^{-1}}$; $\pm 10\ \mathrm{V}$
9	Entrée de consigne en courant pour la génératrice dans le cas de l'utilisation du correcteur PI fourni avec le module .Calibrage : $0{,}138\ \mathrm{Ncm/V}$; $\pm 10\mathrm{V}$
10	Réglage de la constante de temps T_R du correcteur PI analogique. Calibrage : $0\ldots10 \leftrightarrow 0\ldots1\,\mathrm{s}$;
11	Réglage du gain K_R du correcteur PI analogique.Calibrage : $0\ldots100$;
12	Commutateur qui active ou désactive la partie intégrale de la correction. Lorsque qu'elle est désactivée, la fonction de transfert du correcteur devient $C(p) = K_R \cdot T_R$.
13	Commutateur qui connecte la sortie du régulateur de vitesse à l'entrée du régulateur de courant, ainsi que la consigne courant de M2 au régulateur courant de ce moteur.

Annexe 2 : **Mise en équation**



Équation électrique : Loi d'Ohm

$$u(t) = R\,i(t) + l\frac{di(t)}{dt} + e(t) \qquad U(p) = RI(p) + LpI(p) + E(p)$$

Excitation permanente

$$E(p) = C\Phi\Omega(p)$$

Équation mécanique : Loi fondamentale de la dynamique

$$\gamma(t) = \sum \text{couples} = \gamma_{e\_mot} - \gamma_{res} = J\frac{d\omega}{dt} \qquad \Gamma(p) = \Gamma_{e\_mot} - \Gamma_{res} = Jp\Omega(p)$$

$$\Gamma_{e\_mot} = k\,I(p), \qquad \Gamma_{res} = \text{charge} + \text{frottements}$$

$$\Omega(p) = \frac{1}{C\Phi}\frac{1}{1 + T_m s + T_m T_e s^2}U(s) - \frac{R}{kC\Phi}\frac{1 + T_A s}{1 + T_m s + T_m T_e s^2}\Gamma_{res}(s)$$

$$T_e = \frac{L}{R} \qquad T_m = \frac{JR}{kC\Phi}$$

TP N°3

Modélisation et Commande d'un Robot mobile

L'objectif de ce TP est de comprendre le rôle des différentes actions d'un régulateur PID à travers la commande en position et en vitesse d'un robot mobile de type **Khepera II** (Figure 3.1). Ce dernier, conçu par la société suisse **K-Team** (www.k-team.com), est un robot mobile à deux roues motrices, pilotées par des moteurs à courant continu, avec une extension modulaire qui peut porter une caméra et un circuit de communication Bluetooth.

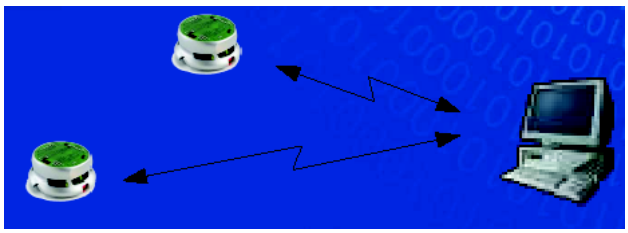


Figure 3.1: Robots Khepera

Pendant la partie expérimentale, vous allez communiquer avec le robot via la liaison série (RS232), en utilisant l'hyperterminal dans un premier temps ensuite Matlab pour la partie commande. La séance de TP comporte donc une partie de simulation et une autre d'expérimentation. Avant d'attaquer la commande du robot, chaque étudiant devra comprendre d'abord le fonctionnement de ce dernier, après un passage par la modélisation physique et par la simulation sous Matlab/Simulink.

Étude théorique à préparer :

Dans le schéma de principe de la Figure 3.2, le centre de gravité du robot est représenté par le point $G(x_G, y_G, \alpha)$, où (x_G, y_G) représente ces coordonnées géométriques dans le repère absolue (monde) et α l'angle d'orientation du robot autour de l'axe vertical **Z** (appelé angle du lacet). Les forces de contacts (de traction), définies sur l'axe longitudinal sont respectivement : F_{x_1} et F_{x_2}. Les coordonnées cinématiques du centre de gravité par rapport au repère relatif au robot sont $(\dot{u}, \dot{y}, \dot{\alpha})$, respectivement vitesses transversale, latérale et de lacet. Dans ce cas, les roues motrices ne sont pas motorisées en direction, seul une différence de vitesse angulaire entre les deux roues, créée un mouvement de lacet autour de G et définie une orientation sur le plan (X,Y).

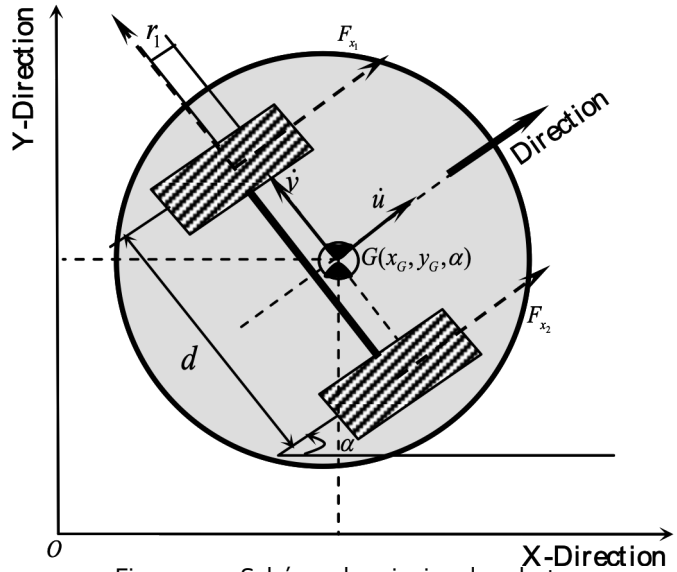


Figure 3.2 : Schéma de principe du robot mobile Khepera

I. Modélisation cinématique :

Dans la plupart des modèles cinématiques de robots mobiles, le glissement au niveau du contact n'est pas pris en considération. Dans ce cas les roues sont considérées rigides de rayon R et le contact devient presque ponctuel. En prenant $\dot{\theta}_1$ et $\dot{\theta}_2$ comme vitesses angulaires de chaque roue motrices (droite et gauche), les expressions des vitesses longitudinale, latérale et de lacet du centre de gravité dans le repère relatif au robot sont représentés comme suit :

$$\begin{cases} \dot{u} = \dfrac{\dot{\theta}_1 + \dot{\theta}_2}{2}.R; \\ \dot{\alpha} = \dfrac{\dot{\theta}_1 - \dot{\theta}_2}{d}.R; \\ \dot{v} = \dfrac{\dot{\theta}_1 - \dot{\theta}_2}{d}.R.r_1 \end{cases} \quad (1)$$

où d est la distance entre les deux roues motrices et r_1 la distance entre l'axe des deux roues et le centre de gravité.

Questions :

1. Expliquer l'origine des formulations mathématiques du système (1) ;
2. Donnez une représentation matricielle du système (1), liant les vitesses longitudinale, latérale et de lacet en fonction des deux vitesses angulaires des roues ;
3. Déduisez le modèle cinématique du robot $(\dot{x}_G, \dot{y}_G, \dot{\alpha})$, défini dans le repère absolu $(O, \vec{X}, \vec{Y})$.

II. Modélisation géométrique :

A partir de l'étude cinématique, déduisez les coordonnées géométriques (x_G, y_G, α) entre deux instants du robot dans son repère absolu.

III. Modélisation dynamique :

Dans cette partie, nous supposons que la dynamique latérale sur le robot est négligée, suite à l'action faible des forces d'impacts latérales sur le robot (roue sous forme de disc plein). Ainsi, l'étude de la dynamique du robot se fera sur deux axes seulement (longitudinal et lacet).

L'ensemble des équations dynamiques sont données par les systèmes d'équations différentielles suivant :

$$\begin{cases} F_{x_1} + F_{x2} = m.\ddot{x} \\ \left(F_{x_1} - F_{x2}\right).\dfrac{d}{2} = I_Z.\ddot{\alpha} \\ U_1 = J_1.\ddot{\theta}_1 + f_1.\dot{\theta}_1 + F_{x_1}.R \\ U_2 = J_2.\ddot{\theta}_2 + f_2.\dot{\theta}_2 + F_{x2}.R \end{cases} \quad (2)$$

avec :

F_{x_1} et F_{x_2} sont les forces de contact sur l'axe longitudinal ;

I_Z est le moment d'inertie du robot de masse **m** autours de l'axe **Z ;**

U_1 et U_2 les couples d'entrées ;

J_1 , J_2 et f_1 , f_2 sont les inerties et frottements visqueux des moteurs de tractions 1 et 2;

R , *d* correspondent respectivement au rayon de chacune des roues et la distance entre ces deux dernières ;

$\ddot{x}, \ddot{\alpha}, \ddot{\theta}_1, \ddot{\theta}_2, \dot{\theta}_1, \dot{\theta}_2$ sont les accélérations et vitesses linéaires et articulaires du robot.

1. A partir des équations du système (2), distinguer les dynamiques suivantes :
   - Longitudinale ;
   - Lacet ;
   - Systèmes électromécaniques.
2. A partir du système (2), déduire les différentes fonctions de transfert pour caractériser le modèle du robot, sachant que :

$U^T = \left(F_{x_1} \;\; F_{x_2} \;\; U_1 \;\; U_2\right)^T$ est un vecteur d'entrée, $Y^T = \left(\dot{\theta}_1 \;\; \dot{\theta}_2\right)^T$ est le vecteur de sortie.

3. Le système du robot décrit par le modèle de l'équation (2) est un système Multi entrées - Multi sorties (MIMO), Multi entrées -simple sortie (MISO), Simple entrée Simple sortie (SISO), ou bien simple entrée -Multi sorties (SIMO) ? Commentez votre choix.

Pendant la séance de TP :

I. Simulation :

1. Initialisation :

Dans cette partie, vous allez simuler le comportement dynamique du robot mobile en utilisant Matlab/Simulink.
Pour cela, copiez le fichier *'robot.mdl'* localisé dans le répertoire 'C:\Etudiants' dans votre répertoire. Ouvrez ce dernier et cliquez sur le bloc '*initialisation*', afin de charger les paramètres du modèle dynamique du robot. Ainsi, les valeurs sont :
m= 0.08kg;
R= 0.005m;
r_1= 0.01m
f_1= f_2=0.0003N.m.s/rad;
J_1= J_2=0.0001kg.m^2;
d= 0.04m;
I_Z= 0.00588kg.m^2;

- Que constater vous sur l'environnement 'workspace' des variables?

2. Régulation en vitesse à partir de Simulink (1h30mn) :

Après le lancement de Simulink, ouvrez un fichier khepera.mdl et enregistrez-le sous un autre nom dans même répertoire que le fichier d'initialisation.

- Le modèle dynamique du robot existe dans le bloc déjà défini dans le fichier ;
- Dans la librairie 'Sources', glissez deux 'Signal Builder' afin de définir le profil des vitesses d'entrée et deux blocs 'Step' pour les forces de contact (on suppose dans un premier temps que les forces de frottement secs sont présents en un seul sens). Par défaut, ces entrées représentent des **échelons unitaires**. A partir de la librairie 'Signal & Routing', ramener un boc multiplexeur 'MUX' pour les quatre entrées et un bloc démultiplexeur 'DEMUX' pour les deux sorties 'vitesses' ;
- Dans la librairie 'Additional Linear' dans 'Simulink Extra', ramener deux bocs PID pour commander les deux moteurs du robot ;

- Chercher deux blocs additionneurs/soustracteurs et glisser les dans votre environnement ;
- Reliez les différentes parties d'entrée et de sortie de votre modèle ;
- Avec le temps, le schéma de simulation s'élargit, dans ce cas, sélectionner avec la souris à la fois le 'MUX', bloc d'état et 'DEMUX' puis faite un CTRL+G. Un sous-système va se créer. Donner un nom à ce dernier (ex . Modèle Dynamique) et aux différentes entrées et sorties. Avec le même principe, réaliser un sous-système, composé des blocs PID et des blocs 'Step', et appelez le 'Bloc Commande'.
- Vous pouvez changer la couleur des blocs sous-systèmes en cliquant à droite sur ce dernier, et en sélectionnant la couleur du contour et du fond ;
- Ramenez deux blocs 'Scope' de la librairie 'Sinks', afin d'afficher les sorties vitesses ;
- Connectez l'ensemble des blocs puis lancer la simulation sur une période de **0.5 sec**, en gardant les valeurs des paramètres par défaut. Visualiser les sorties ?
- Visualisez l'entrée et la sortie correspondante sur un même graphique en utilisant un 'MUX' à deux entrées et un 'Scope' ;
- Visualisez les deux commandes de vitesses U_1 et U_2 ainsi que les erreurs de suivi en vitesses?
- **Enregistrer les données simulées dans l'espace de travail de la vitesse désirée et la vitesse estimée en utilisant le bloc 'To workspace' dans 'Sinks' et le bloc 'Clock'. Pensez à paramétrer le bloc 'To workspace' en mettant 'array' qui veut dire tableau, comme option de sauvegarde.**
- **Visualiser vos courbes de vitesses en les superposant à partir de l'invite de commande en tapant, 'plot', 'grid', 'xlabel', 'ylabel', 'title'. Pensez à utiliser l'aide, ou bien le 'Help' de ces fonctions de commandes pour vérifier la syntaxe leurs syntaxes.**
- **D'après les courbes obtenues sur les vitesses superposées d'entrée et de sortie suite à un échelon unitaire, quel est l'ordre du système étudié ? pourquoi ?**
- ***Donnez un tableau comparatif décrivant les performances des différentes actions P, PI, PD et PID en termes de rapidité, stabilité et précision du système électromécanique contrôlé en boucle fermée.***

3. Simulation des modèles cinématique et géométrique (30 mn) :

- Avec le même principe de simulation décris dans la partie 'Modèle Dynamique', réalisez deux sous-systèmes 'Modèle Cinématique' et 'Modèle Géométrique' du robot Khepera ? Pour cela, vous aurez besoin en plus des blocs Simulink utilisés dans le modèle dynamique les blocs suivants : 'Substract', 'Product', 'Add' et 'Gain' de la librairie 'Math Operations', ainsi que le bloc 'Integrator' dans la librairie 'Continuous'. Pensez à utiliser le système d'équation (1) de la préparation ;
- Visualiser la trajectoire du robot selon X et Y, grâce au bloc 'XY Graph' de la librairie 'Sinks' ;
- ***Trouvez une combinaison sur les consignes vitesses, pour que le robot parcoure une ligne droite sur 60sec et une trajectoire en cercle sur 40sec.***

Commande du robot Khepera IV

Le robot Khepera IV est un robot muni de 2 roues fixes, de capteurs infrarouge, ultrason, d'une caméra, d'un micro. Sa description est donnée ci-dessous.

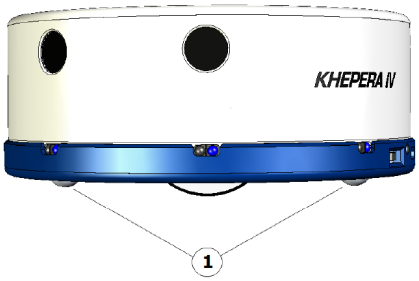


Figure 3.3 : Vue de gauche

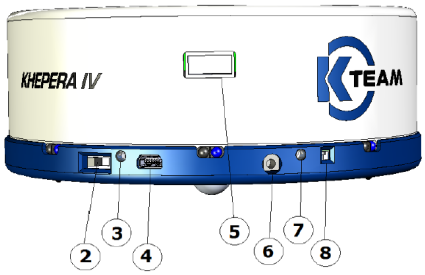


Figure 3.4 : Vue arrière

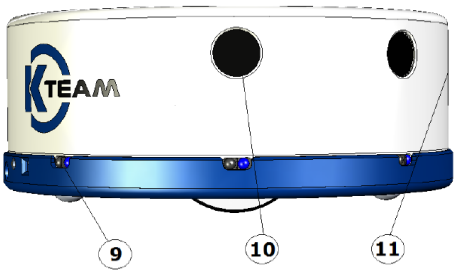


Figure 3.4: Vue de droite

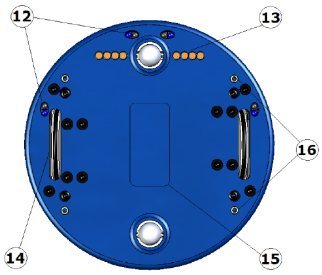


Figure 3.5: Vue du bas

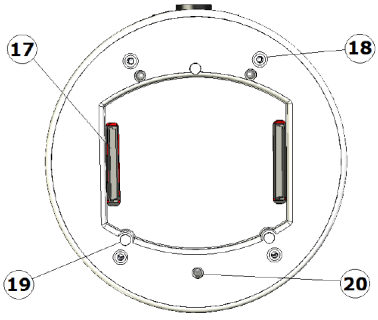


Figure 3.6 : Vue du haut

- 1 : roues
- 2 : bouton d'arrêt de démarrage On/Off
- 3 : LED d'état du robot
- 4 : connecteur mini-USB
- 5 : connecteur USB A
- 6 : port de charge de la batterie (9v, 1.5A)
- 7 : LED d'état de la charge
- 8 : Reset
- 9 : capteur infrarouges (8x) anti collision
- 10 : capteurs ultrason (5x), anti collision
- 11 : camera
- 12 : capteurs infrarouge (4x) antichute
- 13 : contacts pour la base
- 14 : roues

17 : port de connexion

18 : Nœuds M3

- 15 : étiquette

19 : aimants (3x)

20 : LED RGB (3x)

La disposition des capteurs infrarouges et ultrason est donnée par les figures suivantes.

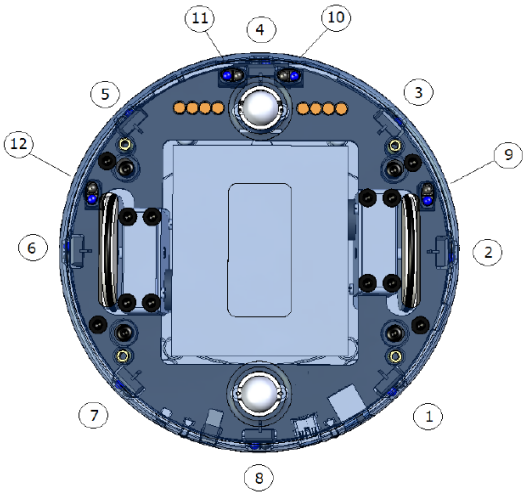


Figure 3.7 : disposition des capteurs IR

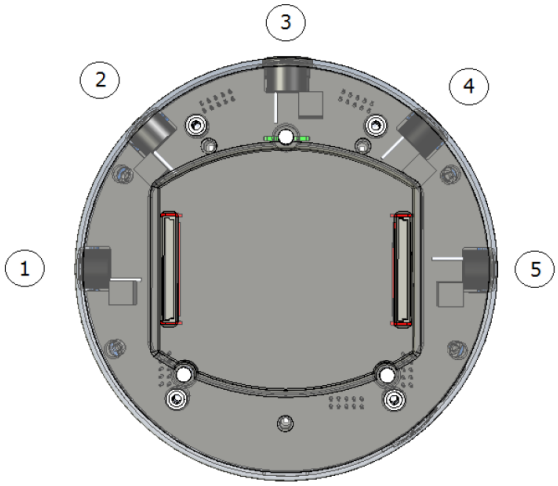


Figure 3.8 : disposition des capteurs ultrason

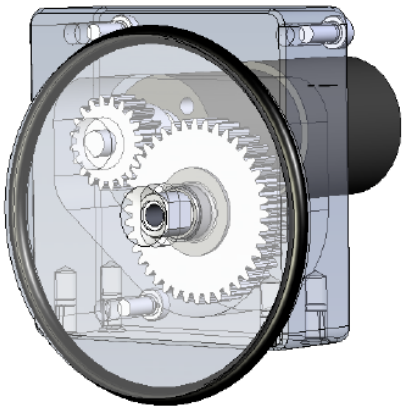


Figure 3.9 : roue et bloc moteur

La figure 3.9 représente un moteur, son réducteur et la roue associée.

Le robot Khepera IV dispose d'un système linux embarqué tournant sur un processeur ARM. Ce robot peut –être commandé en Wifi, Bluetooth ou via le port USB et la communication avec le robot se fait via un contrôle à distance du terminal linux embarqué.

Dans ce TP nous commanderons le robot via le port USB. En interne, la commande **Kh4Start** crée la liaison série (passant par l'USB) sur le port donné en paramètre avec une vitesse de transmission de 115200 baud.

Elle attend ensuite le démarrage du système linux embarqué, se logue sur le terminal et démarre le programme d'interprétation des commandes. Elle en profite pour lancer une commande de désactivation des capteurs ultrasons, ce qui permet d'avoir un retour sur le bon fonctionnement via l'arrêt des cliquetis sonores.

Le protocole de communication sur le port série fonctionne comme suit :

- Nous envoyons une lettre majuscule correspondant à une fonction puis les paramètres nécessaires.

  On envoie par exemple un message du type : F,motor_left,motor_right dans le cas d'une consigne de position

- Le robot répond (si tout fonctionne normalement) par la lettre minuscule correspondante et, s'il y a lieu, par des valeurs (dans le cas de la lecture des encodeurs par exemple).

  On obtient par exemple un message du type : f pour réponse à une consigne de position

La commande du robot peut se faire à l'aide d'une chaîne de compilation proposée par le constructeur. Différents couples message/réponse suivants ce protocole ont été encapsulés dans des fonctions Matlab pour vous offrir une utilisation simplifiée lors de ces séances de TP par l'utilisation des commandes présentées ci-dessous:

- kh4ConfigurePID(P,I,D) : charge les coefficients du régulateur PID
- kh4ResetEncoders : remise à zéro des encodeurs de position des moteurs
- kh4SetPosition(posg,posd) : envoi d'une consigne de position au moteur gauche (posg) et droit (posd)
- kh4ReadEncoders : lecture des positions d'encodeurs courantes
- kh4SetSpeed(vitg,vitd) : envoi d'une consigne de vitesse au moteur gauche (vitg) et droit (vitd)
- kh4ReadSpeed : lecture des vitesses courantes.
- kh4End : sert à la bonne fermeture/destruction en mémoire des objets port_série.

Vous êtes maintenant prêts à contrôler le robot KheperaIV.

Dans la suite, chaque binôme devra commander le robot Khepera à partir de Matlab. Vous devez créer un script pour chaque partie.

2. Régulation en position (45 mn) :

Commandez en position le robot en suivant les étapes suivantes :

- Fixez les coefficients de votre contrôleur PID comme suit : **P=10, I= 5, D=10** et charger les sur le robot l'aide de la commande : **kh4ConfigurePID(P,I,D)**

- Déplacer le robot à une position désirée de 16cm sur une ligne droite en utilisant la commande **success=kh4SetPosition(gauche, droite),** avec gauche et droite sont les positions des roues gauche et droite du robot.

- Vérifier le déplacement exacte du robot en utilisant une règle et comparer le avec la valeur actuelle des codeurs de position l'aide de la commande **kh4ReadEncodersPosition**. Penser à précéder cette commande par une pause. N'oublier pas d'initialiser le compteur des codeurs à zéro au cours de chaque initialisation grâce à l'instruction **kh4ResetEncoders**.

- Pourquoi le robot s'arrête une fois la consigne de position est atteinte ?

- Tester différents valeurs des régulateurs P, PI, PD et PID pour la commande en position du robot. Commentez vos résultats ?

3. Régulation en vitesse (35 mn) :

- Commander en vitesse le robot en suivant les étapes suivantes :

  - Fixez les coefficients de votre contrôleur PID comme suit : **P=10, I= 5, D=10** et charger ces valeurs en exécutant la commande suivant : **kh4ConfigurePID(P,I,D)**

  - Déplacer le robot à une vitesse désirée de valeur 1000 sur chacune des roues en utilisant la commande **kh4SetSpeed(1000,1000)** pendant une durée constante en utilisant une boucle 'while' (faire help while dans l'invite de commande de Matlab). Que constatez-vous ?

  - Afficher les vitesses des roues après les avoir récupérés à l'aide de l'instruction **kh4ReadSpeed**.

  - Penser à remettre les vitesses à zéro après initialisation avec la commande **kh4ResetEncoders**.

  - Comparer les performances du robot régulé en vitesse dans le cas d'un régulateur P, PI, PD et PID. Commentez vos résultats ?

  - Commander le robot en vitesse pour qu'il puisse réaliser une trajectoire linéaire pendant un temps approprié, suivi après d'une trajectoire circulaire, comme dans le cas de la simulation.

4. Conclusion (10 mn) :

Par rapport aux deux dernières expérimentations sur les régulations en position et en vitesse, que pourriez-vous conclure entre les deux types de régulation ?

TP N°4
Régulation Numérique : Simulation d'un système Electromécanique
Préparation

Dans ce travail pratique, nous étudions la commande PID numérique d'un système électromécanique (Figure 4.1) du robot mobile omnidirectionnel Robotino (Figure 4. 2).

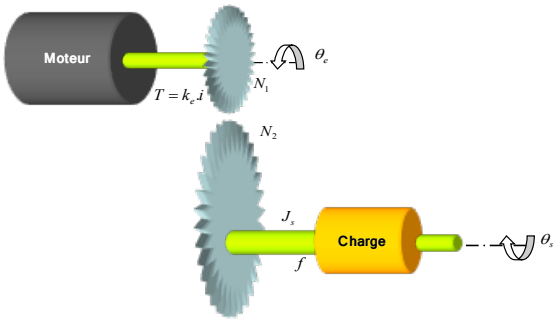


Figure 4.1 Système électromécanique

Figure 4.2. Robot Mobile 'Robotino'

Le système étudié est composé de cinq partie distincts, un circuit électrique de résistance R et d'inductance L, une partie mécanique d'inertie Je et de frottement visqueux f_e, un axe de rigidité K, un engrenage de constante N et enfin une roue d'inertie J_s et de frottement visqueux de contact f_s comme le montre la Figure 4.3. Par analogie avec un schéma bloc, nous retrouvons les cinq parties dans la Figure 4.4.

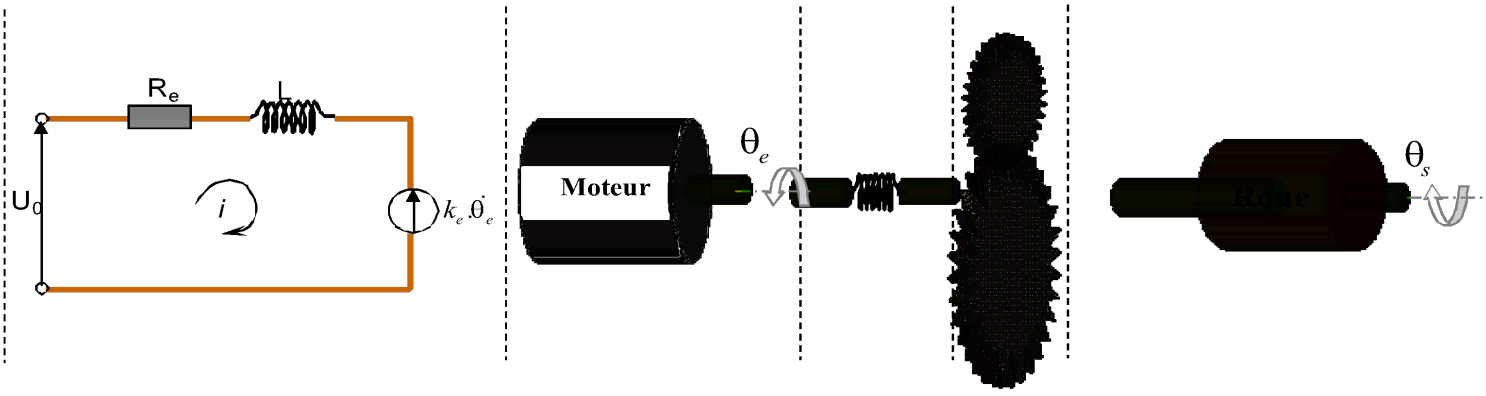


Figure 4.3 Décomposition du système électromécanique

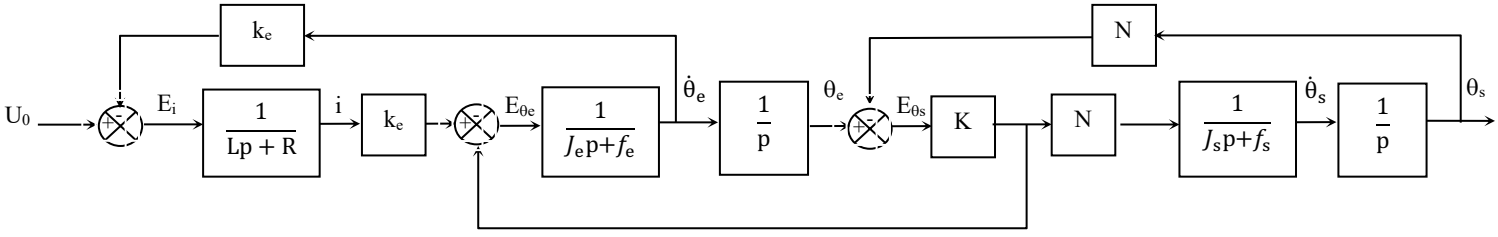


Figure 4.4 Schéma en bloc du système électromécanique en BO

Valeurs numériques :

R=8.13 Ω, Résistance interne du circuit électrique ;
L=8.9 mH, inductance interne du circuit électrique ;
k_e=43.15 mV.sec/rad, constante électrique ;
J_e=7.95x 10^{-6} kg.m^2, inertie interne du moteur ;
f_e=47 µN.m.sec/rad, frottement visqueux du moteur ;
J_s=0.02 N.m.sec/rad, inertie de la charge ;
f_s=504x 10^{-6} kg.m^2, frottement visqueux résistif au mouvement de la charge.
N=18, constante de réduction.
Les couples $(\theta_e, \dot{\theta}_e)$ et$(\theta_s, \dot{\theta}_s)$ représentent respectivement les positions et vitesses de l'arbre du moteur et de la roue

Hypothèses :

- Les non linéarités liées au jeu mécanique et au forces de contact avec la charge ne sont pas considérées dans le système étudié ;
- Les incertitudes paramétriques et de modélisation ne sont pas prises en compte.

Préparation du TP :

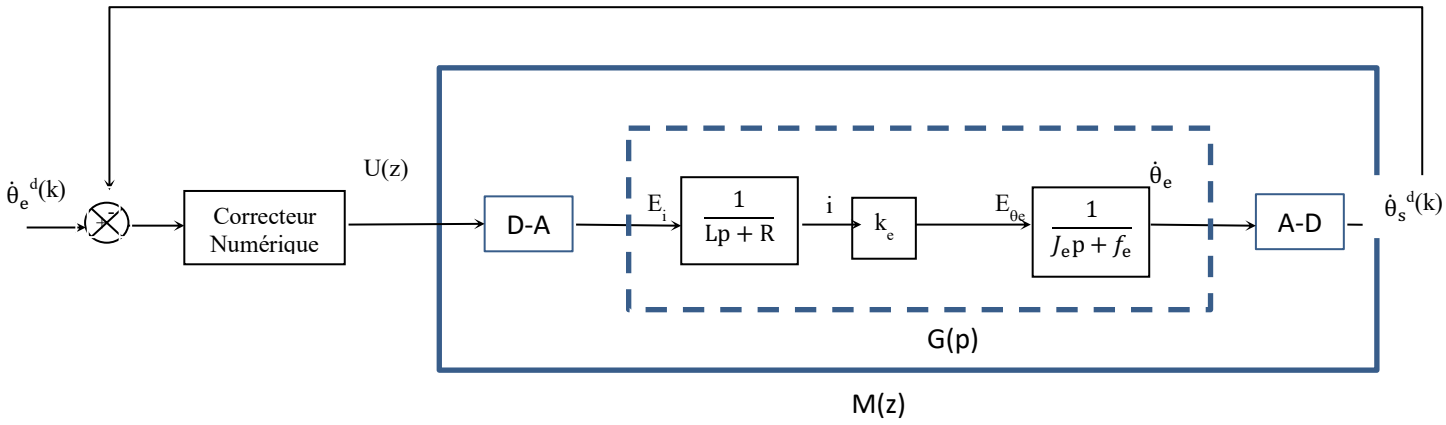


Figure. 4.5. Schéma en bloc d'une partie du système électromécanique en BF

Dans la suite, nous nous intéressons uniquement au moteur seul sans charge ni engrenage.

1.1 Calculer G(p) donné par la figure 4.5
1.2 Déterminer la période d'échantillonnage du système
1.3 En vous basant sur la figure 4.5 démontrer que la fonction de transfert de M(z) est égale à $\frac{z-1}{z} . TZ(\frac{G(p)}{p})$
1.4 Quels dispositifs utilisés dans la pratique sont omis dans la figure 4.5 ; quels rôles jouent-ils ?
1.1 On applique un régulateur P discret, analyser l'influence de K_p sur la stabilité, la rapidité et la stabilité du système

Pratique

A-Détermination de la période d'échantillonnage

A.1 Simuler le système sous Matlab Simulink en utilisant la fréquence d'échantillonnage précédemment déterminé en respectant le théorème de Shannon,

A.2 Faites varier la période d'échantillonnage et analyser son influence sur la stabilité du système.

B-Régulateur P

En appliquant un correcteur proportionnel discret au système

B.1 Simuler le système bouclé avec un gain K_p pour que le système soit stable, rapide et précis,

B.2 Analyser l'influence de la période d'échantillonnage sur la dynamique du système.

C-Régulateur PI

En appliquant un correcteur PI discret parallèle C(z)

$$C(z) = \frac{b_1 + b_0 z^{-1}}{1 - z^{-1}}$$

Où : $b1=K_p + T_i$ et $b_0=-Kp$

C.1 Trouver K_p et T_i en compensant les pôles de M(z) par les zéros du régulateur,

C.2 Analyser l'influence de la période d'échantillonnage sur la dynamique du système.

D-Régulateur PID

D1) Calculer la fonction de transfert discrète d'un régulateur PID ? (Voir Cours)

D2) Trouver les gains du régulateur PID pour que le système en BF soit stable,

D3) Analyser l'influence de la période d'échantillonnage sur la dynamique du système.