CAHIER des CHARGES :
« Mobile à roues omnidirectionnelles »

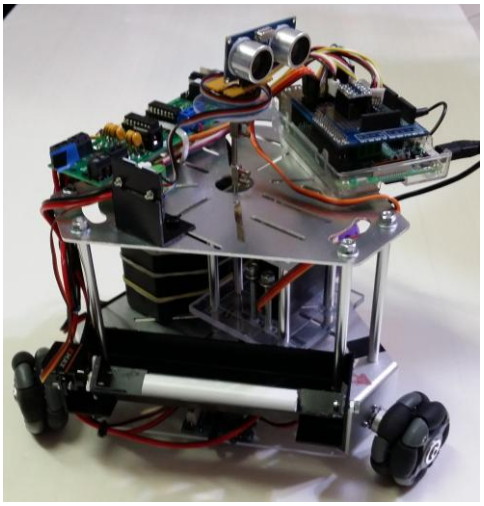
FONCTION GLOBALE

Réaliser un mobile à roues omnidirectionnelles qui peut se déplacer dans toutes les directions (sans faire de virage) quelque soit sa position initiale et qui réalise un suivi de ligne noire définissant son parcours.

Ce système sert dans une usine afin de transporter du matériel d'un atelier à un autre.
Ayant constaté que beaucoup d'accidents étaient liés aux déplacements entre ateliers et à la manutention de matériel, les dirigeants de cette usine ont décidé que certains matériels seraient transportés par des robots.
Cette solution présente différents avantages :

- Réduction du nombre d'accidents liés aux déplacements
- Gain de temps (les ouvriers n'ont plus besoin de se déplacer)
- Moins de pollution (utilisation de véhicules électriques autonomes au lieu de véhicule à essence)

Bien entendu, le robot étudié lors du projet SIN est une version simplifiée et réduite !!!

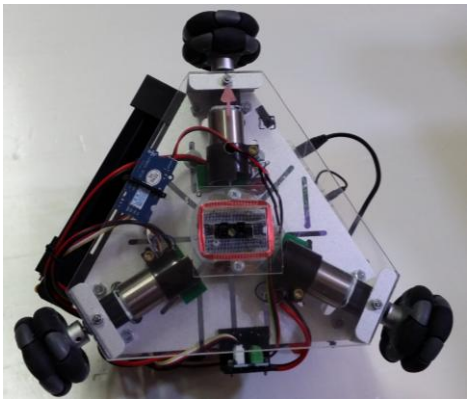


CAHIER DES CHARGES

Châssis avec 3 roues omnidirectionnelles

Est fourni un kit comprenant : un châssis sur 2 niveaux muni de 3 roues omnidirectionnelles et de 3 motoréducteurs, des modules pré-assemblés (capteurs de ligne, capteur RFID, capteur US, trappe oscillante avec servomoteur, un bloc permettant l'orientation de certains capteurs avec servomoteur), la visserie adaptée, . la Raspberry Pi3 avec shield de développement, la carte puissance de pilotage des 3 motoréducteurs, le haut parleur et l'alimentation (batterie + régulateur). L'élève doit procéder à l'assemblage de l'ensemble et aux réglages.

Energie



Le système est autonome en énergie : l'énergie est stockée dans une batterie.
La partie commande et la partie puissance du système fonctionnent respectivement, en 5V DC, et en 12V DC.
Un interrupteur est placé à la sortie de la batterie pour permettre ou arrêter l'alimentation totale du système.

Traction et Direction

Un Motoréducteur (Moteur à Courant Continu + Réducteur) est associé à chaque roue.
L'ensemble réalise à la fois la traction et la direction.
Grâce aux roues omnidirectionnelles, le mobile peut changer de direction sans réaliser de réel virage : elles permettent de se déplacer comme un « crabe » dans toutes les directions quelque soit la position initiale du mobile.

Différents mouvements possibles

Tous les mouvements doivent pouvoir être réalisés à différentes vitesses (définies entre 10% et 100% de la vitesse maximale).
Mouvements que le robot doit pouvoir réaliser:

- "rotate": tourner sur lui-même
- "go straight": avancer tout droit dans la direction d'une des 3 roues
- "linear motion": avancer tout droit dans une direction quelconque définie par un angle
- "stop motion": arrêter

Interface Homme / Machine (Affichage + Parole)

Au début, grâce à un haut parleur, le mobile « parle » en demandant : si il est en phase de tests ou en phase de fonctionnement réel. Dans le 2ème cas, il demande aussi le nom de l'atelier où il doit livrer du matériel (atelier A ou atelier B).
Le système demande sur l'écran du PC externe de rentrer les différents paramètres nécessaires pour réaliser la commande désirée (phase de tests ou phase de fonctionnement réel, nom de l'ordre de test, vitesse, nom de l'atelier…).

Pilotage à distance par commandes clavier via WiFi

Les ordres de pilotage sont tapés à distance sur un PC externe non connecté à la Raspberry Pi3 et transmis via WiFi.
Ces ordres correspondent :

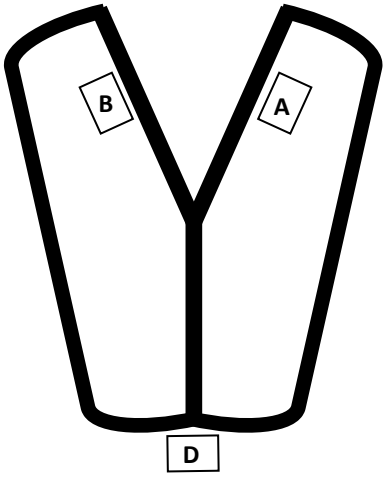
- Soit à des ordres de tests du robot pour les différents mouvements possibles décrits précédemment, dans le cas de la phase de tests.
- Soit au nom de l'atelier où le robot doit livrer du matériel, dans le cas d'un fonctionnement réel.

Phase de tests

Lors de la « phase de tests » les différents mouvements décrits précédemment peuvent être testés.

Détection de ligne noire

Lors de la phase de fonctionnement réel, le robot, muni de capteurs de ligne Emetteur / Récepteur Infra Rouge à réflexion, doit se déplacer de façon autonome, en suivant une ligne noire tracée sur le sol.



Détection de bifurcation

Grâce à un capteur RFID situé sous le robot (position fixe), le robot doit être capable de détecter la bifurcation de la ligne noire en 2 chemins différents.
Un tag avec un certain code est placé au niveau de la bifurcation.
L'atelier « A » se trouve sur le chemin de droite
et l'atelier « B » se trouve sur le chemin de gauche.

Changement de direction et Déplacement de la tige munie de capteurs

Au niveau du tag de bifurcation, le robot doit s'arrêter, puis grâce à un servomoteur, il va devoir déplacer une tige munie de certains capteurs (de ligne noire, de présence d'obstacles), afin que tous ces capteurs se retrouvent toujours dans la même position par rapport à la ligne (et non pas à la même position par rapport au robot).
Ensuite, il doit redémarrer, mais pour emprunter l'un des 2 chemins, le robot ne doit pas faire de réel virage, il doit changer de direction en se déplaçant comme un « crabe ».

Détection de l'atelier

Grâce à un capteur RFID (identique au capteur de détection de bifurcation), le robot doit être capable de détecter l'atelier sélectionné au départ. Devant chaque atelier il y a un tag ayant chacun un code différent.

Livraison de matériel

Quand le robot est arrivé devant l'atelier sélectionné, il doit s'arrêter, et ouvrir la trappe pour qu'on puisse récupérer le matériel livré. La trappe est pilotée par un servomoteur. Après 10s, il referme la trappe avant de redémarrer. Il repart sur son parcours vers le local de départ « D ».

Détection du local point de départ « D » (facultatif)

Grâce à un capteur RFID, le robot doit être capable de détecter le local de départ « D » afin de s'arrêter. Devant ce local il y a un tag ayant chacun un code différent.

Détection d’obstacles

Le robot est muni d’un capteur Ultra Sonique permettant de détecter un obstacle qui se trouve sur son parcours (autre chariot, personne…). Dans le cas de présence d’obstacle, le robot doit s’arrêter jusqu’à ce que l’obstacle ait disparu depuis 5s puis il doit redémarrer.

Programmation

Le programme est réalisé en langage Python sur la Raspberry Pi3 et utilise des « Threads » pour définir la classe « Motor ».
Une partie du programme est fourni : essentiellement la définition de la classe « Motor ».