Première partie

Partie robotique

Dans cette partie nous allons tenter de créer un nouveau système directionnel qui réunira les caractéristiques du cahier des charges présenté prédemment.

Chapitre 1

Un système directionnel parfait

Le fonctionnement de notre système est simple, c'est un robot aspirateur dirigé optimisé, en effet le plus gros problème du système dirigé est son incompatibilité avec de nombreuses pièces remplies de nombreux obstacles. Le robot aura en effet tendance à ne pas passer à certains endroits de la pièce et même au bout d nombreuses minutes de fonctionnement à ne pas passer à ces endroits. Le principe de notre robot est de se diriger de manière dirigé ce qui ne limitera pas la vitesse du processeur. Ce surplu de mémoire vive n'est las de trop ! ! ! Le robot pourra en effet l'utiliser pour améliorer le nombre de mesures de la boussole qui pourra ainsi être plus précise. Le robot devra commencer dans un coin de la pièce puis se déplacera tout droit, tournera puis réavancera tout droit. En arrivant au coin opposé il est fort probable que le robot ait traversé 70% de la pièce . Si la pièce contient beaucoup d'obstacles il faudra relancer le programme d'un autre coin pour garantir la propreté.

Chapitre 2

Construction de notre robot aspirateur

Pour tester notre système nous avons décidé de le modéliser avec une carte Arduino. Cette petite carte programmée permet de créer des circuits électroniques facilement sans forcément s'y connaître très bien en électronique. La carte, dévelopée en open source est livrée avec un logiciel permettant de téleverser un programme écrit en language Arduino, sur la carte. Pour la conception de notre robot nous avons utilisé du materiel que nous avons emprunté au labo de SI des classes préparatoires de notre lycée. Nous avons utilisé un chassis de robot sur lequel nous avons monté deux moteurs et deux roues, un shield moteur pour alimenter les moteurs avec plus de puissance, une boussole électronique et trois capteurs infrarouge. Nous avons ramené le robot deux fois en cours de TPE, et nous avons finalisé le montage. Notre robot enfin construit nous avons entamé la partie la plus fastidieuse de notre projet : la programmation. Même si Kilian avait quelques notions en language Arduino, nous étions tout les deux débutants. Nous avons dabord éssayé de trouver des programmes pour les différents capteurs et composants pour les faire fonctionner individuellement dans un premier temps. Nous avons rencontré beaucoup de problèmes avec la boussole qui est clairement l'élement le plus difficile à programmer du robot. Après de nombreuses recherches sur internet, nous avons réussi à trouver des solutions à la majoritée de nos problèmes. Nous avions comme projet de déguiser notre robot en R2D2, référence évidente au monde fantastique de StarWars, mais le responsable de ce projet, Carl, mon correspondant allemand est malheureusement parti trop tôt. Ne disposant pas de capacités apparentes en dessin, nous avons très vite abandonné le projet.

Chapitre 3

Le Programme

Du fait que le language Arduino soit plus compliqué que le language Python par exemple, nous n'allons pas rentrer dans les détails dans cette partie. Ce programme comme la majorité des programmes Arduino, ce programme est structuré en trois parties : Déclaration des variables, setup, loop.

Dans la partie de déclaration des variables, on donne à la carte les emplacements des différents composants à l'aide de la déclaration int potas = 2 ; Le int de cette séquence signifie que les valeurs renvoyées par cette variable sont entières et comprises entre $-2^(15) et 2^(15)$. Potas est le nom donné à la variable. A2 signifie que le potentiomètre est branché sur le pin analogique 2. Ce type de déclaration est également utilisé pour la déclaration de variables globales et on remplacera le A2 par une valeur. Dans cette partie on déclare également les librairies utilisées et les variables à changer au moment du démarrage du programme comme PCIG qui est changé en 4. La première ligne définit si le programme tourne en mode initialisation qui permettra de régler la vitesse des moteurs.

La fonction setup ne renvoyant pas de variables (void) ne va fonctionner qu'une seule fois au lancement du programme. Nous y avons initialisé les pins : pinMode(PMDS, OUTPUT) ; et nous avons intégré la boucle de calibrage de la boussole.

La fonction loop est le coeur du programme, elle va exécuter les différentes fonctions comme ici la fonction dirig() ;

3.1 compas()

La fonction compas() renvoie l'angle du robot par rapport à l'angle défini dans la phase de calibrage sous la forme d'une valeur entière en faisant appel aux fonctions fournies par la librairie de la boussole

3.2 moteurs()

La fonction moteurs est la fonction permettant de faire tourner les roues. On définit le sens, la vitesse et l'utilisation du frein pour chaque roue et pour chaque cas : si la valeur donnée en argument est négative, le robot recule, sinon il recule. De nombreuses autres fonctions de notre programme font appel à celle-ci pour mettre le robot en mouvement.

3.3 go()

La fonction go est celle qui en fonction de l'angle donné en argument va faire appel à la fonction moteurs() pour mettre le robot en mouvement. Si l'argument vaut 90 ou 270 le robot va tourner à droite ou à gauche de 90 degrés tout en diminuant sa vitesse pour permettre une meilleure mesure de l'angle. Si l'argument vaut 0 le robot va avancer en faisant tourner les roues à une certaine vitesse définie par la fonction d'asservissement k(), permettant de toujours garder le même cap

3.4 k()

La fonction k va comparer le cap que le robot suit avec le cap qu'il devrait suivre et renvoyer un coefficient k qui permettra l'asservissement indépendant des roues pour garder le même cap.

3.5 dirig()

La fonction dirig() est le coeur peincipal du programme c'est lui qui définit les actions effectuées par le robot :

3.6 arret()

La fonction arret est la plus simple ! C'est elle qui fait arrêter le robot.

Chapitre 4

Conclusion

La conception de ce système directionnel nous a permit d'observer les différents avantages et inconvénients de notre système. De plus nous avons compris que la qualité du système directionnel n'est pas le seul élement qui permet au robot de se déplacer correctement. En effet nous avons par exemple rmarqué que le type de sols est également très important : le robot déviera plus facilement de sa trajectoire sur du carrelage car les rainures peuvent influencerr sur la trajectoire du robot qui chez nous était très sensible. Nous avons aussi dû faire face à des problèmes techniques comme l'imprécision des composants ou l'arrondi des valeurs. Notre boussole par exemple nécessitait une phase de calibrage et n'était malgré ça pas précis à moins de deux degrés. La carte Arduino arrondi quand à elle les valeurs renvoyées par une fonction à l'entier près ce qui influence tout de même les calculs comme par exemple dans les calculs du coefficient d'asservissement. Nous avons tiré de notre étude que ce système que nous avons imaginé était assez facile à mettre en plac, ne necessitait que peu de ressources et éffectuait bien ses tâches dans une grande partie des cas. Cependant il ne s'adapte qu'aux pièces rectangulaires et la moindre erreur notamment au niveau de la boussole est fate à toute la suite de la progression du robot. Nous pensons que ce type de robot est une bonne alternative au système fourmi décrit dans la partie théorique.