POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Département de génie informatique et génie logiciel

INF1900
Projet initial de système embarqué

Rapport final de projet

Projet dans SimulIDE

Équipe No **0320**

Section de laboratoire **01**

Einstein Franck Tiomo Epongo
Moussa Abdillahi Daoud
Zabiullah Shair Zaie
Zakarya Khnissi

22 Avril 2021

1. Description de la structure du code et de son fonctionnement

1.1 Code principal

En ce qui concerne la fonction `main()`, on a jugé que la méthode de scrutation convient mieux à nos besoins pour des raisons de performance ainsi que la simplicité de code qui rend le débogage beaucoup plus facile. Compte tenu des nombreuses possibilités d'appuis sur le clavier, le `main()` est munie d'une instruction `switch` qui permet d'exécuter le code correspondant à l'appui d'une touche donnée. Pour l'implémentation de notre programme, l'utilisation d'une deuxième instruction `switch case` a été considéré pour le bon fonctionnement du bouton R et dans le but de rendre le code soit maintenable et facile à suivre. En effet, la première instruction `switch` sert à identifier et afficher le bouton appuyé et la deuxième, pour effectuer son comportement. Enfin pour que tout se déroule bien, la fonction `detectPressedButton()` est appelée avant le `switch case` pour repérer le bouton appuyé pour permettre à l'instruction `switch` de comparer la variable bouton et exécuter le comportement désiré de chacun des boutons.

Une fois que l'on a vérifié si une touche a été appuyée, les distances qui séparent le robot des obstacles qui l'entourent sont mesurées. Une section plus détaillée sur la mesure des distances suivra. Pour chacune des distances mesurées, on détermine à quelle catégorie elle appartient pour pouvoir faire un éventuel affichage et ultimement permettre au robot d'exécuter une manœuvre d'évitement selon la combinaison de catégories obtenues.

C'est la deuxième `switch case` de la fonction `main()` qui permet de faire un affichage en fonction de l'option d'affichage déterminée par l'utilisateur, en appelant la fonction `display()`.

1.1 Fréquence de lecture

La lecture faite sur les capteurs se fait à une fréquence configurable lors d'appuis sur le clavier. Pour mettre en évidence cette fréquence, nous démarrons une minuterie à chaque nouvelle itération de la boucle infinie du programme. De cette façon, le déroulement du programme en entier se fait au même rythme que la lecture des distances. Cependant, le délai qu'on laisse n'est pas un temps pendant lequel le programme se fige. En effet, dans le corps du `do {} while(lecture == 0)`, on vérifie si un bouton du clavier a été pressé.

1.2 Clavier

Le code du clavier est implémenté dans la fonction `detectPressedButton()`, on a choisi que le clavier fonctionnera par scrutation pour des raisons de performances donc notre code principal vérifie toujours si un bouton est appuyé en appelant la fonction dans plusieurs endroits dans le code.

La fonction `detectPressedButton()` est responsable d'initialiser les ports, elle initialise d'abord les ports liés aux lignes du clavier en sortie. Quand un bouton est appuyé le courant va passer par la colonne correspondante pour être détecté par un des ports lies aux colonnes, une fois qu'une broche détecte du courant, la fonction réinitialise les ports de façon que les ports liés aux colonnes du clavier soient en sortie et ceux liés aux lignes en entrée, avec les conditions de détection de courant la fonction fait l'intersection des lignes et colonnes pour savoir le bouton appuyé.

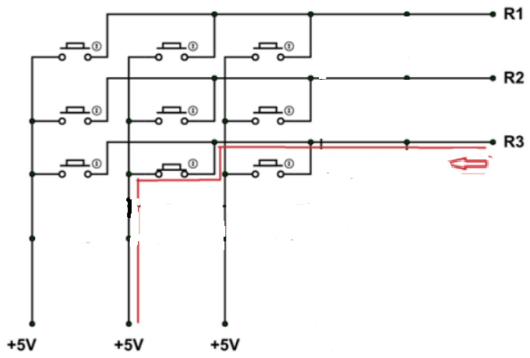

Figure 1: Ports lies aux lignes en sortie

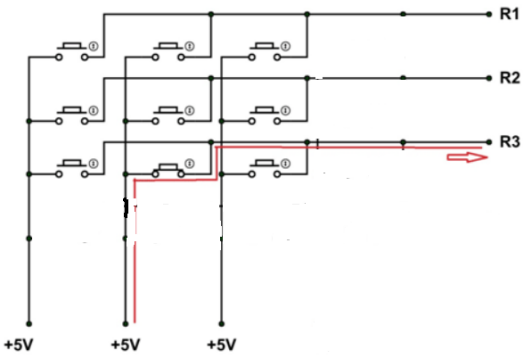

Figure 1: Ports lies aux colonnes en sortie

1.3 Capteurs de distances

Les lectures de distances sont faites à la fréquence à laquelle le programme s'exécute, soit 1, 2 ou 4 fois chaque seconde. Un tableau de distances de taille 3 conserve les distances évaluées par le robot sur chaque côté. Le calcul se fait dans une boucle de 3 itérations dans laquelle chacune des itérations correspond à une lecture sur un capteur. Pour une valeur x de conversion donnée par le convertisseur externe ou interne via la fonction `computeConversion()`, la formule qui permet de trouver une approximation de la distance à laquelle elle correspond est :

$$28.998 \times \frac{5x}{255}^{-1.141}$$

1.4 Afficheurs sept Segments

La façon par laquelle on a procédé pour résoudre le problème des afficheurs afin d'implémenter une interface capable de prendre la distance relevée par les capteurs afin de les afficher en décimal ou en hexadécimal le cas échéant, est l'implémentation de deux fonctions nommées `leftDisplays7()` pour les deux afficheurs à gauche du pont-H et `rightDisplays7()` pour les deux afficheurs à droite du pont-H. Ces fonctions prennent en paramètre deux arguments. Le premier argument est la distance de type `uint8_t` et le deuxième est une variable booléenne qui indique le mode d'affichage des afficheurs (décimal ou hexadécimal). Cette interface est appelée dans l'exécution des manœuvres pour afficher la puissance des roues.

1.5 Mode manœuvre

Pour exécuter les 5 manœuvres, on a opté pour implémenter une générale qui prend en considération toutes les combinaisons des valeurs de puissances possibles pour permettre plusieurs mouvements plutôt que d'avoir plusieurs fonctions pour chacun des mouvements. La fonction est implémentée de sorte à satisfaire toutes les conditions de l'énoncé à l'aide des instructions conditionnelles. Elle prend en paramètre le pourcentage de puissance des roues qui seront converties en rapport de 255 et le faire changer de signe si une valeur négative a été passée en paramètre pour ensuite appeler la fonction de PWM.

2. Expérience de travail à distance

Le projet a été une expérience vraiment enrichissante pour chacun d'entre nous. Au-delà des compétences techniques développées, le projet a également été bénéfique d'un point de vue personnel. En effet, des qualités telles que la rigueur et la discipline étaient d'une importance cruciale pour la progression du projet.

Étant donné la complexité du projet, l'idée première pour commencer le projet était de s'assurer que tous les membres de l'équipe en avaient une compréhension claire et commune. Cette étape passée, nous avons jugé bon de nous subdiviser en sous-équipes de 2 membres et ainsi faire une séparation des tâches. Nous organisions régulièrement des rencontres sur le serveur Discord avec pour but que chacun des membres présente le travail qu'il a accompli ainsi que les difficultés qu'il a rencontrées.

Pour assurer une progression régulière du projet, chacune des sous-équipes se donnait une date limite pour remettre un code fonctionnel de leur tâche sur le serveur git. De cette façon, on accordait au projet la priorité dont il avait besoin pour être menée à terme.

Le passage d'une étape à l'autre n'a pas toujours été facile lors de la réalisation du projet, mais nous pouvions toujours compter sur l'aide de nos coéquipiers ou alors, pour des questions plus compliquées, sur celle des chargés de cours et du professeur qui étaient actifs sur le serveur Discord.