POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Département de génie informatique et génie logiciel

INF1900
Projet initial de système embarqué

Rapport final de projet

Simulation du comportement du robot

Équipe No < **4081** >

Section de laboratoire < 03 >

< Philippe Désilets, Maximiliano Falicoff >
< Guillaume Nadeau, Alexis Thibeault >
17 Avril 2020

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

Commençons par la structure de notre code. En plus d'utiliser les anciens codes fait durant le cours (en librairies), nous utilisons un *main.cpp* ainsi que *manoeuvre.(cpp et h)*. Nous avons utilisé le principe de machine à états pour compléter ce projet. En effet, dans le *main.cpp* notre robot possède un *enum State* qui contient plus de 8 états. Tout au long du code nous utilisons la variable *statePres* pour conserver l'état présent du robot. Ensuite, pour gérer les changements d'états du robot, nous avons utiliser un *switch case statement* englobée dans une boucle *while* infinie. Il y a donc un *case* pour chaque états. Tous les *case* sauf celui pour l'état *detect* fonctionne selon le même principe. Lorsqu'ils sont exécutés, les *case* débutent par appeler une fonction qui va accomplir la manoeuvre souhaité (selon l'état présent). Puis, la variable *statePres* est réinitialisé à l'état *detect*. Pour le *case detect*, on débute par assigner à un *uint8_t* appelé *man* la valeur de retour de la fonction *detection()* situé dans *manoeuvre.cpp*. La variable *man* représente la manoeuvre à exécuter (1,2,3,4,5, 6 ou non évaluée). Elle est utilisé dans une série de *if else statement* afin de mettre à jour *statePres*. Voilà pour la description de la structure du programme.

Dans un même ordre d'idée, si la structure du programme dépend grandement du *main.cpp* il est maintenant temps d'expliquer le fonctionnement de *manoeuvre.cpp*. Dans ce fichier, nous retrouvons toutes les fonctions appeées par le *main.cpp* et plus. Ce fichier contient les fonctions suivantes: *detection(), man1()*, *man2(), man3(), man4(), man5(), man6(), manIconnue(), finman()* et *choixManoeuvre()*.

Pour continuer, la fonction *detection()* contient un *while* qui est exécuté tant que le bouton poussoir n'a pas été appuyer. *detection()* retourne un *uint8_t* qui correspond au choix de la manoeuvre selon la combinaisons des distances. Pour ce faire, la fonction appel *sonar()* (tiré de notre librairie) qui permet d'assigner à la variable *distance* (de type Distance, une struct définie dans sonar.h qui contient trois doubles, un pour chaque sonar). Ensuite, avec la variable *distance* on assigne à *choix* la valeur de retour de *choixManoeuvre(distance)* qui contient des *if else* afin de retourner la manoeuvre à effectuer selon la combinaison de *distance*. Voilà comment la fonction *detection()* est capable de retourner *choix* qui porte la valeur de la manoeuvre à faire.

D'autre part, les fonctions *man* (de 1 à 6) sont plutôt semblables. En effet, elles commencent par initialiser les ports (on set DDRA et DDRC en sortie et DDRD avec 0b11110111 touts les pins en mode sorti sortie sauf le bouton poussoir dans la fonction *initman()*), puis elles appellent *afficherManoeuvre()* (prend en argument le numéro de la manoeuvre) qui est responsable pour afficher «Manoeuvre X» sur l'écran LCD. Ensuite, elles entreprennent la manoeuvre à faire en appelant la fonction *ajustementPWM()* et *affichage7FPS()* pour afficher en temps réel le comportement des PMW sur l'afficheur 7 segments. Finalement, les fonctions *man* (de 1 à 6) appelle *finman()* qui à son tour appel *stopPWM(*arrête les moteurs*), clear7Seg(*réinitialise l'afficheur 7 segments*), clearDispl(*réinitialise l'écran LCD*) et* donne la valeur *false* à *bouttonPoussoir*. Pour la fonction *manInconnue()*, plutôt que d'afficher «Manoeuvre X» sur l'écran LCD, elle fait afficher «Combinaison non evaluee». Ensuite, elle attend 2000 millisecondes puis elle appelle *finman()* comme les autres fonctions de manoeuvre.

Il faut maintenant parler des fichiers PWM_roues, il est composé de deux fonctions. On a premierement *ajutementPWM()* qui prend en paramètres des int (pour garder le signe signifiant, avance ou recule) correspondant au pourcentage de la roue droite et gauche. On prend ensuite les valeurs absolues de ces pourcentages et on les met sur 255 que l'on assigne aux registre OCR1A et OCR1B, puis en dépendance du signe des paramètres, on allume la led rouge (recule) ou droit (avance) pour chaque roue. On set notre timer1 avec un clock/8. La fonction stopPWM fait juste reset tous nos registres à 0 et éteint la led.

Parlons maintenant des méthodes du fichier sonar, nous avons ici deux fonctions *evaluerDistances() et sonar()* qui retournent un objet de type Distances. Lorsque l'on veut savoir la distance et l'afficher à l'écran LCD, on appelle *sonar(),* cette fonction va set le timer 1 avec un clock/64 et puis on appelle la méthode *evaluerDistances()* qui va envoyer un écho pour chaque sonar un à la fois et débute le timer TCNT1 lorsque le pin est high. Le timer continue jusqu'à que le trigger vient à 0. On peut ensuite calculer la distance avec TCNT1 * 8/580 pour avoir la distance en cm.

Un choix est survenu lors de la programmation du robot, l'utilisation entre les minuteries ou les delays, on a remarqué que chaque fois que l'on veut afficher la vitesse à l'écran, on doit laisser passer un certain temps. Notre fonction qui *affichage7FPS()* prend trois paramètres: les vitesses des deux roues et le temps que l'on veut afficher ces vitesses. On affiche les displays un à la fois avec un delay de 1 millisecondes entre chaque display, tout ça dans un for allant de 0 au paramètre du temps / 5 (5 car on a 1ms de delay par display), on peut alors gérer tous nos delays de toutes nos manoeuvres dans cette fonction en ajustant le paramètre au temps nécessaire du delay.

2. Expérience de travail à distance

Suite à l'arrêt total des activités à Polytechnique, nous avons vu notre plus grand lien d'équipe disparaître, c'est-à-dire nos rencontres physiques lors des laboratoires. Ainsi, nous devions absolument trouver des alternatives pour remplacer cette lourde perte. Nous avons donc décidé d'utiliser plusieurs outils pour y arriver.

Messenger : Bien évidemment, l'un des premiers moyens employés a été l'application de messagerie par Facebook « Messenger ». Cet outil très accessible nous a permis de communiquer rapidement les uns avec les autres pour nous permettre de nous rediriger vers d'autres plateformes plus adéquates pour le travail en équipe. Elle permet aussi d'envoyer des messages simples comme de petites questions ou encore des photos pour nous entraider.

Discord : Cette plateforme qui devient de plus en plus populaire pour les appels vocaux par Internet nous a permis de communiquer plus fluidement entre nous. De plus, ce programme nous permet, en quelques clics, de faire un partage d'écran ce qui nous rend la tâche bien plus facile lorsqu'il est temps de montrer notre progrès. Aussi, ce programme permet de créer plusieurs salles pour communiquer ce qui devient utile pour un projet divisé en plusieurs parties différentes.

Trello : Un site web moins populaire que les deux derniers outils, mais qui a été tout aussi pratique pour la réalisation du projet. Cet outil a permis la gestion en ligne de notre projet. Le projet a rapidement été divisé en plusieurs tâches ce qui nous a permis d'avancer plus méthodiquement dans notre projet. D'ailleurs, il était plus facile de voir la progression globale du projet, puisque l'on pouvait voir toutes les tâches en même temps ainsi que leur avancement.

Un autre problème pour certains d'entre nous était de ne pas avoir accès à un ordinateur Linux, parce que les ordinateurs de l'école nous étaient bien évidemment plus disponibles. Il va de soi qu'une machine virtuelle n'est pas aussi performante et pratique que l'appareil informatique en tant que t-elle. Ainsi, pour quelques personnes, il était plus difficile d'utiliser le simulateur ainsi que certains programmes utiles pour le projet.

Bref, il nous était possible de réduire les mauvais impacts du travail à distance avec des outils disponibles sur Internet pour nous permettre de terminer le projet malgré tout.