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 <113114>

Section de laboratoire <3>

<Nathan Ramsay-Vejlens, David Saikali,
Agnès Sam Yue Chi, Jefferson Lam>

17 Avril 2020

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

Il est possible de séparer le travail selon les différentes composantes du circuit dans la simulation, ainsi que le fonctionnement global de la machine à états.

Le code est séparé en plusieurs parties indépendantes que nous avons ensuite intégrer et encapsuler dans une classe principale.

D'abord, nous disposons de la librairie composé de nombreuses fonctions utiles que nous avons choisies du code commun de l'équipe, telles que des fonctions pour le contrôle des moteurs, des delay, des interruptions et des initialisations. Cette librairie se trouve dans /tp/tp7/codeCommun/lib_c. Il faut donc s'assurer de la compiler avant de compiler la solution du projet.

Deuxièmement, nous avons ajouté à la librairie une classe qui permet la fonctionnalité du LCD. Ce dernier nous est utile pour afficher entre autres les mesures prises par le robot et la vitesse de celle-ci.

Ensuite, nous avons encapsulé la fonctionnalité des sonars dans une classe afin de pouvoir fournir une interface qui permet son utilisation.

Finalement, nous avons créé une classe robot où nous avons intégré le LCD et le sonar. Cette classe s'occupe de la logique de la tâche principale et elle unit toutes les différentes composantes indépendantes dans certaines actions du robot.

Affichage LCD

Les ressources fournies liées au LCD contiennent plusieurs fonctions utiles quant à la manipulation de l'affichage. Une variable globale *disp* de type LCM, c'est à dire le type d'objet contrôlant l'écran LCD, est définie dans projet.cpp afin de pouvoir afficher les éléments voulus sur l'écran.

Affichage 7 segments

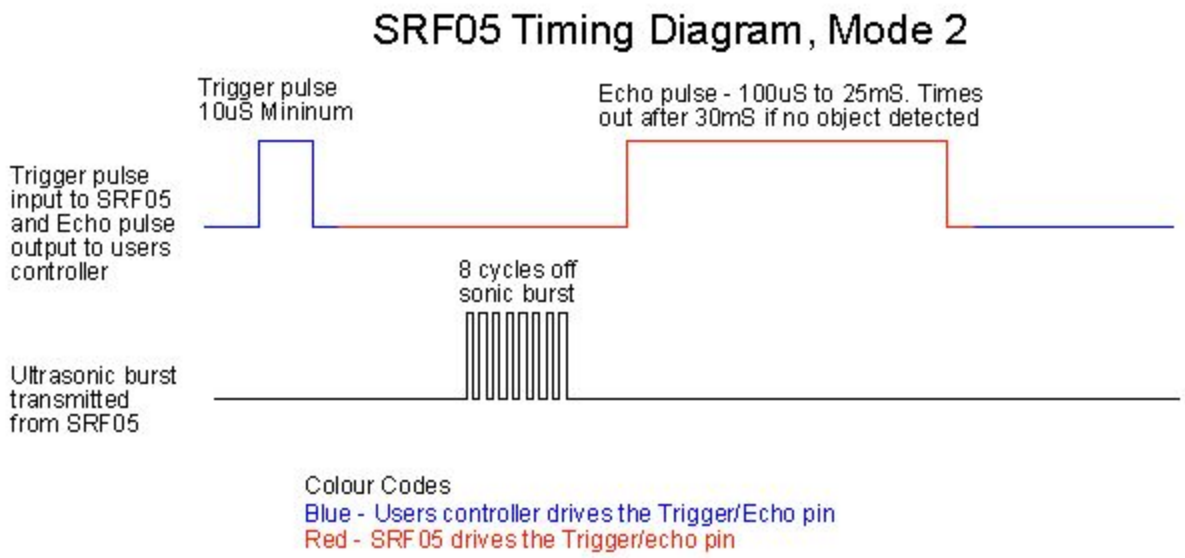
Pour utiliser les afficheurs 7 segments, il fallait alterner rapidement entre les 5 afficheurs et afficher les bons segments au bon moment. Pour réaliser cette tâche, nous avons utilisé les interruptions et nous avons regroupé les changements de valeurs du PORTC en une fonction nommée affiche(). Cette abstraction facilitait grandement la compréhension. De plus, la fonction activerAfficheur() permet d'initialiser DDRA et DDRC en mode sortie. L'afficheur allumé en premier est en fait le 5ième puisque le changement d'afficheur se fait tout juste avant l'affichage. La première itération passe donc du cinquième afficheur au premier.

Les attributs pGauche_ et pDroite_ du robot sont utilisés pour l'affichage des PWM afin de déconstruire le nombre en deux chiffres, l'un étant les dizaines et l'autre les unités.

Détection par les sonars

Afin de calculer la distance que mesure le sonar, nous nous servons de la largeur du signal *Echo*. Sachant que chacun des *Echo* des 3 sonars sont connectés aux pins A0-A2, nous pouvons déduire que la réception du signal se fait lorsque la valeur du pin est à 1, soit le front montant du signal. Dès la réception du signal par principe de scrutation, nous utilisons le timer TCNT1 afin de calculer la largeur du signal. Nous arrêtons le timer lorsque le signal atteint son front descendant, et nous divisons le signal par un facteur de conversion pour obtenir la distance en mètres.

Ensuite, nous avons créé une classe pour encapsuler l'utilisation des sonars. Elle est composée de 3 attributs *distances* et de 3 méthodes principales : *calculerDistance(), detecterObjets(),* et *afficherMesures().* La méthode *calculerDistance()* fait la procédure décrite ci-dessus pour calculer la distance entre le robot et l'objet virtuel. La méthode *afficherMesures()* s'occupe d'afficher les mesures prises par les sonars sur l'écran LCD, en bonne et dûe forme. Finalement, nous avons la méthode *detecterObjets()* qui fait appel aux deux autres méthodes pour effectuer la combinaison des deux tâches. Nous obtenons ainsi une classe Sonar que nous pouvons facilement réutiliser dans d'autres parties du logiciel pour avoir accès aux distances.



Contrôle des “moteurs’ durant les manoeuvres

Dans la classe Robot, nous avons défini la manipulation des moteurs selon chaque manoeuvre voulue, et ce, dans 6 fonctions différentes pour faciliter leur appel. Ces fonctions consistent simplement à contrôler le PWM des “moteurs” (représentés par les oscilloscopes et les LED) en utilisant les fonctions de *controleMoteur* dans la librairie. De plus, le nom de la manoeuvre en cours est affiché sur le LCD.

Machine à états

Afin de contrôler le fonctionnement du circuit, une machine à états a été établie avec un état pour chaque mode possible: la détection, les 6 manoeuvres configurées, ainsi qu’un état pour toutes les autre combinaisons non configurées du robot. L’état dans lequel le robot se trouve dépend entièrement de la distance aux objets que l’on détecte grâce aux sonars. Le robot va maintenir son état de détection jusqu’à temps que l’on appuie sur le bouton-poussoir. Dès que ceci est fait, il prendra l’état le plus approprié selon ce que les sonars détectent autour d’eux. Le robot exécutera ensuite le code qui est associé à son état présent. Ce code est généralement constitué d’ajustements divers aux moteurs afin de changer la trajectoire du robot et éviter les obstacles possibles. Une fois ce code terminé, le robot rentre à nouveau en mode détection et y reste jusqu’au prochain moment où le bouton poussoir se fait appuyer.

2. Expérience de travail à distance

Heureusement, grâce aux outils fournis, le projet était bien adapté pour le travail à distance. Le plus grand défi n’était donc pas l’accès aux ressources, mais l’établissement d’une routine de travail. Puisque les séances de laboratoire sont devenues libre-accès grâce à la plateforme Discord, notre équipe n’avait plus de réunion bi-hebdomadaire. Nous avons donc initialement séparé les tâches selon les différentes composantes du robot en utilisant différentes branches de travail.

Un autre désavantage de l’impossibilité de rencontre physique est la communication, ou l’accès aux moyens de messagerie. Un des membres de notre équipe avait une situation particulière dû à la pandémie en cours. Il n’était pas à sa résidence et n’avait donc pas accès à un réseau Wi-Fi illimité. Il avait accès au réseau à une fréquence irrégulière, mais au moins 2 fois par jour. À ces occasions, il communiquait avec le reste de l’équipe pour indiquer son avancement et quelle partie il pensait entreprendre le lendemain.

Toutefois, l’accès à la simulation a simplifié grandement la programmation du robot, puisque le robot physique n’était plus nécessaire pour tester du code. Le fait que chaque membre pouvait librement essayer de rouler son programme est en effet plus simple que devoir

partager deux robots pour quatre personnes. On pourrait néanmoins noter qu'il est moins satisfaisant d'appliquer son projet final sur une simulation que sur une machine mobile.