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 5367

Section de laboratoire **2**

Hee-Min Yang
Chun Yang Li
Jean Janssen
Adam Halim

8 décembre 2020

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

Notre code est séparé en plusieurs classes selon les différentes composantes à implémenter ainsi que des encapsulations de méthodes utiles pour le bon fonctionnement du système. Voici une brève description des principaux éléments de chaque classe ainsi que leur fonctionnement.

Can

Cette classe sert de convertisseur analogique numérique. Elle nous permet de prendre un voltage (signal analogique) et de le convertir en signal numérique. En l'occurrence, on l'utilise pour la source de tension variable pour l'horloge afin de réguler sa vitesse. Cette classe a été fournie.

Clock (horloge)

Attributs:

La classe contient un pointeur vers un port du microcontrôleur. Il y a deux attributs de type uint8_t représentant les pins pour le PWM et le pin pour réinitialiser l'horloge. Il y a également un objet de type "can" et un de type "sonar". Le sonar sert à interrompre la simulation lorsqu'un objet est assez proche de celui-ci.

Fonctions publiques:

Il y a différentes méthodes servant à manipuler l'affichage de l'horloge, notamment des méthodes servant à initialiser, arrêter et réinitialiser l'horloge. Il y a également une méthode pour définir la fréquence de l'horloge.

Customprocs/lcm (LCD)

Cette classe se charge de l'affichage sur l'écran LCD. Nous avons retiré la protection contre la copie dans le fichier .h, car lorsqu'on essaie d'appeler le constructeur, la protection contre la copie est appelée à la place. Autre que cela, rien n'a été modifié du code qui nous a été fourni.

Debug

Cette classe sert à déboguer le programme. Elle permet d'afficher des messages à partir d'un objet de type uart. On peut faire appel à deux méthodes de débogage, une pour envoyer un tableau de char, et une autre pour envoyer un uint8_t.

Eeprom

Attribut:

L'attribut "endPointer_" sauvegarde l'adresse eeprom après la fin de la dernière instruction. Lorsqu'une instruction est ajoutée, la valeur d'"endPointer_" augmente et lorsqu'une instruction est supprimée, celle-ci diminue.

Fonctions:

Cette classe manipule les instructions qui sont sauvegardées dans l'eeprom de l'atmega. Effectivement, les méthodes disponibles nous permettent d'ajouter et de supprimer des instructions ainsi que de lire le temps ou les instructions matérielles d'une instruction. Nous avons aussi une fonction de triage qui permet de trier les instructions en fonction du temps de départ de la simulation.

Keyboard (clavier)

Attributs privés:

La classe contient principalement un enum de uint8_t qui représente chaque touche du clavier ainsi que les variables pour les pins et les ports.

Fonctions publiques:

La classe contient un constructeur par paramètre qui permet d'assigner les ports de l'atmega aux ports "s0" et "s1" du démultiplexeur et du multiplexeur ainsi que le port "y" du multiplexeur.

Elle contient aussi la méthode "readKey" qui retourne le char selon la touche du clavier appuyée par scrutation. Les touches du clavier sont en fait des interrupteurs qui laissent passer un courant lorsqu'elles sont appuyées. Donc en ouvrant les portes du multiplexeur et du démultiplexeur à l'aide des combinaisons de "s0" et "s1", on peut vérifier si la touche est appuyée en regardant si le courant passe.

LEDBar (barre de DEL)

Attributs:

La classe contient les pins pour les registres de décalage, soit shc, stc, ds, mr et oe. De plus, il contient un pointeur pour le port utilisé pour les registres à décalage et un attribut de type uint32_t qui représente l'état des portes ensembles.

Fonctions:

Les fonctions de cette classe permettent d'ouvrir ou de fermer une porte spécifique.

Servomotor (servomoteur)

Attributs:

Contient un attribut de type uint8_t qui représente le pin qui contrôle le servomoteur et un pointeur vers le port contenant ce pin.

Fonctions:

Les méthodes de cette classe nous permettent de changer l'angle du servomoteur et de convertir les entrées de l'utilisateur en angle pour alors appeler la méthode mentionnée précédemment.

Sonar

Attributs:

Contient les variables volatiles pour affecter les ports et les pins. De plus, il y a la variable volatile "lastDistance_" qui nous permet de retourner la dernière distance captée par le sonar.

Fonctions:

Contient deux ISR qui permettent de détecter un objet en recevant une tension par interruption. Lorsqu'un objet est détecté, la valeur de "lastDistance_" est modifiée selon la tension de la source de tension variable reliée au sonar.

Time

Cette classe sert principalement à s'assurer que le temps entré par l'utilisateur est valide et à effectuer des conversions. Une de ses méthodes permet de convertir une série de caractères représentant le temps en format "HHMM" en temps de type uint16_t. Cette classe est utile lorsqu'on va définir des heures (ex. option programmer une action) à partir du clavier.

Uart

Cette classe contient les méthodes qui permettent d'écrire et de lire des instructions dans l'eeprom ainsi que l'affichage de texte ou des données dans le terminal de SimulIDE via RS232.

2. Expérience de travail à distance

Travailler à distance n'affecte pas beaucoup notre performance grâce aux différentes applications à notre disposition tel que Discord pour communiquer, SimulIDE pour tester et Visual Studio Live Share pour coder ensemble. On dirait même que l'accès à SimulIDE nous sauve des nuits blanches à Polytechnique. Effectivement, en temps régulier on testerait avec le vrai robot et avec les outils disponibles dans les salles informatiques qu'on ne peut pas apporter chez nous. De plus, le fait de travailler à distance épargne le long trajet pour aller à l'école pour certains membres de l'équipe.

Toutefois, même si les rencontres en présentiel entre les membres n'était pas possible, le travail à distance a tout de même apporté quelques avantages. Effectivement, il est beaucoup plus facile de se fixer des rencontres en ligne plutôt que d'organiser des rencontres en personne. Cependant, la cohésion en équipe se fait beaucoup plus difficilement dû à l'environnement de travail virtuel. En effet, le manque de rétroaction visuelle des coéquipiers fait en sorte que beaucoup d'information est perdue au niveau de la communication non verbale. Néanmoins, d'autres activités de cohésions étaient possibles grâce au travail à distance.