Polytechnique Montréal

Département de génie informatique et génie logiciel

Cours INF1995 :
Projet initial en génie informatique et travail en équipe

Travail pratique 8

Makefile et production de librairie statique

Par l'équipe

No 2330

Noms :

Tamer Arar
Guy Frank Essome Penda
Asmae El hajla
Maroua Djaroud

Date :
1 novembre 2016

Partie 1 : Description de la librairie

Notre librairie est composée de plusieurs classes qui représentent chacune une fonctionnalité matérielle ou logicielle. Chacune de ses classes contiennent des fonctions très utiles et permettent de réutiliser le code sans avoir à refaire l'implémentation de nouveau lorsqu'ils seront nécessaires pour notre projet final qui arrive à grand pas.

Classes :

1-Del

Cette classe a 2 fonctions :

A-couleurAmbree

Cette fonction sert à appeler la façon de faire la couleur ambrée à partir des 2 couleurs que peut prendre la DEL (vert ou rouge). Elle ne prend aucun paramètre et est de type *void*.

B-allumerDel

Cette fonction prend en paramètre un *char* soit *a*, *r*, ou *v* pour allumer la DEL en ambrée, en rouge ou en vert, respectivement. Par défaut, la DEL est éteinte. La sortie est sur PORTC. C'est une fonction qui ne prend aucun paramètre et est de type *void*.

2-Bouton

Cette classe contient une fonction :

A-isPressed

Cette fonction booléenne, qui prend aucun paramètre, retourne vrai si *PIND* & 0x04 est vrai. Ainsi, cela permet de savoir, par scrutation, quand le bouton-poussoir est appuyé. Bien sûr, l'effet de rebond est géré avec un petit délai (10ms).

3-Can

Cette classe représente un convertisseur analogique numérique capable de convertir un signal analogique (voltage) en données numériques. Elle est essentielle lorsque le système possède des capteurs dépourvus de capacités de communication numérique. C'est une classe qui nous a été donnée.

4-interruptions

Cette classe contient 2 fonctions :

A-initialisationBouton

Elle sert à initialiser les registres utiles pour une interruption par l'appui d'un bouton poussoir. En mettant cette fonction dans notre librairie, on évite de consulter régulièrement la documentation de Atmel. Elle est de type *void*.

B-partirMinuterie

C'est une fonction qui prend en paramètre une durée (entier non signé sur 16 bits) et qui sert à initialiser les registres permettant le début d'un compteur. Le compteur est arrêté lorsqu'il atteint la durée fixée. Il y a présence du mode CTC et l'horloge est divisée par 1024. Cette fonction est de type *void*. Elle nous évitera de toujours fouiller dans la documentation de Atmel.

5-Memoire

Cette classe regroupe les fonctions de lecture et d'écriture sur la mémoire de notre carte-mère. Elle nous a été fournie et elle sera utile pour notre projet final.

6-Moteur

Cette classe contient 5 fonctions :

A-initialiserMoteur

Cette fonction sert à définir les ports de sortie du moteur pour le signal PWM soit les ports 4 et 5 de la broche D. Les ports 6 et 7 de la broche D sont utilisés pour la direction. Elle initialise également les registres utiles à l'initialisation de moteur. En mettant cette fonction dans notre librairie, on évite de consulter régulièrement la documentation de Atmel.

B-changerVitesse

Cette fonction permet de varier la vitesse des roues en utilisant les deux variables passées en paramètres soit 2 entiers non signés sur 8bits et qui seront affectées aux registres OCR1A et OCR1B. Le PWM est utilisé dans cette fonction afin de faciliter le changement de vitesse. La fonction est de type *void*.

C-avancer

Cette fonction, qui prend aucun paramètre et qui ne retourne rien (*void*), permet de définir l'orientation du mouvement que les roues prendront lorsque celles-ci tourneront. Ici, le robot doit avancer.

D-reculer

La fonction reculer est semblable à la fonction avancer, c'est à dire, qu'elle ne prend aucun paramètre en compte et est de type *void*. Elle change aussi l'orientation du mouvement des roues. Par contre, le robot se déplacera vers l'arrière.

E-tourner

Cette fonction prend en paramètre un *char* soit *d* ou *g* pour modifier la direction des roues. Elle ne retourne rien. C'est une fonction qui utilise la méthode switch () case. Avec *d* comme état, les roues seront dirigées vers la droite, et avec *g* comme état, les roues se dirigeront vers la gauche.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Tout d'abord, avant de créer la librairie statique, il faut séparer le contenu du dossier de travail en deux dossiers différents. Dans le premier, lib_dir, on met tous nos fichiers de fonctions sélectionnée. La librairie sera générée à l'aide de ces fichiers. Il y aura alors ici le premier makefile à ce sujet. Dans le deuxième dossier, exec_dir, ça sera l'emplacement du fichier exécutable qu'on voudra créer en s'aidant des définitions de la librairie. Pour faire appel à cette librairie, on doit modifier le makefile donné.

Makefile de la librairie lib2330:

En premier lieu, on devait faire en sorte que le makefile arrive à lire tous les fichiers d'extension .cpp qui se retrouvent dans le même répertoire, à la place de toujours les spécifier manuellement :

ligne 31 PRJSRC= $(wildcard *.cpp)

Ensuite, on sait qu'après avoir compiler les fichiers cpp, on utilise les fichiers objets (.o), générés par cette compilation, pour créer la librairie statique. Ainsi, on fera appel à une commande ar et à 3 commandes options c, r et s :

Exemple : $ ar crs lib2330.a fichier1.o fichier2.o

Pour implémenter cette commande dans le makefile, on commence par ajouter à la section #variables# un archiveur, et, à la sections #Options de compilation#, les options utilisées :

ligne 73 AR=avr-ar
ligne 91 ARFLAGS=crs

Aussi, puisqu'une archive de librairie a une extension *.a*, on modifie celle du fichier de sortie :

ligne 98 TRG=$(PROJECTNAME).a

Pour finir, comme démontré plus haut, la commande de l'implémentation de la cible doit être :

```
ligne 137   $(TRG): $(OBJDEPS)
                $(AR) $(ARFLAGS) $(TRG) $(OBJDEPS)
```

Makefile de l'exécutable :

En deuxième lieu, l'utilisation de -L../lib_dir -llib2330 n'a pas fonctionner correctement. Ainsi, on a préféré inclure une inclusion additionnelle qui est notre répertoire de l'emplacement de la librairie :

ligne 34 INC=-I../lib_dir

De plus, on a fallu ajouter le chemin exact à notre fichier librairie lib2330.a :

ligne 37 LIBS=../lib_dir/lib2330.a

De cette manière, à la compilation, le compilateur ira chercher directement la définition de fonctions qu'on aura appelé dans notre fichier source par des includes. Ce fichier source doit être explicitement spécifié à la ligne 31 dans l'emplacement PRJSRC (Project Source).