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 375465

Noms:

Armand Hyeudip

Olivier Boivin

Camille Bergeron-Miron

Samuel Meyer

William Harvey

Mathieu Bélanger

Date:

14 mars 2017

Partie 1 : Description de la librairie

Décrire la librairie construite et formée (définitions, fonctions ou classes, utilité, etc...) pour que cette partie du travail soient bien documentées pour la suite du projet pour le bénéfice de tous les membres de l'équipe.

Notre librairie contient plusieurs fonctions d'aspect général qui peuvent être facilement réutilisables. On y compte des fonctions se rapportant à la verification du rebond mécanique, à la génération de PWM, d'initalisation des interruptions, d'initialisation de la minuterie et d'écriture sur les broches de la carte mère. On y inclut aussi l'implémentation de certaines méthodes propres à la classe PWM qui contrôle le PWM.

Fonction: bool verifierRebondMecanique()

Description: On appelle la fonction après qu'un signal du bouton-poussoir soit détecté. Elle s'assure que celui-ci (le signal) est encore present 10ms plus tard pour etre sur que le contact soit vrai.

Fonction : void ajustementPwmProcesseur(int temps, double pourcentage, double frequence)

Description: Prend en paramètre un pourcentage et génère un PWM qui est utilisé dans la fonction ajustementPWM. Le paramètre de fréquence permet d'initialiser la fréquence désirée du PWM. Le paramètre temps permet d'initialliser la durée du PWM.

Fonction: void ajustementPwmMicrocontrolleur(uint8_t pourcentageA, uint8_t pourcentageB)

Description: Permet de générer un PWM avec deux minuteries. Les paramètres pourcentageA et pourcentageB initialisent les pourcentages de temps actifs du PWM désirés pour les deux moteurs, par la modification des Output Compare Register 1 A et B. . Lorsque les valeurs des minuteries sont équivalentes à OCR1A et OCR1B respectivement, l'output est à 1.

Fonction: void initialisationINT0(bool modeBit0, bool modeBit1)

Description: Initialise l'interruption INT0. Le "sense control" de l'interruption est initialisé par les paramètres de la fonction. Le paramètre modeBit0 permet d'initialiser la valeur de ISC00 à 0 ou 1. Le paramètre modeBit1 permet d'initialiser la valeur de ISC01 à 0 ou 1.

Fonction: void initialisationINT1(bool modeBit0, bool modeBit1)

Description: Initialise l'interruption INT1. Le "sense control" de l'interruption est initialisé par les paramètres de la fonction. Le paramètre modeBit0 permet d'initialiser la valeur de ISC10 à 0 ou 1. Le paramètre modeBit1 permet d'initialiser la valeur de ISC11 à 0 ou 1.

Fonction: void initialisationINT2(bool modeBit0, bool modeBit1)

Description: Initialise l'interruption INT2. Le "sense control" de l'interruption est initialisé par les paramètres de la fonction. Le paramètre modeBit0 permet d'initialiser la valeur de ISC20 à 0 ou 1. Le paramètre modeBit1 permet d'initialiser la valeur de ISC21 à 0 ou 1.

Fonction: void minuterie(uint16_t duree)

Description: Permet d'initialiser une minuterie/compteur de 16 bits dont la durée est indiquée en paramètre (nombre sur 16 bits). La fréquence d'incrémentation de la minuterie a été divisé par 1024 par rapport à la fréquence du CPU.

Fonction: void ecrire1(char port, int broche)

Description: Fonction permettant mettre un 1 en sortie d'une certaine broche d'un certain port de la carte mère.

Fonction: void ecrire0(char port, int broche)

Description: Fonction permettant mettre un 0 en sortie d'une certaine broche d'un certain port de la carte mère.

Classe: PWM

Descrition: Classe permettant la gestion du PWM. Elle contient pour méthodes setFrequence qui permet d'ajuster la fréquence du PWM, setPourcentage qui permet d'ajuster le pourcentage décrivant le rapport a/b et setDirection qui permet d'activer le «mode reculons» du robot. Pour l'instant, cette classe n'est pas utilisée, car on ne connaît pas les requis de l'implémentation, toutefois elle sera implémenter plus tard lorsque les requis du projet se spécifieront.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Décrire les quelques modifications apportées au Makefile de la librairie pour démontrer votre compréhension de la formation des fichiers. Faire de même pour les modifications apportées au Makefile du code (bidon) de test qui utilise cette librairie.

Le rapport total ne doit par dépasser 7 pages incluant la page couverture.

Tout d'abord, afin de créer la librairie statique, il est nécessaire de produire les fichiers objets qui dépendent des fichiers cpp et h qui composent la librairie. Cela peut se traduire par la règle suivante:

```
%.o: %.cpp
   $(CC) $(CFLAGS) $(CXXFLAGS) -c $<
```

Ensuite, la cible $(PRJSOURCE), soit notre librairie statique .a, a comme dépendance l'ensemble des fichiers objets $(OBJDEPS). Dans l'instruction avr-ar crs, c permet de créer une archive et s place les fichiers objets dans cette archive.

Enfin, le Makefile d'un projet utilisant la librairie statique doit spécifier ses fichiers d'inclusions (include) lors de la création des fichiers objets. Ainsi, la variable $(INC) = -I./tp8 donne le répertoire de la librairie et LIBS= -L.tp8 -lA le nom de la librairie qui sont utilisés par la variable $(CFLAGS).

Afin de rendre plus léger les Makefiles, les variables communes au Makefile créant la librairie et celui produisant un exécutable qui utilise la librairie comme $(CFLAGS), MCU, OPTLEVEL ainsi que les commandes produisant les fichiers objets (%.o: %.cpp
$(CC) $(CFLAGS) $(CXXFLAGS) -c $<) sont rassemblés dans un fichier texte nommé Makfile_commun qui est inclut par les autres Makefile. Cela permet d'avoir un code léger contenant uniquement les commandes uniques à un projet.

Barème: vous serez jugé sur:

- *La qualités et le choix de vos portions de code choisies (5 points sur 20)*
- *La qualités de vos modifications aux Makefiles (5 points sur 20)*
- *Le rapport (7 points sur 20)*
- *Explications cohérentes par rapport au code retenu pour former la librairie (2 points)*
- *Explications cohérentes par rapport aux Makefiles modifiés (2 points)*
- *Explications claires avec un bon niveau de détails (2 points)*
- *Bon français (1 point)*
- *Bonne soumission de l'ensemble du code (compilation sans erreurs, …) et du rapport selon le format demandé (3 points sur 20)*