Polytechnique Montréal

Département de génie informatique et génie logiciel

Cours INF1900:
Projet initial de système embarqué

Travail pratique 7

Makefile et production de librairie statique

Par l'équipe

No 4041

Noms:
Laurence Robert
Hugo Palisson
Rania Abid
Semen Sukhov

Date:
28 février 2019

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 comporte 14 documents. Tous les documents .cpp qui sont présent dans celle-ci ont un document .h (avec la déclarations des fonctions) qui les accompagnent.

button_interrupt.cpp

La fonction initInterruptButton() présente dans le document button_interrupt.cpp permet une interruption lorsque le bouton poussoir est enfoncé. Nous avons décidé d'ajouter cette fonction à notre librairie, car appeler cette fonction nous évite d'aller chercher dans une documentation quel bit changer dans quel registre pour activer une interruption.

can.cpp

Ce document contient des méthodes permettant de créer un objet servant à convertir des signaux électriques analogiques en signaux numériques, ce qui permet de détecter une différence de voltage entre une valeur arbitraire et le voltage reçu par le port A. Cela peut être très utile pour utiliser des capteurs analogiques simples.

lumin_detect.cpp

Ce document contient la fonction brightnessDetection() : elle a besoin de can.h pour fonctionner car une de ses entrées est un objet de type "can". Cette fontion simplifie la lecture en donnant directement en sortie un uint8_t qui représente le rapport entre le voltage référence donné en entrée de la fonction et le voltage capté par le port A. Cela simplifie l'utilisation de capteurs analogiques simples.

memoire_24.cpp

Ce document contient des fonctions permettant d'écrire et de lire des données dans la mémoire interne de notre micro-controleur ou dans une mémoire externe.

phase_corr_pwm.cpp

Ce document contient la fonction setPhaseCorrectPWM() permettant de régler et d'activer deux signaux PWM et de les envoyer au port D. Les deux uint8_t pris en paramètre servent chacun à réguler la puissance du signal PWM qui leur correspond. Cela sera très certainement utile pour, entre autres, faire fonctionner les roues de notre robot.

transmission_uart.cpp

Ce document contient trois fonction : initialisationUART(), transmissionUART() et transmissionChaineUART(). La première sert à faire les manipulations nécessaires pour permettre le transfert de données vers l'ordinateur. La deuxième sert à transmettre un seul uint8_t (ou char) vers l'ordinateur. La troisième sert à envoyer une chaine complète de caractères (uint8_t* ou char*) vers l'ordinateur. Cela pourrait être très utile dans le future pour des opérations de débogage.

lib4041.h

Nous avons fait un document d'entête lib4041.h qui contient tous les autres documents d'entête de notre librairie. Nous les avons tous inclus au début de lib4041.h pour permettre d'inclure seulement lib4041.h au début de nos projets si ces derniers utilise beaucoup de documents de notre librairie, au lieu d'inclure plein de documents indépendants.

Makefile

Le makefile sert à recompiler la librairie dans le cas où des documents seraient modifiés ou ajoutés.

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.

Makefile de la librairie

Voici une liste des modifications effectuées par rapport au Makefile de base :

1. Au lieu de lister manuellement le nom de chaque fichier à compiler, nous avons mis "PRJSRC=$(wildcard *.cpp)". Cela permet de compiler tout les fichiers .cpp présents dans le dossier courant.

2. Nous avons ajouté une variable CL=ar crs. Cette dernière remplace la variable CC lors de l'implémentation de la cible, ce qui compile un fichier .a plutôt qu'un .out.

3. Nous avons changé la valeur de la variable TRG pour "$(PROJECTNAME).a" pour clarifier le fait que nous compilons une librairie.

Makefile du code de test

Voici une liste des modifications effectuées par rapport au Makefile de base :

1. Au lieu de lister manuellement le nom de chaque fichier à compiler, nous avons mis "PRJSRC=$(wildcard *.cpp)". Cela permet de compiler tout les fichiers .cpp présents dans le dossier courant.

2. Nous avons ajouté une variable LIB_DIR=./lib4041 qui contient l'adresse du dossier de la librairie.

3. Nous avons ajouté une variable LIB_NAME=4041 qui contient le nom de la librairie.

4. Nous avons ajouté "-L $(LIB_DIR) -l $(LIB_NAME)" à la variable LIBS, ce qui donne les informations nécessaires à l'implémentation de la cible pour permettre d'y lier notre librairie.

5. Nous avons ajouté "-I $(LIB_DIR)" à la convertion de .cpp à .o, ce qui permet l'utilisation du code requis contenu dans le dossier de la librairie.

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

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)*