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 68116

Noms :

Bourque Bédard Christophe
Ouaissa Fares
Moreau Simon
Saddik Mohamed

Date :
12 Mars 2018

Partie 1 : Description de la librairie

Après comparaison du code fait par nos deux équipes, nous avons décidé de garder quelques fonctions que nous jugeons nécessaires pour notre librairie utilisée pour le projet final.

Pour le TP2, nous avons décidé de garder les fonctions de délais en microsecondes et millisecondes. Au niveau du TP5, nous avons décidé de prendre la totalité des fonctions : contrôle par interruption, interruption de la minuterie « timer », et le PMW matériel.

Au niveau de TP6, nous avons décidé de garder la classe mémoire24cxxx et ses fonctions, et finalement pour le TP7 nous avons gardé la classe can pour la conversion des données reçues par capteurs.

Afin de structurer nos fichiers, nous avons décidé de séparer les définitions de nos fonctions et nos classes dans un dossier « include » et nos implémentations dans le dossier src.

Voici ci-dessous une description de chacune des fonctions :

- delai.cpp contient deux fonctions WaitForUs(uint16_t us), WaitForMs(unint16_t ms). Lors de nos TPs nous avons remarqué l'utilité de ces fonctions, surtout pour les DELs sur la carte mère. Elles seront d'une grande utilité si le mandat de notre projet final nous impose d'allumer une DEL ou de tout autre délai qu'il ne serait pas nécessaire de faire avec la minuterie si le microcontrôleur n'est pas trop occupé pendant une partie du programme.

- pwm.cpp contient les fonctions initPWM() et ajustementPWN(const uint8_t& pourcentage). Nous allons sans doute utiliser les notions de PWM pour faire avancer, reculer, arrêter, et tourner notre robot. Il nous suffira d'appeler cette fonction et de l'adapter au parcours voulu lors du mandat.

- minuterie.cpp contient l'interruption ISR(TIMER1_COMPA_vect) et les fonctions initMinuterie(func_t func), startMinuterie(const uint16_t duree), stopMinuterie(). Nous avons jugé que l'utilisation des interruptions sera une partie indispensable lors de l'épreuve finale, où les délais ne pourront pas toujours être faits avec les fonctions précédentes par souci d'efficacité du programme.

- memoire_24.cpp contient init(), choisir_banc(const uint8_t banc), lecture(const uint16_t *donnee), lecture (const uint16_t adress, uint8_t *donnee, uint8_t longueur), ecriture(const uint16_t adress, const uint8_t donnee), ecriture(const uint16_t adresse, unit8_t * donnee, const uint8_t longueur), ecrire_page(const uint16_t adresse, unit8_t* donnee, const uint8_t longueur). Nous l'avons inclus dans notre librairie, puisque la communication RS232 est nécessaire pour les valeurs inscrites en mémoire vers l'ordinateur.

- can.cpp est une classe fournie, nous l'avons incluse puisque la conversion analogique numérique est fondamentale pour le parcours de notre robot.

- uart.cpp qui contient initialisationUART(), transmissionUART(const uint8_t& donnee) qui sont utiles pour la communication RS232, permettant de transmettre les valeurs inscrites en mémoire vers l'ordinateur.

- interruption.cpp qui contient l'interruption ISR(INT0_vect) et la fonction initInterruption(func_t func, const TypesTriggerInterrupt type) qui permettent de gérer les interruptions par interrupteur. Ceux-ci pourraient être utiles pour démarrer le programme ou pour toute autre utilisation de l'interrupteur.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Le makefile_common est essentiellement le makefile qui nous a été fourni en début de session, avec les modifications nécessaires aux besoins de ce TP, alors que le makefile qui fait la librairie et le makefile qui crée le projet ont été écrits pour le TP. L'utilisation de la commande echo dans le code est uniquement pour faciliter le débogage du code.

➢ MAKEFILE COMMON

SRCDIR= src le répertoire contenant les fichiers sources .cpp
PRJSRC= $(filter %.cpp, shell echo $(SRCDIR)/*) tous les fichiers avec l'extension .cpp dans le répertoire source
AR= avr-ar variable pour l'appel de la commande ar qui peut créer l'archive

On modifie aussi la variable CXXFLAGS pour inclure une option spécifiant la fréquence du microcontrôleur ainsi que la version de C++ utilisée. On exclut le nom des cibles par défaut qui ne sera pas utilisé dans le makefile_common, mais plutôt dans les makefiles spécifiques auquel il fait appel. La modification sur OBJDEP permet de faire la liste de tous les fichiers .o à partir des fichiers .cpp, patsubst remplaçant /$(SRCDIR)/%.cpp par /$(BUILDDIR)/%.o. Les règles de construction de chaque fichier objets et hex dans le répertoire build à partir des fichiers .cpp trouvés dans le répertoire source sont les mêmes, seules les dépendances sont modifiées. On a aussi ajouté la règle de construction du répertoire build qui est ensuite incluse dans la construction des fichiers objets. La commande clean est modifiée pour effacer le répertoire build, les fichiers de librairie, les fichiers objets et les fichiers de dépendance en plus de la commande déjà présente dans le makefile fourni. Cela simplement parce qu'ils ne sont pas inclus dans la commande de base.

➢ MAKEFILE LIB

Nous l'avons commencé par une précision du nom du fichier et par une définition des différents répertoires :

LIBNAME= librobot le nom de la librairie
INCDIR= include le répertoire pour des fichiers entêtes .h
BUILDDIR = build le répertoire pour les fichiers et les fichiers objets
INC = -I $(INCDIR) pour les inclusions additionnelles
TRG = $(BUILDDIR) / $ (LIBNAME).a la cible qui sera construite
Include ../makefile_common inclusion du makefile commun qui nous permet d'utiliser les variables définies dans ce fichier

Pour la règle de construction de la cible, on utilise avr-ar avec les options c, r et s pour la création de la librairie plutôt que la commande avr-gcc.

➢ MAKEFILE PROJET

Ce makefile contient d'abord la même chose que le makefile de la librairie, avec quelques modifications :

INC = -I ../lib/$(INCDIR)/	Inclusion du répertoire lib
LIBS=-lrobot -L ../lib/$(BUILDDIR)/ -L ../lib/src	L'emplacement des fichiers de la librairie liée

Dans la règle de construction de la cible :

-lm	
$(LIBS)	La librairie nécessaire au makefile