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 95102

Noms:

Nicole Joyal
Yuri Azar
Emilio Gagnon
William Valiquette
Pascale Sylvestre
Jérôme Désilets

Date:
12 mars 2017

Partie 1 : Description de la librairie

Considérant les demandes du projet, nous avons fait nos choix de fonctions et classes par rapport à leur utilité à faire exécuter un parcours au robot.

Dans cette optique, nous avons tout d'abord inclus les codes reliés aux moteurs du robot, soit notre classe *pwm*. En effet, il va nous être primordial de pouvoir non seulement activer et éteindre les moteurs à notre guise, mais également moduler leur vitesse ainsi que changer la direction.

Ensuite, il va nous être utile de pouvoir écrire dans les registres du robot, ainsi que lire dans ceux-ci. Dans ce but, nous avons inclus la classe *communicationUART*, qui permet de communiquer avec les registres. Cette classe est fortement liée à *memoire_24,* qui elle, permet de lire et écrire dans les registres. Nous l'avons donc également incluse.

Pour continuer, nous avons jugé l'utilisation de la DEL sur le robot comme pouvant être nécessaire dans le futur. Nous avons donc inclus tous les fichiers nécessaires à son utilisation minimale, en commençant par le fichier *del* qui contient toutes les méthodes permettant de modifier l'état de celle-ci. Il a fallu évidemment rajouter le fichier *can*, permettant de lire une entrée analogique et la convertir en numérique, qui nous est essentiel lorsque nous travaillons avec un capteur, ce qui sera nécessaire pour exécuter le parcours durant l'épreuve finale. Nous avons également ajouté la classe *delais*, qui permet de créer un délai variable à l'aide de la fonction *_delay_ms().* En effet, s'il est question de minuteries, cette classe nous aidera grandement à faire des minuteries précises en utilisant des variables.

Finalement, nous avons inclus le fichier *constante* pour définir plusieurs constantes au même endroit qui seront utilisées dans tous les autres fichiers.

1. del

Ce fichier contient des fonctions pour contrôler la diode électroluminescente.

- *class del*
  - **eteindre()**
  - **allumerRouge()**
  - **allumerVert()**
  - **allumerAmbre()**

2. can

Contient la fonction servant à lire l'état du capteur.

- *class can*
  - **lecture()**
    Cette méthode fait la conversion analogique-numérique afin de pouvoir analyser les données reçues. Ceci facilite le traitement des données environnementales (luminosité) à l'aide d'opérations arithmétiques et logiques.

3. memoire_24

Contient les méthodes associées à la lecture et à l'écriture en mémoire

- *class Memoire24CXXX*
  - **choisir_banc()**
    Permet de choisir une plage de mémoire à traiter.

  - **lecture()**
    Lire à partir d'une adresse, soit une donnée soit plusieurs données (faut passer une longueur dans le deuxième cas)

  - **ecriture()**
    Écrire à partir d'une adresse, soit une donnée soit plusieurs données (faut passer une longueur dans le deuxième cas)

  - **ecrire_page()**
    Méthode privée, permet d'écrire un bloc de données (en tenant compte des limites de grandeur d'une page)

4. constantes

Ce fichier contient les constantes utilisées dans le projet.

- Définitions de :
  - *F_CPU*
  - *SORTIE*
  - *ENTREE*
  - *ROUGE*
  - *VERTE*
  - *ETEINT*
  - *AVANCER*
  - *RECULER*

5. pwm

Cette classe permet de contrôler la vitesse du moteur et sa direction.

- *class pwm*
  - **arretMoteur()**
    Arrête les moteurs en mettant les registres à zéro.

  - **AjustementPWM(uint8_t droite, uint8_t gauche, uint8_t directionGauche, uint8_t directionDroite)**
    Sert à ajuster la puissance de chaque moteur, donc permet de faire tourner le robot, de contrôlé sa vitesse, sa direction et son sens.

6. delais

Ce fichier sert à introduire des délais variables.

- *class delais*
  - **delaisVariable(int n)**
    Permet de faire une fonction délais qui prend en paramètre une variable.

7. communicationUART

Cette classe sert à communiquer avec les registres du robot.

- *class communicationUART*
  - **initialisationUART()**
    Initialise les registres pour permettre la transmission et la réception du UART

  - **TransmissionUART(uint8_t donnee)**
    Transmet les données entrées en paramètre dans les registres du robot.

Partie 2 : Décrire les modifications apportées au Makefile de départ

Pour le tp8, nous avons utilisé trois *makefiles* : un *makefile* commun qui comprend les variables communes des *makefiles*, un *makefile* pour créer la librairie et un *makefile* pour exécuter le programme *main* servant de test.

Dans le *makefile* commun nous avons inclus les variables communes comme le nom du microcontrôleur, le compilateur, la fréquence du CPU et les f*lags* de compilation. Nous avons conservé les valeurs des variables du fichier *makefile* donné en exemple.

Dans le *makefile* de la librairie, nous avons tout d'abord inclus le *makefile* commun. Ensuite, nous avons modifié la variable *PROJECTNAME* à *libProjet*. Nous avons aussi ajouté tous les fichiers sources faisant partie de la libraire, soit *can.cpp, communicationUART.cpp, del.cpp, delais.cpp, mémoire_24.cpp* et *pwm.cpp.* Nous n'avons pas ajusté ni les inclusions ni les librairies liées, car notre librairie ne dépend pas d'autres librairies non-standardisées.

Afin de produire une librairie statique (fichier.a) nous avons ajouté une variable *AR* qui correspond au compilateur d'archive *avr-ar* et une variable *OPTION_AR* qui correspond aux options utilisées, soit *crs* (*c* pour créer l'archive, *r* pour insérer les fichiers membres et *s* pour produire l'index des fichiers objets). Nous avons aussi effacé les option *hex* et *install,* car la librairie n'a pas à être installée sur le robot.

Dans le *makefile* servant à exécuter le code, nous avons aussi inclus le *makefile* commun afin de déclarer les variables communes. Ensuite, nous avons modifié la variable *PROJECTNAME* à *TP8*, car c'est le *makefile* qui permet d'exécuter le tp8. Comme fichier source, nous utilisons seulement le fichier *main.cpp*. Nous n'avons pas ajouté d'inclusions, mais nous avons ajouté la librairie que nous venons de créer (*libProject.a*) à la variable *LIBS*. Pour la suite du *makefile*, nous avons conservé celui fourni en exemple en nous assurant que l'option *-lm $(LIBS)* était bien incluse dans la commande permettant l'implémentation de la cible.