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 204-208

Noms:

Nicolas Deloumeau
Laurent Langlois
Thomas Duperron
Hisham Boulifa

Date:
17 mars 2021

Partie 1 : Description de la librairie

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

Notre librairie (contenue dans le dossier *librairie*) est composée de 6 fichiers .h, 6 fichiers .cpp et d'un Makefile (permettant de créer un exécutable
). Ces derniers sont nécessaires au bon fonctionnement du robot et surtout à ce que nous désirons que ce robot exécute. Cette librairie contient donc "les fonctionnalités" de notre robot. Elle est composée de :

Bouton

La librairie Bouton permet de lire l'état d'une pin. De plus, un mécanisme d'anti-rebond est implémenté. La classe Bouton est initialisée avec les informations sur le port et la pin. Ensuite, une méthode *lire()* permet d'obtenir l'état de la pin. Cette méthode ajoute un délai de 10 ms. Aussi, une méthode *click()* permet de comparer l'état précédent de l'état actuel de bouton. Cette méthode retourne 1 lorsqu'il y a un front-montant. Cette méthode ajoute aussi un délai de 10 ms. Cette librairie permet de simplifier le code, car l'utilisation des boutons est courante.

Can

La librairie Can permet de lire une valeur analogique et la convertir en valeur numérique. Un constructeur permet d'initialiser les registres. Aussi, la méthode lecture() permet d'obtenir la valeur de la pin sous forme d'un entier de 16 bits.

DEL

La librairie DEL permet le contrôle simple d'une DEL. Une classe est chargée de stocker les informations sur le port et la pin. Le port et la pin ne sont spécifiés que lors de l'initialisation. Ensuite, il y a 3 méthodes pour modifier l'état de la DEL. Il y a les méthodes *allumer()* et *eteindre()* qui permettent de faire les opérations simples sur la DEL. Aussi, une méthode *inverser()* permet de changer l'état de la DEL en fonction de son état précédent. Cette librairie est intéressante car l'utilisation de DEL est utile dans beaucoup de projets pour afficher ou débugger de simples informations.

Minuterie

Le rôle de cette minuterie est de contrôler le *timer1*, qui est nécessaire au fonctionnement de la prochaine librairie (moteur). Elle doit donc faire partie de notre projet, car elle assiste une des composantes principales qu'est le moteur. D'abord, la classe *Minuterie* initialise le contenu des registres, puis la méthode *partirMinuterie()* permet, comme son nom l'indique, de la lancer correctement. Les registres sont

implémentés selon la documentation *Atmel* et OCR1A est chargé de la durée que couvre l'intervalle de la minuterie.

Moteur

Cette librairie est chargée du contrôle des moteurs. Nous avons décidé de choisir cette librairie, car elle est fondamentale dans la conceptualisation du robot. En effet, les moteurs sont une des composantes à absolument intégrer dans notre projet final, auquel cas le produit fini ne pourrait tout simplement pas se déplacer. Pour ce faire, on génère un signal PWM sur le port D en utilisant le timer1 du ATmega324PA. Ici, le signal sort par le port D5 et nous avons choisi un port voisin pour la direction, D6. Cette approche permet de rendre les moteurs autonomes, puisque le microcontrôleur ne sera pas nécessaire. La classe *Moteur* stocke les informations d'initialisation des différents registres. De plus, la méthode *ajustementMoteur()* permet de régler la vitesse ainsi que la direction du moteur. Enfin, le destructeur à la fin est implémenté pour arrêter le signal PWM.

UART

La librairie UART permet de transmettre des informations via le module UART. Un constructeur permet d'initialiser les registres. Aussi, une méthode *transmissionUART()* permet d'envoyer un byte. La méthode *transmissionPhrase()* permet d'envoyer une chaîne de caractères en faisant une boucle avec la méthode précédente. Ces méthodes sont bloquantes et se terminent seulement lorsque l'information a été transmise au complet. Cette librairie permet de simplifier le débogage en permettant d'envoyer du texte et des informations via le câble usb.

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.*

Les modifications apportées au Makefile de la librairie sont :

- **PROJECTNAME=liba** : Précise le nom de la librairie.
- **PRJSRC=$(wildcard *.cpp)** : Permet d'inclure tous les fichiers .cpp en source.
- **AR=avr-ar** : Nous avons ajouté au Makefile un créateur de librairie statique (AR pour archive).
- **ARFLAGS=-crs** : Flags pour créer la librairie statique qui remplace Linker.
- **TRG=$(PROJECTNAME).a** : Nous avons remplacé ".elf" par ".a", ce qui permet de spécifier en cible qu'il s'agit ici d'une librairie (statique).
- **all: $(TRG)** : Suppression de $(HEXROMTRG) car nous n'en avons plus besoin pour générer une librairie.

- Dans l'implémentation de la cible : Nous avons remplacé $(CC) $(LDFLAGS), permettant d'indiquer au compilateur quand il doit faire appel au linker, par $(AR) $(ARFLAGS) afin de pouvoir créer une archive à partir de tous de les fichiers objet et nous avons supprimé de la commande "-lm $(LIBS)" puisque nous n'avons pas besoin d'ajouter une autre librairie dans cette librairie.
- Nous avons aussi retiré la commande "make install" car elle n'est pas utile dans notre librairie.

Les modifications apportées au Makefile du code de test utilisant la librairie sont :

- **PROJECTNAME=tp7** : Précise le nom du projet.
- **PRJSRC= main.cpp** : Permet d'inclure le fichier "main.cpp" en tant que fichier source.
- **INC=-I ../../librairie** : Indique les inclusions additionnelles en précisant le chemin de la librairie à inclure.
- **LIBS=-L ../../librairie -la** : Indique la liaison des librairies en précisant le chemin de la librairie à lier.
- Nous avons enlevé la commande "make install" car elle n'est pas utile dans ce cas.

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

Barème: vous serez jugé sur:

- *- La qualité et le choix de vos portions de code choisies (5 points sur 20)*
- *- La qualité 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)*