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 1217

Noms:
Andrew Abdo
Wail Ayad
Youva Boutora
Aymeric Labrie

Date:
21 octobre 2019

Partie 1 : Description de la librairie

La librairie est formée de plusieurs classes, elles contiennent des fonctions qui permettent de contrôler les parties du robot. Voici un aperçu des différentes classes qui forment la librairie :

Common

Source : commun.cpp, commun.h
Objectif : Contenir des fonctions qui sont souvent répétées dans les autres fichiers sources de la librairie.
Namespace commun :

- void delai_ms(const uint16_t duree_ms):
  Fonction constituée d'une boucle pour produire un délai de la durée désirée.

Interrupt

Source : interrupt.cpp, interrupt.h
Objectif : Permettre la gestion des interruptions externes offertes par le ATmega324p qui sont INT0, INT1 et INT2.
Attributs :

- ISCn1, ISCn0
- PX
- PINX

Méthodes :

- void initINTX(uint8_t id) :
  Initialise les interruptions pour un PIN dépendamment de la valeur de id.
  Si id = 0, la PIN D2 est initialisée
  Si id = 1, la PIN D3 est initialisée
  Si id = 2, la PIN B2 est initialisée
- void initEICRA(TYPE_CONTROLE type) :
  Initialise le registre EICRA, qui permet de décider quel signal génère une interruption en fonction de la valeur de type.
  Si type = FALLING_EDGE, un front descendant déclenche une interruption.
  Si type = RISING_EDGE, un front montant déclenche une interruption.
  Si type = ANY_EDGE, un front montant ou un front descendant déclenche une interruption
  Si type = LOW_LEVEL, une interruption est générée lorsque INTX est actif bas
- Interrupt(uint8_t id, TYPE_CONTROLE type) :
  Construit les objets de la classe Interrupt.
- bool isButtonPressed() :
  Vérifie si le bouton est appuyé.

DEL

La DEL définit l'énumération globale COULEUR servant à modifier la couleur (VERTE, ROUGE, OFF) de la DEL.
Source : del.cpp, del.h
Objectif : Allumer la DEL branchée au port voulue au port 0 et 1 par défaut
Attributs :

- volatile uint8_t* portx_

- volatile uint8_t* ddrx_
- uint8_t PinMSB_
  Le numéro de pin à utiliser sur le portx (ayant Most Significant Bit)
- uint8_t PinLSB_
  Le numéro de pin à utiliser sur le portx (ayant Least Significant Bit)

Méthodes:

- DEL():
  Constructeur par défaut qui utilise le PORTB (PB0 et PB1) pour la DEL.
- DEL(volatile uint8_t* portx, volatile uint8_t* ddrx):
  Constucteur avec le choix de PORT la DEL.
- void allumerDEL(const COULEUR couleur) :
  Cette fonction modifie la couleur de la DEL.
- void allumerDEL (const COULEUR couleur, uint16_t duree_ms) :
  Cette fonction modifie la couleur de la DEL pour une durée spécifiée et ensuite l'éteint.

Memoire_24

Source : memoire_24.cpp, memoire_24.h
Objectif: gérer les opérations mémoire.
Attributs :

- static uint8_t m_adresse_peripherique
- const uint8_t PAGE_SIZE

Méthodes:

- Memoire24CXXX() :
  Constructeur de mémoire qui appelle init().
- ~Memoire24CXXX() :
  Destructeur.
- void init() :
  initialise à 0 le "memory bank"
- static uint8_t choisir_banc() :
  Cette méthode est appelée seulement si l'adresse doit changer.
- uint8_t lecture(const uint16_t adresse, uint8_t *donnee):
  uint8_t lecture(const uint16_t adresse, uint8_t *donnee, const uint8_t longueur):
  Une valeur à la fois ou bloc de données (longueur de 127 ou moins)
- uint8_t ecriture(const uint16_t adresse, uint8_t *donnee):
  uint8_t ecriture(const uint16_t adresse, uint8_t *donnee, const uint8_t longueur):
  Une valeur à la fois ou bloc de données (longueur de 127 ou moins)

Minuterie

Source : minuterie.cpp, minuterie.h
Objectif : Exécuter la minuterie (jusqu'à interruption) en fonction du TIMER que l'on doit utiliser.
Méthodes :
Toujours en mode CTC et prescaler de 1024 en utilisant les registres TCNTn, OCRnA, TCCRnB et TIMSKn.

- void partirMinuterie0(const uint16_t dureeA) :
- void arreterMinuterie0():
- void partirMinuterie1(const uint16_t dureeA) :
- void arreterMinuterie1():

- void partirMinuterie2(const uint16_t dureeA) :
- void arreterMinuterie2():

Moteur

Source : moteur.cpp, moteur.h
Objectif : Gérer les déplacements du robot.
Méthodes :

- Moteur() :
  Constructeur qui initialise le DDRD en mode sortie pour PD4 à PD7 pour contrôler les roues.
- void avancer(const uint8_t vitesse):
  void avancer(const uint8_t vitesse, const uint16_t duree_ms):
  Avance le robot soit perpétuellement ou pour une durée spécifiée.
- void reculer(const uint8_t vitesse):
  void reculer (const uint8_t vitesse, const uint16_t duree_ms):
  Recule le robot soit perpétuellement ou pour une durée spécifiée.
- void arrêter() :
  Immobilise le robot.
- void tournerDroite() :
  void trounerGauche():
  Tourne le robot dans une direction d'une manière prédéterminée.
- commandManuel() :
  Sert à contrôler les roues de façon plus précise.
- void ajusterPWM(const uint8_t roueG, const uint8_t roueD) :
  Méthode privée qui est utilisée pour changer la vitesse des roues.

UART

Source : uart.cpp, uart.h
Objectif: réception et transmission des données entre le ATmega324PA et l'ordinateur (par le UART0).
Méthodes :

- UART() :
  Constructeur appelant la méthode privée initUART().
- void transmissionUART(uint8_t donnee):
  Sert à la transmission de données.
- unsigned char receptionUART():
  Sert à la réception de données.
- void initUART() :
  Modifie les registres pour utiliser le RS32 avec le robot.

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 :

1. PRJSRC a été modifié à $(wildcard *.cpp) $(wildcard *.c) pour ajouter tous les fichiers .cpp et .c présents dans le répertoire.
2. La variable AR a été ajoutée, elle correspond à l'archiveur qui est dans ce cas avr-ar.
3. La variable ARFLAGS a été ajoutée, elle correspond aux options d'archivage, -r -c -s dans ce cas.
4. La variable TRGEXT a été crée pour décider l'extension de la variable TRG, la variable TRG a été modifiée à TRG = $(PROJECTNAME).$(TRGEXT), et TRGEXT a été initalisée à a pour créer un fichier .a.
5. L'instruction install a été supprimée parcequ'il ne fait pas de sens d'installer une librairie.
6. La recette pour créer TRG a été modifiée, $CC devient $AR vu qu'on veut archiver des fichiers objets, pas les compiler en un exécutable, $CFLAGS $CXXFLAGS devient $ARFLAGS pour la même raison. $OBJDEPS est quand même nécessaire pour créer les fichiers .o.
7. PROJECTNAME a été modifié pour librob.a.

Makefile du code :

1. La variable LIBDIR a été ajoutée, c'est le répertoire dans lequel se trouvent la librairie et les headers. On a décidé de mettre le code source de la librairie et les headers dans le même répertoire.
2. La variable INC a été modifiée à -I $(LIBDIR), ce qui dit au compilateur de chercher des fichiers headers dans LIBDIR
3. La variable LIB a été modifié à -L $(LIBDIR) -lrob, ce qui dit au compilateur de chercher une librairie dans le répertoire LIBDIR qui s'appelle “librob.a“.
4. L'instruction update a été ajoutée, l'instruction est cd $(LIBDIR) && make, l'instruction consiste à aller dans le répertoire LIBDIR et (si le répertoire existe) exécuter la commande make, ce qui permet de mettre à jour la librairie si elle a été modifiée.

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)