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 5367

Noms:

Hee-Ming Yang
Adam Halim
Chun Yang Li
Jean Janssen

Date:
3 novembre 2020

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 est composée de quatres classes de base, soit can, Led, Motor et Uart. Chacune de ces classes réutilisent des fonctions prélevées des tps précédents. Parfois, les méthodes sont modifiées pour permettre un usage plus flexible.

can

Cette classe contient un constructeur, un destructeur et une fonction "lecture". La fonction "lecture" sert à convertir les signaux analogiques (en volt) en valeurs numériques dans une variable à 16 bits où seuls les 10 premiers bits sont significatifs. Cette classe nous a été fournie lors du tp6, donc rien n'est à modifier pour l'implémenter en librairie.

Led

Cette classe contient une fonction "blink" et une fonction "buttonIsPressed". Cette classe regroupe le code qui sont en lien avec les manipulations de la DEL ainsi que le bouton-poussoir.

La fonction "blink" est un code simple qui sert à allumer une DEL pour un certain temps. Elle prend en paramètres:

- uint8_t ledPinSource: pin qui génère le courant.
- uint8_t ledPinGround: pin qui reçoit le courant.
- uint8_t duration: durée pour laquelle la DEL sera allumée
- volatile uint8_t& port: port connecté à la DEL sur l'Atmega.

Comparé au code original, nous avons ajouté un paramètre "port". Ce paramètre est dû au fait que le "blink" original ne fonctionnait que sur le port A de l'Atmega. Pour l'implémenter en librairie, il faut le rendre moins spécifique (ou plus flexible). En ajoutant ce paramètre, on permet alors à l'utilisateur de choisir le port voulu. Les paramètres ledPinSource et ledPinGround sont aussi ajoutés pour permettre à l'utilisateur de choisir les pins à utiliser du port ainsi que de changer la couleur du DEL (selon le sens du courant). De plus les opérateurs d'affectations (=) pour assigner les ports (ex. PORTA = 1 << PORTA1) sont désormais des opérateurs de logique OR et AND avec complément (pour éteindre) sur les pins spécifiques à la méthode pour préserver la valeur des autres pins non reliés.

La fonction "buttonIsPressed" vérifie si le bouton est pesé avec un anti-rebond. Elle retourne un true ou false selon si c'est pesé ou non. Elle prend en paramètres:

- uint8_t button: pin lié au bouton à peser.
- volatile uint8_t port: port configuré en entrée qui est relié au bouton.

Similairement à la fonction "blink", cette fonction contient un paramètre volatile pour permettre à l'utilisateur de choisir son port en entrée. De plus, elle a une variable locale "DEBOUNCE_DELAY" qui contrôle le délai entre les 2 vérifications de l'anti-rebond. La fonction originale utilise une variable globale à la place.

Motor

Cette classe contient une fonction "turnMotorPWM", une fonction "convertPercentInPWM8BitTimer" et une fonction "adjustPWM". Cette classe regroupe les codes qui sont en lien avec l'ajustement de la vitesse des roues par le biais de PWM sur 8 bits.

La fonction "adjustPWM" sert à ajuster le PWM. Elle prend en paramètres:

- uint8_t percentage: pourcentage du PWM.
- volatile uint8_t ocrnx: pin responsable de générer l'onde PWM.

La fonction originale ajuste le PWM pour 2 pins fixes et comporte aussi l'initialisation du timer très spécifique au tp. L'initialisation est enlevée de la fonction en librairie pour le rendre plus flexible et permettre à l'utilisateur d'initialiser son propre timer. La fonction ne peut désormais qu'ajuster un PWM à la fois, puisque le nombre de PWM à ajuster dépend de la situation et avec l'ajout du paramètre "ocrnx", l'utilisateur peut choisir son pin sur l'Atmega.

La fonction "convertPercentInPWM8BitTimer" permet de prendre un pourcentage et de retourner une valeur équivalente sur 8 bits soit entre 0 et 254. Cette fonction prend en paramètre:

- uint8_t percentage: permet de choisir le pourcentage à convertir.

Cette fonction est déjà très peu spécifique, donc aucune modification n'est nécessaire.

La fonction "turnMotorPWM" permet de faire tourner une roue pour un certain intervalle de temps. Elle prend en paramètres:

- double PWM: pourcentage de PWM voulu.
- double frequency: fréquence à laquelle la roue va tourner.
- uint8_t enablePin: pin enable relié à la roue sur l'Atmega.
- uint8_t directionPin: pin direction relié à la roue sur l'Atmega.
- double duration: durée où la roue va tourner.
- volatile uint8_t port: port reliée à la roue sur l'Atmega.

Le code original a une roue fixe sur un port spécifique et cette roue tourne pendant un intervalle de temps qui est aussi spécifique. Dans la nouvelle fonction, les paramètres "duration", "enablePin", "port" et “directionPin” sont ajoutés pour avoir plus de flexibilité en permettant à l'utilisateur de choisir le temps pendant lequel la roue va tourner. Le pin enable ainsi que le port déterminent où l'utilisateur veut relier la roue. Finalement, le “directionPin’’ sert à déterminer la direction que tourne la roue. De plus, les opérateurs

d'affectations sont changés en opérateur de logique OR sur les pins spécifiques à la méthode pour préserver la valeur des autres pins non reliés.

Uart

Cette classe contient un constructeur par défaut, une fonction "initialisation" ainsi qu'une fonction "transmission". Cette classe regroupe les codes qui sont en lien avec la transmission des données avec le protocole RS232 vers le PC.

Le constructeur utilise la fonction initialisation.

La fonction "initialisation" active les registres qui permettent à l'Uart de transmettre et de recevoir les données. Cette fonction n'a pas été modifiée.

La fonction "transmission" permet de transmettre des données vers le PC. Elle prend en paramètre:

- uint8_t data: donnée ASCII à transmettre.

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.

<u>Le rapport total ne doit par dépasser 7 pages incluant la page couverture.</u>

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)*