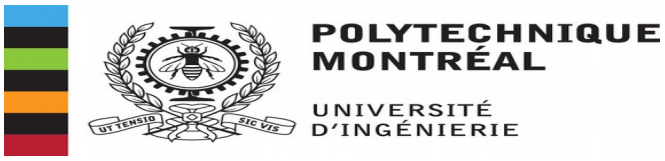

Département de génie informatique et génie logiciel

INF1900
Projet initial de système embarqué

Rapport final de projet

Projet dans SimulIDE

Équipe No <5773>

Section de laboratoire <2>

Kheloufi Abdelaziz
Oudada Marouane
Bourai Sami
Mounib Djellab

8 décembre 2020

1. Description de la structure du code et de son fonctionnement

Cette session, exceptionnellement, le projet final consistait à concevoir un panneau de contrôle plutôt qu'un robot comme les années précédentes. Le panneau de contrôle qu'on devait réaliser avait pour but de simuler un système pouvant contrôler plusieurs dispositifs que ce soit de manière manuelle ou préprogrammée. En effet, les dispositifs à programmer sont les suivants : 4 portes LED, 2 servomoteurs, un clavier, un sonar, une horloge, un écran LCD et un bouton poussoir. En résumé, le système possède 6 options qu'on peut sélectionner à l'aide du clavier. Dans les prochains paragraphes, nous allons montrer la structure et le fonctionnement de notre code. Chaque objet utilisé comporte des méthodes qui lui sont utiles et se trouvent dans les fichier.h et les fichier.cpp, le tout inclus dans notre librairie.

- **Sonar :** Ces méthodes nous permettent de contrôler le sonar et se trouvent dans les fichiers: sonar.h et sonar.cpp

activerTrigger10ms() : Cette fonction nous permet d'envoyer un signal en mettant la pin PB0 à 1 et ensuite à 0.

detecterObjet() : Celle-ci est utilisée pour détecter l'écho du sonar et retourne un int.

distanceObjetMetre() : Cette dernière est utilisée pour calculer la distance de l'objet avoisinant s'il y en a un, grâce a ce qui est retourné par *detecterObjet().*

- **Horloge :** L'horloge avance d'une minute à chaque fois qu'un signale est envoyé sur la deuxième ligne de signal. Pour ce faire, lorsqu'il s'agit de définir une heure de départ, on calcule seulement le nombre de signale équivalent à l'heure entré par l'utilisateur, soit 60*(nbHeure)+nbMinute. Pour ce qui est de la simulation, l'horloge utilise l'interruption du timer0 en mode CTC avec un prescaler de 1024. Or, comme il existe un overflow de 30, on incrémente un compteur dans l'ISR du timer tant qu'il n'a pas atteint la valeur 30. Lorsque le compteur atteint 30, un signal est envoyé et ainsi l'horloge avance. Compte au voltage, lorsque celui-ci est modifié, la valeur maximale OCR0A est modifiée.

void envoyerSignalHorloge(int signal) : Cette méthode envoie le nombre de signale passé en paramètre à l'aide d'une boucle pour démarrer l'horloge à une heure précise.

void reinitiHorloge() : Cette méthode envoie un signale sur la première ligne de signale pour réinitialiser l'horloge a 0;

void ajusterVoltage() : Cette méthode ajuste OCR0A en fonction du voltage selon la logique suivante :sachant que 30*255=7812 (environ);

voltage=(voltageEnBinaire/255)*5; accelerateur= (voltage/5)*60

OCR0A=255/accelerateur;

- **LCD :** Pour ce qui est de l'écran LCD, on utilise les classes qu'on nous a fournies pour afficher sur l'écran. Les méthodes les plus utilisées sont: disp.write(), disp.clear() et disp<<.

- **Servomoteur :**Pour contrôler le servomoteur, il nous a fallu 3 méthodes qui se trouvent dans les fichiers moteur.h et moteur.cpp

angleServo(): Cette méthode nous permet de convertir l'angle entrée à l'angle réel sur le servomoteur ,en utilisant une formule mathématique qui est ANGLE_ZERO + (angle/ RAPPORT_ANGLE). Ou ANGLE_ZERO = 126 et RAPPORT_ANGLE = 1.44

faireTournerServoE(): Cette méthode nous permet de positionner le servomoteur E à l'angle voulu à l'aide d'un signal pwm. En effet, on utilise le timer1 pour activer le mode fast pwm et une division d'horloge de 64 . Aussi, on met OCR1A = angleServo().

faireTournerServoF(): c'est la même méthode que la précédente mais pour le servomoteur F. Donc, OCR1B = angleServo().

- **Clavier :**

donnerCourantColonne() : Cette fonction consiste donner le courant à la colonne concernée. Dépendamment de s0 et s1. Une fois qu'on a donner le courant, il est maintenant temps de scruter les lignes une par une. Pour faire cela, nous avons fait appel à la fonction qui suit :

envoyerCaractèreLcd() : cette dernière contient en elle une boucle qui parcours les lignes et cette boucle est imbriquée dans une boucle qui parcours les colonnes. Ainsi nous savons ou nous avons envoyer le courant, maintenant on cherche avec le s0 et s1 du mux quelle ligne qui fait en sorte que notre PINC6(en entrée) est à 1. Ainsi, grâce à un compteur ligne et un compteur colonne, initialement à -1, on va leur donner la valeur de la ligne et celle de la colonne de la touche appuyée. Par le fait même on en profite pour afficher cette valeur dans le Lcd et dans le terminal RS232 en allant la cherchée dans le tableau suivant const char TAB_CLAVIER [LIGNE][COLONNE]

- **Bouton-poussoir :**Le bouton-poussoir nous permet d'arrêter la simulation lorsqu' il est actionné. Pour y arriver, nous avons opté pour une routine d'interruption externe.

- **Barres de DEL :**Pour programmer ce dispositif, nous avons utilisé d'abord des boutons de 5V déjà branchés au système au début afin de bien comprendre le fonctionnement. Une fois bien compris, nous avons lié celui-ci au processeur et avons créé la même logique que lorsqu'il y avait les boutons, mais avec du C cette fois.

D’abord nous avons define pour chaque bouton une variable lorsqu’il est appuyé (ON) et lorsqu’il ne l’est pas (OFF). Ensuite nous avons juste programmé des fonctions pour effectuer les manipulation manuelles qu’on faisait avec les boutons mais automatiquement maintenant grâce à la programmation.

void lireDS() cette fonction représente un appui sur le bouton SH_CP et le relâcher, mais avec un temps d’attente de 125 ms entre chaque.

void copierRegistre() Cette fonction représente un appui sur le bouton ST_CP et le relâcher, mais avec un temps d’attente de 125 ms entre chaque.

void etteindreLED() Cette fonction sert à effectuer la manipulation qui permet de réinitialiser tout le module et de ainsi fermer toutes les lumière LED et d' oublier tous les appuis antécédents sur d’autres boutons.

void ignorerPorte() Cette fonction de remplir huit LED d'affilée avec des 0, c’est-à-dire leur stocker en mémoire 8 LED éteintes. Cette astuce nous permet d’effectuer la manipulation de fermeture ou d'ouverture de porte sur la porte qu’on veut tout dépendamment du nombre d’ignorePorte qu’on dépose dans le code.

void ouvrirPorteA / B / C / D Cette fonction est le fruit final de l’utilisation des fonctions nommées précédemment et permet d’effectuer une animation d’ouverture d’une porte. C’est possible grâce à une boucle for qui va à chaque fois allumer une LED, éteindre la LED est allumé la subséquente.

void fermerPorteA / B / C / D Cette fonction est le fruit final de l’utilisation des fonctions nommées précédemment et permet d’effectuer une animation de fermeture d’une porte. C’est possible grâce à une boucle for qui va à chaque fois allumer tout LED, éteindre la LED est allumée 7, ensuite 6 jusqu'à que ça s'éteint

- **Fonctionnement :** Important : pour l’option 3-4-5-6, nous avons fait appel à une structure de donnée nommée Action. Celle-ci contient 4 champs. Un tableau de caractère pour stocker l’heure, un autre pour l’action à exécuter, un boolean pour dire si la case est pleine ou non.

Option 1 : Pour réaliser l’option qui permet de saisir une heure, on appelle la méthode *reinitiHorloge() qui permet d’envoyer un signal sur la pinA7.* Ensuite, on appelle la méthode *envoyerHeure()* et *envoyerSignalHorloge()* pour stocker l’heure dans un tableau.

Option 2 : Pour définir l'état des dispositifs, on fait appel a 2 méthodes: *recupererDispositif() et option2()* qui s'occupe de récupérer ce que l'utilisateur a saisi au clavier et d'exécuter les actions avec les méthodes des barres de DEL et des servomoteurs.

Option 3 : L'option 3 comporte 2 méthodes *trierActions() et afficherActions()*. La première trie la liste d'action selon l'heure et la deuxième affiche la liste d'actions.

Option 4 : Cette option fait appel à 2 méthodes, la première étant entrerHeure() pour saisir l'entrée de l'utilisateur. La deuxième c'est récupérerDispositif() qui lit l'action à effectuer. Enfin, on ajoute ces derniers champs dans un tableau d'action et on met le bolean à vrai pour dire que la case est pleine.

Option 5 : Ici nous faisons usage de lireActionSupprime() qui lit l'index de l'action et nous le retourne. Ensuite, on supprime l'action avec supprimerAction();

Option 6 : Ici on fait appel aux fonctions liée à l'horloge afin de pouvoir ajuster le temps l'accéléré ou le ralentir. Nous faisons appel aux interruptions (bouton,sonar) donc, c'est ici aussi que nous utilisons les fonctions en lien avec le sonar afin d'interrompre la simulation.

2. Expérience de travail à distance

Nous trouvons que l'expérience de travail à distance fut assez compliquée. Il est vrai que nous avions eu beaucoup de difficultés lors du projet final, non à cause de la difficulté, mais des circonstances de travail elles-mêmes. Nous avions de nombreuses remises et de nombreux travaux à remettre pendant la même période, donc nous étions limités au temps qu'on possédait ensemble en équipe face à une charge de travail pareille. Nous sommes cependant une bonne équipe techniquement et chacun de nous possède des compétences d'analyse et de programmation assez bonnes. On s'est donc débrouillés pour pouvoir remettre un projet fonctionnel et qui respecte les directives.

Concernant nos compétences de communication et de travail d'équipe, certains d'entre nous peuvent parfois trouver des difficultés pour travailler selon le stress qu'ils ressentent. Avoir plusieurs examens en même temps et des remises peut peser sur le mental des membres de l'équipe et la qualité de travail et de coopération peuvent en subir les répercussions. De nouvelles stratégies pour le développement des compétences de gestion de soi et de gestion de temps sont des pistes d'amélioration possibles à apporter. Cela dépasse peut-être le cadre du cours et relève de défis personnels. Tout un chacun devrait prendre un moment d'introspection à la fin de ce cours pour se demander comment réutiliser les connaissances acquises pour atteindre ses objectifs personnels.