Département de génie informatique et génie logiciel

INF1900
Projet initial de système embarqué

Rapport final de projet

Projet dans SimulIDE

Équipe No **1112**

Section de laboratoire 01

Maxime Laroche
Ahmed Gafsi
Eduardo Falluh
Marie-Claire Taché
Nemro Yapni Nji Monluh

21 avril 2021

Robot qui détecte des obstacles et qui effectue des manœuvres d'évitement

L'objectif de ce projet est de programmer un microcontrôleur Atmega324pa d'une simulation d'un robot qui se déplace et lorsqu'il rencontre des obstacles, celui-ci déploie des manœuvres afin de les éviter.

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

Machine à états finis (FSM):

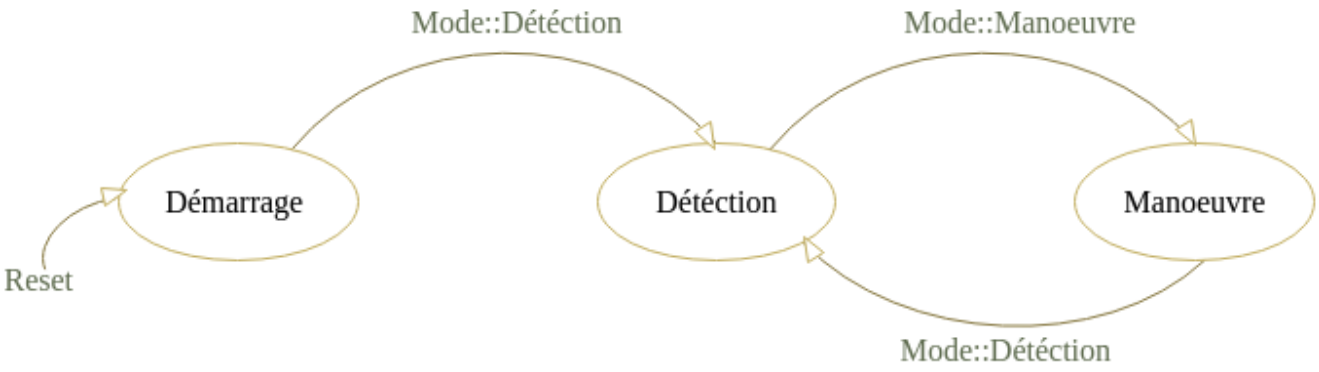


Figure 1: diagramme d'état du projet

Mode démarrage:

La classe startMode qui s'occupe de ce mode implémente 3 fonctions qui permettent d'afficher sur le terminal de communication série le taux de transmission de la communication RS-232 qui est de 9400 bps dans notre cas. Puis elle affiche respectivement les lettres A, B, C et D sur les 7-segments durant 2 secondes, ensuite ils s'éteignent et les deux roues tournent à pleine puissance en sens horaire pendant 1 seconde, puis en sens antihoraire pendant 1 seconde. Après, on passe au prochain mode détection.

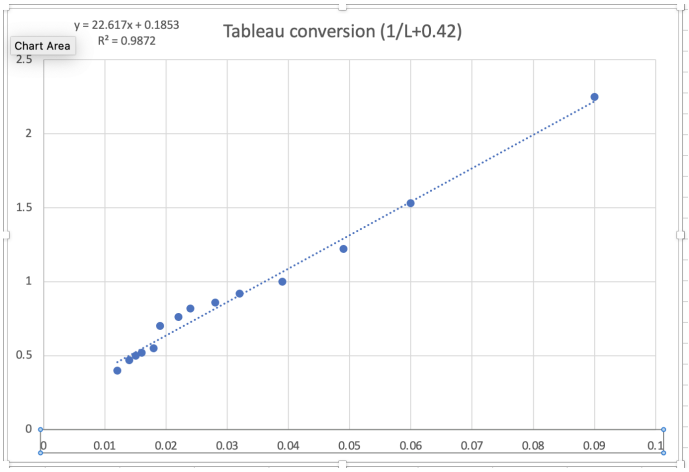
Mode détection

Capteurs et Détection

La classe détection est celle qui sert à lire les valeurs des capteurs, imprimer le temps, l'état (OK, ATTENTION, DANGER) des capteurs.
Afin de pouvoir traiter les résultats émis par les capteurs de distances, nous avons dû déterminer, à partir de la documentation, une équation pouvant représenter la relation entre le voltage et la distance en cm. En prenant des points du graphique dans la documentation nous avons pu en tirer un graphique comme suit :

Figure 1 : Graphique de courbe de conversion SHARP GP2D12 et tableau associé

DISTANCE	VOLT
0.09	2.25
0.06	1.53
0.049	1.22
0.039	1
0.032	0.92
0.028	0.86
0.024	0.82
0.022	0.76
0.019	0.7
0.018	0.55
0.016	0.52
0.015	0.5
0.014	0.47
0.012	0.4



L'équation obtenue est y = 22.61x + 0.1853. Toutefois, en testant notre système, nous nous sommes rendu compte que la distance obtenue manquait de précision. Ceci est dû au fait que nous avons dressé une courbe avec seulement 14 points. Par contre, avec la documentation fournie il nous est difficile d'en trouver d'autres à l'œil nue. Ainsi, nous avons simulé la courbe, comme si elle contenait un plus grand nombre de points, et ce en faisant varier le taux de variation. Finalement nous sommes tombées sur cette équation :

$$y = 26x + 0.09 \approx 26x + 0.1.$$

Ce n'est toutefois pas l'équation finale, car il faut tenir compte de deux éléments.
1- La valeur que transmet le canEXT et le canINT est différente que celui reçu:

$$V\,ext = \frac{Vreçu}{256} \times 5V \int ¿ \frac{Vreçu}{1024} \times 5$$

2- Le graphique tient en compte une distance selon la relation suivante: (1/Distance + 0.42)
Ce qui donne, au final, l'équation suivante:

$$Distance = \frac{(26\,x\,Resolution)}{(Aref\,x\,valeur) + (0.1\,x\,Resolution)} - 0.42$$

Aref : Aref canEXT = 2.48 (le Aref du canEXT a été modifié pour permettre une meilleure précision) , Aref canINt = 5

Resolution : Resolution can EXT = 256, Resolution canINT = 1024

Pour ce qui est du temps, la classe contient un objet de la classe Time pour gérer le temps. Les fonctions pour imprimer utilisent la classe UART de la librairie pour accomplir leurs fonctions

Mode manoeuvre

Ici, nous avons défini une classe Manoeuvre qui regroupe les différentes manœuvres sous forme de méthodes. Chaque méthode exécute la manœuvre concernée. Pour pouvoir exécuter notre manœuvre, l'on doit créer un objet Manoeuvre et appeler la méthode de la manœuvre appropriée. Nos différentes manœuvres permettent de contrôler les moteurs de notre robot, d'ajuster les vitesses des roues et d'afficher les différentes vitesses sur les afficheurs 7-segments. Le mode d'affichages des vitesses peut passer de décimal en hexadécimal selon que l'on appuie sur la touche #.

Time:
Contient un ISR pour mettre à jour le temps et une classe Time initialise le tout, stocker les valeurs de temps et convertir le temps actuel en string pour pouvoir imprimer le tout.

Clavier:
Nous avons défini 5 fonctions importantes pour que notre clavier fonctionne. Les voici:
Dans notre code au fichier clavier.cpp, nous implémentons deux fonctions qui nous permettent de représenter les lignes et les colonnes qui représentent le clavier 3x3. C'est pour cela qu'au début de l'ISR, nous définissons les lignes et colonnes à 3 pour démontrer ce clavier 3x3. De plus, ces deux fonctions ci-dessus sont synchronisées/définis plus bas dans clavier.cpp avec leur PIN respectif. Ensuite, une fois rentrées dans la boucle, nous avons une fonction qui sert à mettre nos colonnes et lignes à entrée et sortie et vice-versa. De plus, nous avons implémenté une fonction qui initialise notre code ce qui permet de set notre interrupteur correctement. Nous utilisons cela dans presque tous nos laboratoires, il serait donc redondant de l'expliquer en détail. Ensuite quand nous rentrons dans la deuxième boucle dans l'ISR, nous rencontrons une fonction qui permet de représenter les caractères enfoncés du clavier. Celui-ci est défini lui aussi en bas du clavier.cpp. En gros, nous avons décidé de faire un gros switch case pour nous permettre de représenter nos caractères avec leurs positions qui sont définies. On utilise le state.frequency défini au début du fichier clavier.cpp(ou keyboard provient de detection.cpp) pour envoyer la valeur 1,2 et 4. Ensuite, nous utilisons state.print = PrintMode::R, ou PrintMode provient du fichier detection.cpp, pour print R, V, et C. Pour les caractères I et E, les deux lignes sous les deux cases permettent de placer les switch de façon à lire les can interne et can externe. Finalement, pour le #, ceci affecte le fichier manœuvre.h. En effet, lorsque l'on clique sur celui-ci, il convertit ce qui est en hexadécimal en décimal. Si nous revenons à la fin de l'ISR,

celle-ci permet simplement d'imprimer le message attendu à la bonne place avec les bons caractères en place. Le PCMSK2 permet simplement de choisir nos PINS pour l'interruption.

Affichage 7 segments:
Au nombre de quatre (04), les deux (02) les plus à gauche permettent d'afficher les vitesses de la roue gauche tandis que ceux les plus à droite permettent d'afficher les vitesses de la roue droite. Au démarrage de notre robot, ces afficheurs affichent de gauche vers la droite les caractères A, B, C, D.

Le bouton
Le bouton connecté PD2 est implémenté par interruption sur INT0 pour permettre de passer en mode Manœuvre.

2. Expérience de travail à distance

Afin de faciliter la communication, nous avons privilégié l'utilisation de la plateforme Discord. Nous l'utilisons d'abord pour faire des rencontres d'équipes ou nous avons la possibilité de partager notre écran ce qui favorise l'entraide. De plus, hors de nos rencontres nous pouvons écrire à toute l'équipe pour poser des questions et nous mettre à jour sur nos parties individuelles du projet. Étant donné les horaires chargés de tous, c'était un défi de trouver un moment avec tout le monde, il est devenu très important qu'on communique par message au fur et à mesure que le projet avançait.

Concernant notre méthode de travail, nous avons décidé de nous séparer les tâches du travail en nous donnant une date pour la terminer. Par contre, nous nous sommes rendu compte que cette méthode était à notre désavantage puisque nous nous retrouvions à travailler chacun de son côté, sans nécessairement avoir de l'aide et avec certains qui ont plus de travail que d'autres. En commençant le projet, nous avons encore séparé les tâches, mais nous avons insisté sur l'importance de communiquer entre nous pour se tenir au courant ou pour demander de l'aide. Ceci a été un fort succès et nous a permis de finir le travail à l'avance tout en nous entraidant.

En termes de décisions d'équipes, nous optons dans la majorité des cas, pour un consensus où l'on essaye de rassembler les idées de tous.

De manière générale, nous aurions tous aimé vivre ce projet en présentiel puisque nous aurions évidemment eu l'occasion de manipuler un vrai robot avec les composantes matérielles. Le fait d'être en présentiel aurais aussi eu un impact sur notre cohésion d'équipe ainsi que notre communication.