Projet Robot 2

Contrat 7 : Caniche IR

**Introduction :**

Le but ce contrat est de faire acquérir au robot un certain nombre de fonction lui permettant de se déplacer dans un environnement prédéfinit. Il doit être capable, grâce aux signaux issus des capteurs infrarouges (IR) gauche et droite, de se déplacer vers une cible mobile devant le robot. Si cette cible est à moins de 1,5m le robot avance vers celle-ci et s’arrête lorsqu’il est à moins de 40cm de la cible. Il doit se déplacer à une vitesse comprise entre 7 et 14 m/min.

I- Séance 1 :

1) Les objectifs du contrat 7 :

Les objectifs qui doivent être impérativement réalisés sont :

a- mettre en marche et atteindre le robot en appuyant sur le bouton du centre de la télécommande.

b- acquérir les informations à travers le bus I²C, des informations fournies par la télécommande, avec une vitesse de transmission de 50kBauds.

c- afficher les données grâce aux LED, D1 à D8 commandées via le bus I²C.

d- mémoriser la tension issue de la batterie.

e- mesurer la tension issue de la batterie en faisant une moyenne de 4 valeurs sur un intervalle régulier.

f- générer les signaux modulés en largeurs d’impulsions droite et gauche avec une période de **1ms** et un rapport cyclique strictement **inférieur à 50%.** En effet la tension d’alimentation de chaque moteur étant de 12V alors que les moteurs ont une tension nominale de 6V, le moteur tourne donc à sa vitesse maximale avec un rapport cyclique de 50%.

g- changer le sens de rotations des moteurs gauche et droite afin de changer le sens de rotations des chenilles de sorte à **tourner**.

h- allumer et éteindre les capteurs IR droite et gauche, afin d’alimenter uniquement les capteurs quand ils sont sollicités pour limiter la consommation **énergétique** de la carte.

i- acquérir les tensions issues des **deux capteurs** Infra Rouge droite et gauche donnant une représentation de la distance entre le capteur et l’obstacle.

j- utiliser la liaison série **RS232** avec une vitesse de 9,6kBauds, afin d’afficher la tension moyenne de la batterie, la fin de l’initialisions des variables ainsi que leur état.

k- réaliser des **interruptions** via la télécommande et le timer0.

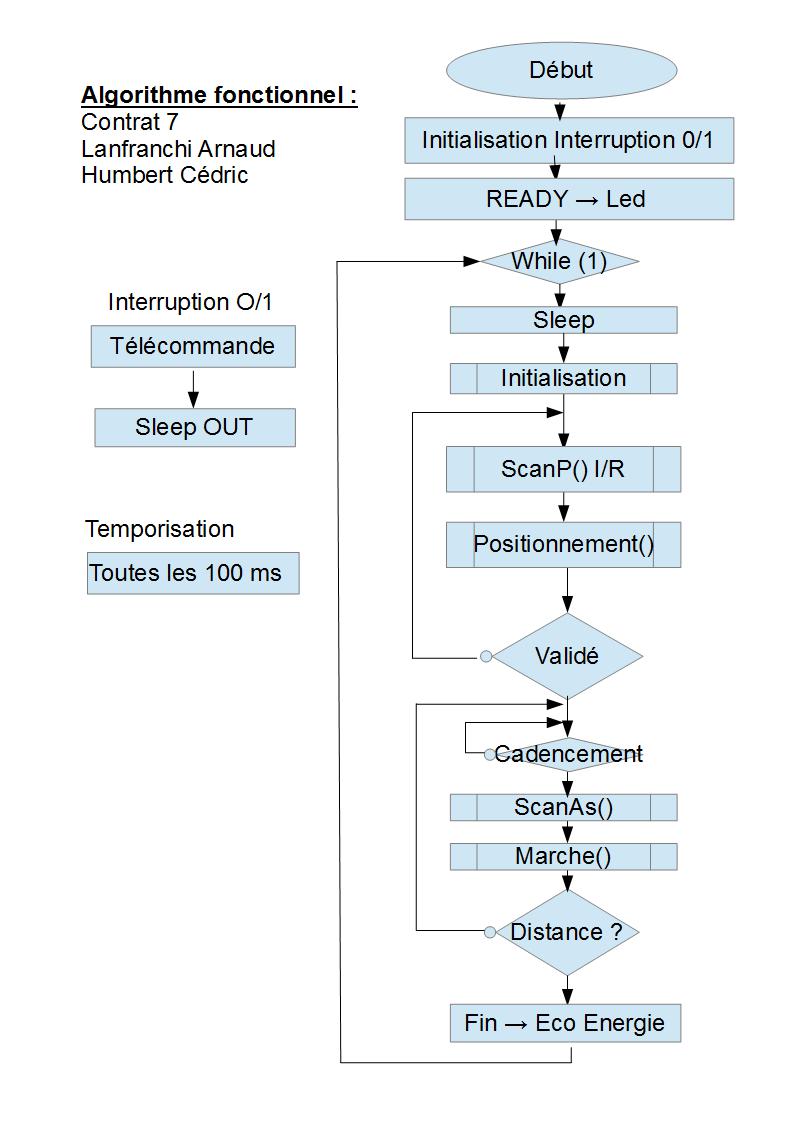
m- Utiliser un minimum de mémoire pour la ram et le code, qui sera sauvegardé sur une ROM programmable et effaçable électriquement. Pour cela, nous portons une attention particulière à la taille des variables en fonction de leur utilisation.

2) Repérage des composants nécessaires pour atteindre les objectifs impératifs :

|  |  |
| --- | --- |
| Fonctions | Composants concernés ainsi que leurs broches |
| **on/off** du robot via bouton centrale de la télécommande via la liaison **I²C** qui relie le µC et le bloc de réception de la **télécommande** | RX1 broche 3 in analog  broche 14 out analog  U4 PIC16 broche 2 in analog P2A  broche 6 out digital RC4  broche 9 out digital (data I²C) SDA  broche 10 out digital (clock I²C) SCL  U1PIC18 broche 21 in digital INTO  broche 14 in digital (clock I²C) SCL  broche 15 out digital (data I²C) SDA  broche 21in digital INT0 (IT) |
| afficher données I²C grâce aux LED D1 à D8 | U8 PCF8574 broche 14 in digital clock I²C  broche 15 in digital data I²C  broche 4 out digital LED D1 P0  broche 12 out digital LED D8 P7 |
| mesurer, mémoriser et faire une moyenne de 4 valeurs la tension issue de la batterie | U1PIC18 broche 4 in analog AN2 (SURVBAT) |
| générer signaux PWM avec T=1ms et α<0,5. | U1PIC18 broche 12 out digital CCP2 (PWMD)  broche 13 out digital CCP1 (PWMG)  Timer 1 et 2 |
| changer le sens de rotation des moteurs | U1PIC18 broche 9 out digital RA7 (DIRG)  broche 10 out digital RA6 (DIRD) |
| allumer et éteindre les capteurs IR | U1PIC18 broche 22 out digital RB1 |
| acquérir les tensions issues des deux capteurs IR | U1PIC18 broche 2 in analog AN0 (CAPTD)  broche 3 in analog AN1 (CAPTG) |
| utiliser la liaison série RS232 avec une vitesse de 9,6kBauds, afin d’afficher la tension moyenne de la batterie, la fin de l’initialisions des variables ainsi que leur état. | U1PIC18 broche 17 out analog TX (TX1)  broche 18 in analog RX (RX1)  MAX232 broche 11 in analog T1IN (TX)  broche 12 out analog R1OUT (RX) |
| réaliser des interruptions via la télécommande et le timer0 | U1PIC18 broche 22 out digital INT1  broche 23 out digital INT2 (CAPTARR) |
| faire un système économique en énergie | U1PIC18 broche 22 out digital INT1  broche 23 out digital INT2 (CAPTARR) |

II- Séance 2 :

1) Algorigramme fonctionnel :



2) Détails des différentes fonctions :

On utilise une fréquence de fonctionnement pour le PIC18F de 8MHz. Il faut donc configurer l’oscillateur interne du microcontrôleur via le registre OSCON :

D’après la page 32 de la documentation du PIC18 il faut mettre les bits IRCF2, IRCF1 et IRCF à ‘1’ :

OSCCONbits.IRCF2 = 1;

OSCCONbits.IRCF1 = 1 ;

OSCCONbits.IRCF0 = 1 ;

a- mettre en marche et atteindre le robot en appuyant sur le bouton du centre de la télécommande :

La télécommande génère une interruption sur la broche 21qui réveille le robot.

Le bouton central de la télécommande transmet le code 3h=0011% cela correspond au code ASCII 33h. Si la broche 14 (SDA) reçoit la donnée 33h=0011 0011%.

C’est la broche 0 du port B, c’est une entrée numérique en mode INT0 (interruption).

Il faut configurer cette broche en entrée : TRISBbits.RB0=1;

Pour ce qui est des interruptions, nous pensons qu’une gestion des priorités est adéquate, pour cela il faut mettre le bit IPEN du registre RCON à ‘1’ (RCONbits.IPEN=1 ;). Il faut activer les interruptions de hautes priorités (INTCONbits. GIE=1 ;) et les interruptions de basses priorités (INTCONbits.PEIE=1 ;)

Ensuite via le registre INTCON, il fait les valider (INTCONbits.INT0IE=1 ;) .

Puis il faut activer l’interruption pour la mise en marche du robot avec INTCONbits.INT0IF=0;

b- acquérir les informations I²C fournis par la télécommande, avec une vitesse de transmission de 50kHz=50kBauds :

Il faut mettre les broches 14 et 15 en entrée (TRISCbits.RC3=1; TRISCbits.RC4=1;) I²C (SSPCON1bits. SSPEN=1 ;) Avec cette dernière commande on utilise le mode SCL (respectivement SDA) de la broche 14 (respectivement SCL). Pour avoir une vitesse de transmission de 50kHz, on sélectionne d’abord le mode Master I²C :

SSPCON1bits.SSPM3=1 ; SSPCON1bits.SSPM2=0 ; SSPCON1bits. SSPM1=0 ; SSPCON1bits.SSPM0=0 ;

On utilise la formule Avec Fosc=8MHz et SSPADD une valeur numérique.

= 0010 0111%

Puis on met la valeur 39 dans SSPADD : SSPADD=39 ;

d- mémoriser la tension issue de la batterie et e- mesurer la tension issue de la batterie en faisant une moyenne de 4 valeurs sur un intervalle régulier :

La tension de la batterie arrive sur la broche 4 du PIC18 (au fait c’est la tension de la batterie divisé par 3,2), il faut donc d’abord configurer la broche 4 en entrée (TRISAbits.RA2=1 ;) et analogique (PORTAbits.RA2=1 ;). --------------

f- générer les signaux modulés en largueur d’impulsions :

Les signaux modulés en largeur d'impulsions, qui permettent des contrôles la vitesse de rotation des moteurs sont en sortie des broches 12 et 13 du microcontrôleur.

Il faut dans un premier temps configurer ces broches en sortie (TRISCbits.RC1=0;

TRISCbits.RC2=0;).

On souhaite un période PWM de 1ms soit une fréquence de 1kHz.

Le choix de la période PWM se fait grâce à la valeur stockée dans le registre

PR2 = 1. Cette valeur doit être inférieure à 255, le prescaler peut être seulement de 1,4 ou 16. Avec un prescaler de 4 PR2=499 ce qui est trop grand donc il faut un prescaler de 16 (T2CONbits.T2CKPS1=1;) PR2=124 avec ce prescaler.

Ensuite pour générer les signaux PWM il faut utiliser le mode CCP2 pour la broche 12 et CCP1 pour la broche 13.

CCP2CONbits.CCP2M3=1;

CCP2CONbits.CCP2M2=1;

CCP1CONbits.CCP1M3=1;

CCP1CONbits.CCP1M2=1;

CCP1CONbits.P1M1=0;

CCP1CONbits.P1M0=0;

h- allumer et éteindre les capteurs IR droite et gauche :

Le contrôle de l'alimentation des capteurs IR de fait grâce à la broche 22 du microcontrôleur. Il faut d'abord la configurer en sortie (TRISBbits.RB1=0;).

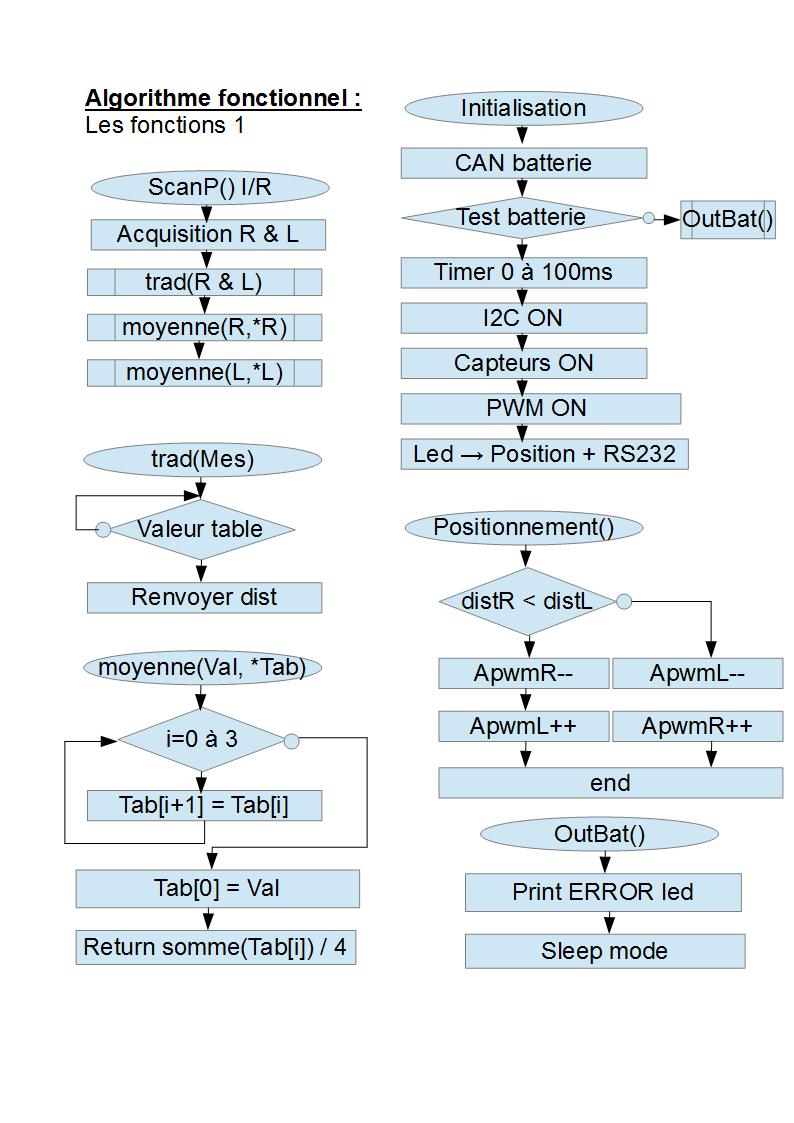
Ensuite pour alimenter les capteurs IR, il faut saturer le transistor pnp, donc mettre un niveau bas sur la base de ce transistor (PORTBbits.RB1=0;) et pour éteindre le capteur il faut bloquer le transistor avec un niveau haut sur sa base (PORTBbits.RB1=1;).

3) Architecture du logiciel :

Le programme de contrôle du robot tel que décrit plus haut dans une modélisation fonctionnelle est décomposé et organisé en mémoire tel que suit.

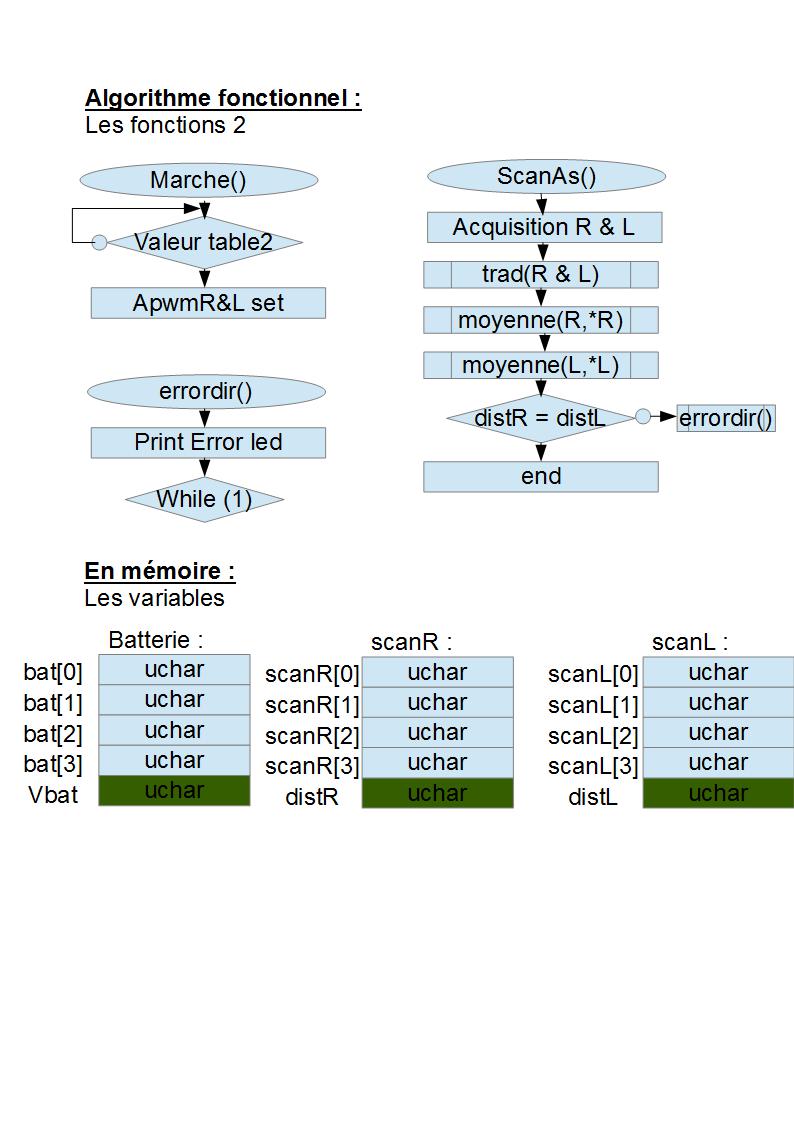
Le projet contiendra 4 fichiers, le fichier principal, mainC7VX.c contenant le main et le programme d’initialisation et les interruptions un header général, headC7VY.h ou l’on trouvera les prototypes des différentes fonctions. De plus, le projet contiendra un fichier C, fonctionC7VX.c contenant les fonctions nécessaires au fonctionnement du robot aussi qu’un fichier header contenant des tableaux de correspondance pour les différentes phases de pilotage du robot.

Ci-joint les alogorigrammes des fonctions annexes.



Les fonctions présentées ici sont cependant soumises à des modifications et adaptations suite à des erreurs ou des bugs relevés lors de la phase de test. Les termes décrivant les actions sont volontairement orienté fonction plus que code, ils serviront en effet à commenter et organiser le code, la phase de code consistera à faire la correspondance entre les instructions et ces notions. En somme, certaines parties du code correspondent déjà à des parties vues en TD, il ne s’agit donc pas de les développer outre mesure. Les tables 1 et 2 (cf plus bas) sont des tables de correspondance pour la commande de la machine.

Suite des fonctions :



De plus, dans la fonction moyenne, i est un indice codé donc sur un char.

Les valeurs ApwmR et ApwmL sont les variables correspondant au rapport cyclique de la commande du PWM. Les valeurs L, \*L, R et \*R sont les symboles des tableaux et des valeurs représentés plus haut. Les entrées sur les CAN se feront sur 8bits en effet, notre table de correspondance est indexée sur des pas variables d’environ 25mV, un quantum de 5mV n’aurait donc aucun sens.

Il a été fait le choix d’utiliser des tableaux afin de gommer les erreurs de mesures liés à des fortes demandes de courant sur la batterie ou à un fort changement dans éclairage de la pièce.

La table 1 fait une correspondance entre les distances mesurées par les capteurs IR et la valeur en cm, le tableau excel où figurent cette table est joins. Ces tables se trouvent en ROM et seront donc étalonnés durant les tests. La table 2 fait correspondre des valeurs de ApwmR et ApwmL avec la distance retrouvée de sorte à générer une accélération et une décélération amorties dans le temps.

Les signes d’égalité en particulier dans la fonction ScanAs() sont des égalités relatives couvrant les erreurs éventuelles de mesure.

Nous avons fait le choix de diviser la fonction d’initialisation en deux parties en effet, nous désirons pour des raisons d’économie d’énergie mettre notre robot en mode sommeil lorsqu’il n’est pas utilisé, il nous faut donc procéder à un arrêt / allumage d’un certain nombre de ses périphériques que nous plaçons dans une fonction d’initialisation. Cependant, un certain nombre de réglage n’ont pas à être refaits lors de ces arrêt / marche donc nous procéderons aussi à une première initialisation en dehors de la fonction d’initialisation.

4) Les test et le Debug :

Pour ce qui est de la partie test, nous envisageons de tester nos fonctions au cours du processus de codage sur les modèles numériques afin de s’assurer du bon fonctionnement de notre commande et de procéder aux réglages des constantes pour les timers de scan par exemple ou pour les tables. Pour cela, nous construirons les modèles avec les signaux correspondants aux tests des fonctions spécifiques.

La phase de debug avec la liaison RS232 sera cadencée sur les temps de bouclage des différents instants du système. Ces temps ne sont encore pas connus à l’heure où nous écrivons ce texte et seront des paramètres importants de nos premiers tests sur simulateur et sur le robot.