Projet
UBI

Compagnie UBICOP
D
Rapport Final

Membres

Sébastien Pouliot
Pierre Martin-Dumont
Simon Mercier
Dominic Giroux
Bruno Boulianne
Tommy Desrosiers
Charles Hamel
Donald Séguin

Dans le cadre du cours
Système électronique

Université Laval
17 décembre 1998

Table des matières

Liste des figures

Introduction

De nos jours, plusieurs domaines technologiques sont en progression constantes. Notamment la robotique et l'intelligence artificielle, il nous est maintenant possible d'intégrer ces disciplines dans un projet d'envergure. Justement, le MIT a développé un robot qui allie ces deux techniques. Notre but dans cette course à une meilleure connaissance, a été de développer et de rendre plus autonome ce petit engin.

Le but de cette exercice aura été d'intégrer différentes connaissances techniques et de gestion de projet dans le cadre d'un cours universitaire. Nous devions intégrer au robot un système de localisation sonore ainsi qu'une détection d'obstacle. Ce robot devra être contrôlé à distance à partir d'un interface simple sur ordinateur hôte. Le plus gros du travail aura été le développement d'une communication sans fil entre le robot et l'ordinateur. Dans ce rapport, nous présenterons chacune des parties sur lesquels les membres de notre équipe auront travaillé. Sous chacun des aspects, une présentation détaillée sera fait pour bien comprendre le fonctionnement des différents modules. En plus, les difficultés et les erreurs rencontrées seront présentées. En premier lieu, nous aborderons l'interface graphique sur l'ordinateur hôte. Ensuite suivra le contrôle du robot ainsi que sa localisation dans l'environnement. En troisième lieu viendra le module de détection d'obstacle et pour finir, nous aborderons une description du module de la communication sans fil.

Chapitre I

Interface graphique

Chapitre 1 : Interface graphique

1.1 Objectifs

Créer une interface usager pour effectuer le contrôle du robot. De plus, pour ne pas surcharger les fonctions exécutées sur le robot, la plupart des traitements seront effectués sur l'ordinateur.

Dans ce qui suit, vous trouverez le compte-rendu du développement de l'interface usager (UBINTERFACE) ainsi que les différents algorithmes nécessaires au contrôle de notre robot UBICOP.

1.2 Plan général

La figure 1.1 représente le plan général de notre programme en mettant en évidence les principaux modules ainsi que les échanges d'informations entre le robot et l'interface usager.

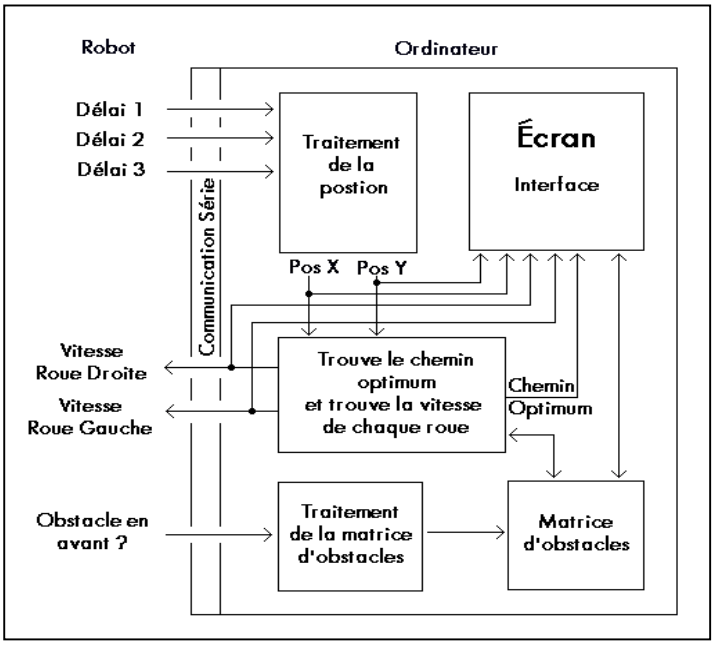

Figure 1.1 - Plan du programme

1.3 Matrice d'obstacles

Nous commençons par décrire la matrice d'obstacles, car tous les autres modules dépendent de celui-ci.

La matrice d'obstacles représente le terrain de 5 m par 5 m que nous avons divisé en carreaux. La grandeur des carreaux peut être changée facilement dans le programme. Nous avons choisi cette grandeur en fonction de la précision que nous avons pour la position du robot et pour le positionnement des obstacles. La taille optimum des carreaux est de 50 cm par 50 cm, ce qui est un bon compromis entre la précision du positionnement spatial et l'exactitude des déplacements.

La matrice contient la position des obstacles physiques ainsi que les obstacles virtuels que nous pouvons ajouter à partir de l'interface. Chacun de ces deux types d'obstacles possède son propre code dans la matrice de façon à ce que l'on puisse les différencier dans la matrice. Sur l'interface, les obstacles virtuels sont représentés par des carrés noirs et les obstacles réels, par des carrés verts. Pour faciliter le traitement dans les algorithmes, nous simulons un obstacle autour de la matrice afin que le robot ne sorte pas du terrain. De plus, pour éviter que le robot passe trop près des obstacles, nous avons ajouté une zone grise autour de chaque obstacle.

1.4 Traitement de la matrice d'obstacles

Nous faisons une mise à jour dynamique de notre matrice d'obstacles. Nous n'avons pas à nous soucier des obstacles virtuels placés par l'interface usager, car c'est l'usager qui va décider s'il veut les enlever ou non. Cependant, il peut arriver que sur le terrain, il y ait des obstacles physiques qui s'enlèvent, s'ajoutent ou se déplacent. Il est donc primordial de faire la mise à jour de notre matrice de façon dynamique.

Le positionnement des obstacles physiques est possible grâce à l'information transmit par le robot. Les routines de contrôle implantées dans le robot vérifient à chaque cycle si un obstacle lui fait face. Si c'est le cas, l'information nécessaire est aussitôt envoyée à l'interface via le lien série. Ainsi, nous pouvons vérifier dans la matrice s'il y a un obstacle ou non et faire les ajustements nécessaires.

Le robot utilise deux types de détection. Le premier est réalisé par les infrarouges qui permettent de voir les obstacles à distance(environ 30 cm). Le deuxième type est les représenté par les détecteurs de collision du robot. Dans les deux cas, il est possible de savoir si l'obstacle est à gauche, à droite ou au centre et il est ajouté en conséquence dans la matrice d'obstacle.

1.5 Communication Série

La communication entre le robot et l'interface usager est possible via le lien série RS-232 du PC. La transmission se fait sur le lien TX et la réception sur le lien RX. Il suffit de placer les données dans les registres tampons du port et celle-ci sont envoyées directement sur le lien série.

Pour notre programme d'application, nous effectuons le contrôle du lien série avec un objet de la classe CSerialPort. À l'ouverture de l'application, le port série est ouvert. Il est possible de modifier la configuration de la communication série comme le montre la figure 1.2. En fait, les seuls paramètres qui nous sont utile de modifier est l'identification du COM et la vitesse de transmission. En effet, chaque station de travail sur lesquelles nous avons travaillé n'utilise pas nécessairement le même COM, donc la possibilité de modifier ce paramètre nous à été fort utile. Pour ce qui est de la vitesse de transmission, l'équipe qui s'occupe de créer le lien série sans fils n'était pas en mesure de nous dire quelles seraient les vitesses supportées par ce module, donc nous avons conservé la possibilité de modifier ce paramètre. Les autres paramètres sont fixés car on sait que le robot communique toujours sur son port série avec la même configuration.

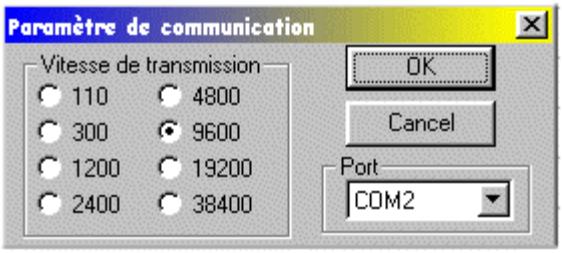

Figure 1.2 - Configuration du Port Série

La gestion du port série est prise en charge par un processus indépendant du logiciel d'application (thread). Donc, lorsque des données sont envoyées par le robot, ce processus envoie le message WM_COMM_RXCHAR à l'application qui exécute une routine de traitement en réponse à ce message. C'est cette dernière qui détermine l'action à prendre lors de la réception des données. Les données susceptibles d'être reçues sur le lien série, ainsi que les actions à effectuer sont résumées dans le tableau suivant :

Information reçue	Action à effectuer
Détection d'un obstacle	Mise à jour de la matrice et affichage
Délai entre les impulsions sonores	Validation des délais ; Calcul de la position et du chemin optimum ; Affichage du robot et du nouveau chemin optimum ;

Détection d'une collision	Mise à jour de la matrice d'obstacle ; Calcul du nouveau chemin optimum ;

Pour ce qui est de la transmission, les données sont placées sur le lien série par une fonction WriteToPort qui reçoit en paramètres les données à transmettre.

Le protocole de communication utilise l'octet comme unité de base. Les données sont transmises directement, sans aucun protocole de communication. Ce point serait éventuellement à améliorer, mais l'échéancier ne nous permet pas un tel ajustement et la communication a toujours été efficace de cette façon.

1.6 Traitement de la Position du Robot

Pour trouver la position du robot, nous avions pensé à deux approches. La première était de calculer la position en résolvant des équations et l'autre était d'utiliser un outil intelligent. Ce dernier est en fait un réseau de neurones qui, à partir des trois délais entre les impulsions sonores, fourni la position en 2 dimensions. Dans les deux cas, le robot envoi 4 valeurs de compteurs à l'ordinateur à partir desquels les délais sont calculés sur l'ordinateur pour qu'il puisse faire le traitement de la position. Le plan de la salle avec les haut-parleurs est montré à la figure 1.3. La stratégie qui à été adopté est celle du réseau de neurones, pour sa simplicité et sa précision.

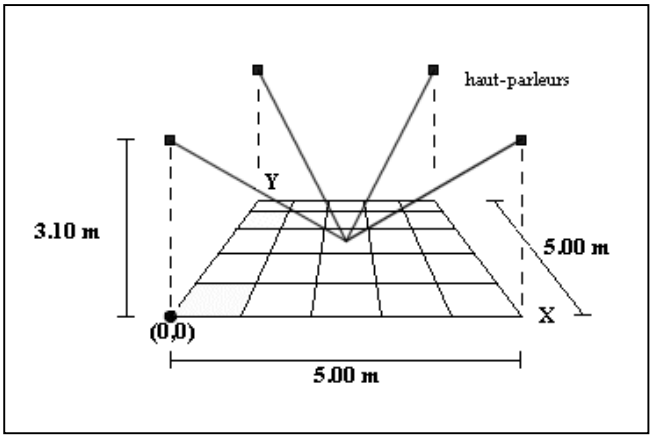

Figure 1.3 - Plan de la salle avec les haut-parleurs

1.7 Positionnement par réseau de neurones

L'utilisation du réseau de neurones, connu sous le nom de «Perceptron», rend très facile le positionnement spatial. La méthode consiste à recueillir une banque d'information relative à la position, soit les trois délais entre les impulsions sonores, et la position réelle dans l'espace. Comme on sait que le robot se promène sur une surface plane, sa position est ainsi réduite à une coordonnée cartésienne à deux dimensions. Une fois que le nombre de données amassé est suffisamment grand, on entraîne le perceptron.

L'entraînement du perceptron consiste à lui donner en entrées les 3 délais et en sortie la position cartésienne. Grâce à une fonction de rétropropagation de l'erreur, le réseau est ajusté pour obtenir une correspondance entre les entrées et les sorties. En lui fournissant ainsi une quantité suffisante d'informations, on ajuste les paramètres du réseau et on peut par la suite l'interroger. Nous lui donnons les 3 délais en entrées et il nous fournit la position cartésienne en sortie.

Revenons maintenant sur la façon dont on a recueilli l'information nécessaire à l'entraînement du réseau. Premièrement, nous avions besoin de plusieurs mesures. Pour effectuer celles-ci, nous avons positionné notre robot à 36 endroits équidistants sur le terrain et demandé au robot de mesurer 3 délais. En fait, le robot envoi 4 valeurs de compteur à partir des quels, l'ordinateur trouve les 3 délais, tel que décrit précédemment. Ces trois délais sont facilement mesurables par le robot, car nous avons quatre haut-parleurs. Le UBIP envoie des séquences de 4 impulsions (une dans chaque haut-parleur) et chaque séquence est séparée par un temps plus long que le temps qui sépare les impulsions. Ainsi, le robot peut facilement savoir lorsqu'une nouvelle séquence débute et ainsi mesurer les trois délais. Nous ne transformons pas ces délais en distance car nous utilisons directement les délais pour entraîner notre réseau de neurones, tel qu'illustré sur la figure 1.4.

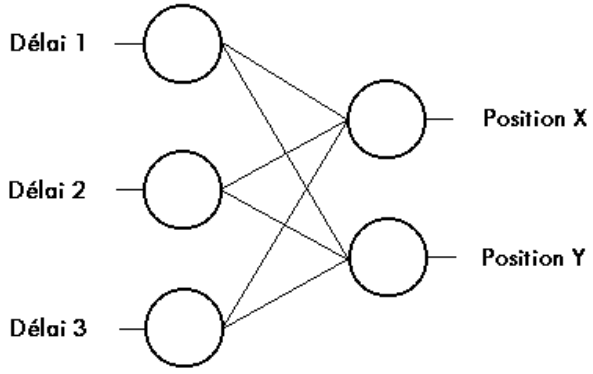

Figure 1.4 - Réseau de neurone

La figure précédente montre une vue simplifiée du réseau de neurones, puisque nous avons utilisé un réseau comportant quatre couches (soit deux couches cachées, une couche d'entrée et une couche de sortie) à raison de 10 neurones par couches cachées. Avec les données utilisées pour l'entraînement, le perceptron a réussi à modéliser une erreur de moins de 2% erreur soit une erreur réel de plus ou moins 10 cm. Une fois bien entraîné, le perceptron est en mesure de nous fournir d'une façon précise et rapide la position du robot. Par contre, cette précision est tout de même relative et nous avons prévu un mécanisme de correction des erreurs. En effet, dans le cas ou la position calculée est irréaliste vue la vitesse de déplacement du robot, nous comparons la nouvelle position avec l'ancienne pour voir si le déplacement est réaliste. Dans le cas contraire, la nouvelle position est tout simplement ignorée et on attend l'envoie de la prochaine position pour modifier le parcours du robot.

1.8 Calcul du chemin optimum

Avant de diriger le robot, il faut savoir quel chemin on veut lui faire suivre. C'est pourquoi nous avons fait une fonction qui calcul le chemin le plus court entre le robot et sa trajectoire. Cette fonction utilise la matrice d'obstacles, la position du robot et la position d'arrivée. Avec ces arguments, elle détermine le chemin optimum et retourne à l'algorithme qui trouve la vitesse de chaque roue (voir plus loin)la direction future, et le trajet complet calculé est ainsi envoyé à l'interface usager. La figure 1.5 montre le résultat de l'algorithme du chemin optimum. Il est facile de voir que l'algorithme fonctionne très bien, et ce, dans des temps plus que raisonnables, ce qui nous permettra d'avoir un affichage en temps réel de la position du robot. Nous allons maintenant voir plus en détails le fonctionnement de l'algorithme.

Nous avons utilisé une dérivée de l'algorithme de Dijkstra. Comme vous avez dû le remarquer sur la figure 1.5, le robot ne tourne qu'à 45 degrés. La raison est que de cette manière, il est plus facile de faire les calculs du chemin optimum et il sera plus facile de diriger le robot. Nous avons donc défini 8 directions possibles (voir figure 1.6). Pour trouver le chemin optimum, nous vérifions tous les chemins possibles entre le robot et la destination et nous choisissons le chemin le plus court. En fait, l'algorithme arrête de calculer les chemins aussitôt qu'il en trouve un qui se rend à l'arrivée, car grâce au principe de l'algorithme de Dijkstra, nous pouvons être certains que c'est le chemin le plus court.

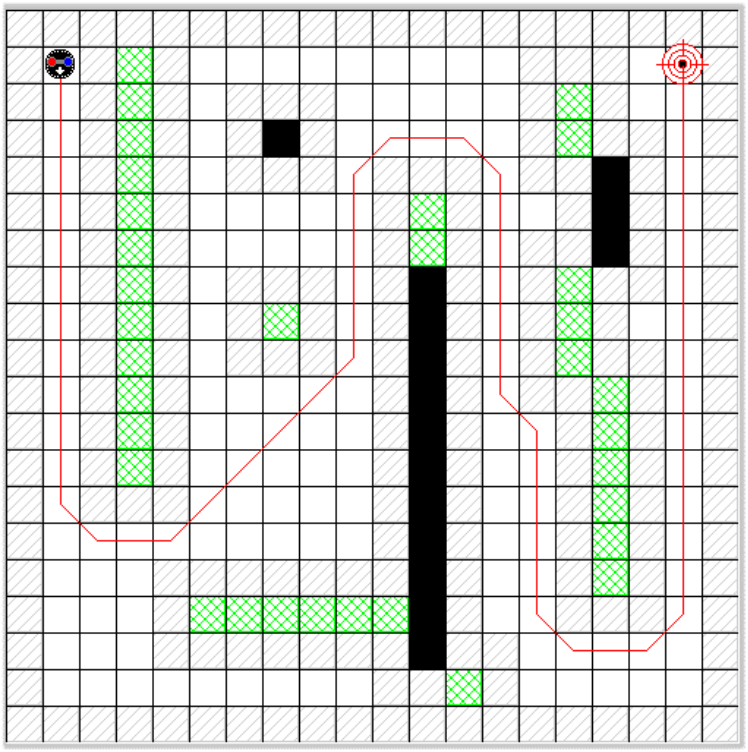

Figure 1.5 - Résultat de l'algorithme du chemin optimal

Pour nous aider à calculer la distance, nous avons associé à un déplacement horizontal et vertical une valeur de 1 et à un déplacement en diagonal une valeur de 1.4 (les déplacements diagonaux sont plus longs).

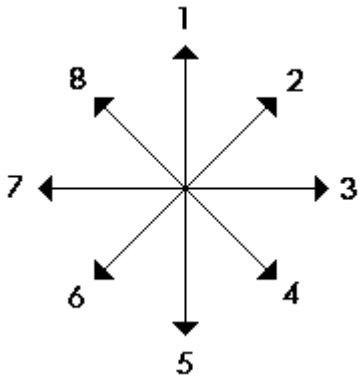

Figure 1.6 - Code des directions

L'implantation de la dérivée de l'algorithme de Dijkstra est assez simple. Nous utilisons une matrice de chemins de même dimension que la matrice d'obstacles pour garder les chemins déjà trouvés. De plus, nous utilisons une structure qui comprend la position en X, la position en Y, ainsi qu'une variable Distance. Nous commençons par le carreau de la matrice où se trouve le robot. On regarde toutes les directions autours du carreau. Lorsque, pour une direction

donnée, il n'y a pas d'obstacle, nous insérons dans la matrice de chemin, cette direction. De plus, nous plaçons dans notre structure la position en X et la position en Y, ainsi que la distance de 1 ou 1.4. Ensuite, on copie les éléments de cette structure dans un monceau. Un monceau est une structure de donnée en arbre qui met toujours l'élément le plus petit à la racine, soit la distance dans notre cas.

Après avoir terminé le traitement avec le premier carreau, on prend un élément du monceau, qui est en fait une de nos structures. On a donc la position d'un carreau. Pour ce carreau, on vérifie tous les carreaux autours de celui-ci. Pour chaque carreau, on vérifie si celui-ci n'est pas occupé par un obstacle, et on vérifie aussi si le chemin pour se rendre à ce carreau n'a pas déjà été trouvé en regardant dans la matrice de chemins. Si ce n'est pas le cas, on indique la direction dans la matrice de chemins. On met la position de ce carreau dans une nouvelle structure et on additionne 1 ou 1.4 (dépendant de la direction) à la variable Distance du carreau d'avant. On met ensuite, cette structure dans le monceau.

On répète ceci jusqu'à ce que l'on arrive au point d'arrivé ou jusqu'à ce que le monceau soit vide. Si le monceau est vide, c'est qu'il n'y a pas de chemin possible. Si l'on arrive au point d'arrivée, nous pouvons être certains que le chemin est le plus court possible, car le monceau retourne toujours la distance minimum pour arriver à un carreau.

Ensuite, pour trouver le chemin à suivre, nous n'avons qu'à commencer à partir de la position d'arrivée dans la matrice de chemins et suivre la direction inverse qu'indique le code de direction.

1.9 Direction du Robot

Maintenant que nous connaissons le chemin que doit suivre le robot, il ne nous reste simplement qu'à demander au robot de suivre ce chemin. Nous avions pensé à plusieurs solutions pour remédier à ce problème. L'une d'elles est de faire une fonction sur le robot qui demanderait à celui-ci de tourner d'exactement 45 degrés et d'avancer de 1 ou 1.4 selon la direction. Ceci implique que le robot doit être extrêmement précis et il n'y a pas de place pour des perturbations extérieures. De plus, cela implique aussi que le robot doit arrêter pour tourner.

Une autre solution, celle que nous avons adoptée, est de calculer dynamiquement la direction du robot au fur et à mesure que celui-ci se déplace. En effet, nous recalculons la position du robot ainsi que sa direction à chaque fois qu'il change de carreau dans la matrice. Donc, le robot va pouvoir effectuer

les modifications nécessaires s'il y a des perturbations extérieures qui le font dévier de sa trajectoire (ex : une roue qui tourne dans le vide).

La figure 1.7 montre le diagramme du fonctionnement de l'algorithme qui s'occupe de diriger le robot.

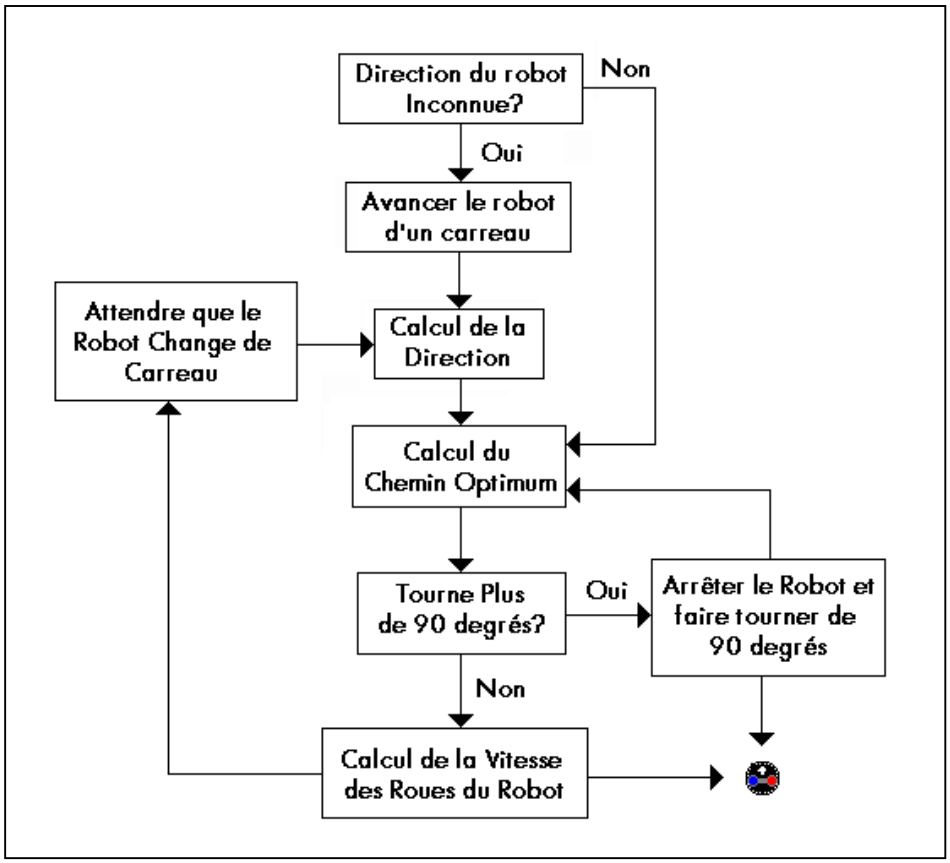

Figure 1.7 - Diagramme de l'algorithme qui dirige le Robot

Au début, si on ne connaît pas la direction du robot, on fait avancer le robot de 1 carreau dans la matrice. Ainsi, on peut connaître la direction du robot. Ensuite on calcule le chemin optimum. S'il faut faire tourner le robot de plus de 90 degrés, nous arrêtons le robot et nous le faisons tourner de 90, -90, 135, -135 ou 180 degrés. Cependant, lorsque l'angle est inférieur à 90 degrés, nous ne voulons pas que le robot s'arrête pour tourner. Pour ce faire, lorsque l'on voudra tourner à gauche, nous mettrons la vitesse de la roue gauche à 1 et l'autre à 2. Ainsi, le robot va tourner sans arrêter. De plus, lorsqu'il va changer de carreau dans la matrice, on pourra détecter si le robot est dans la bonne direction. S'il l'est, nous n'aurons qu'à mettre les deux roues à la même vitesse pour qu'il avance droit. Ainsi, en jouant avec la grandeur des carreaux de la matrice, nous

pensons pouvoir faire une sorte d'asservissement qui permettra au robot d'avancer sans trop osciller.

1.10 Interface usager

L'interface usager, tel que nous l'illustre la figure 1.9, possède plusieurs caractéristiques :

- Affichage en temps réel de la position du robot, de sa destination et du chemin qu'il doit suivre ;
- Affichage de la position du contrevenant ;
- Affichage de l'information reçue et transmise sur le port série ;
- Affichage de la tension des batteries ;
- Affichage de la vitesse de chaque roue du robot.

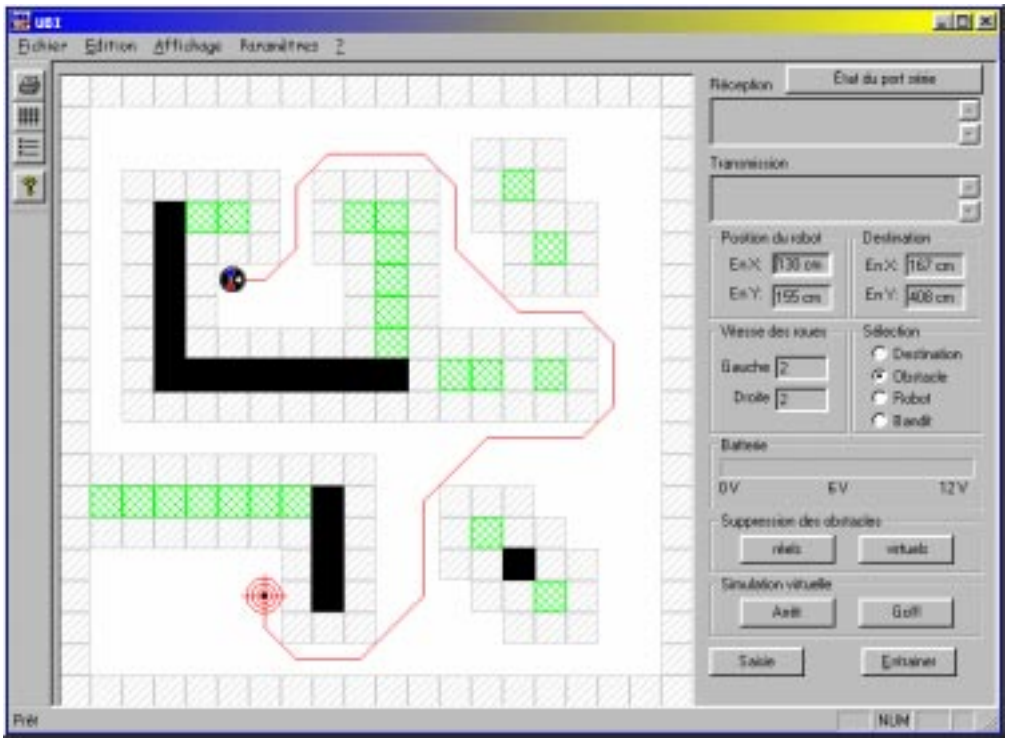

Figure 1.9 - Interface usager

Pour ce qui est de la direction du robot, on peut la repérer facilement à l'aide de la flèche situé sur le robot virtuel de l'interface. Le contrevenant est représenté par une petite voiture sport rouge (figure 1.10).

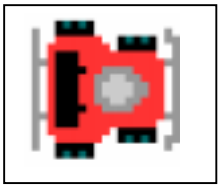

Figure 1.10 - Contrevenant

Les différents contrôles qu'on retrouve sur l'interface nous permettent d'effectuer plusieurs tâches :

- Sélection de la position virtuelle du robot (pour fin de test) ;
- Sélection de la destination ;
- Sélection de la position du contrevenant ;
- Démarrage/arrêt de la simulation virtuelle ;
- Démarrage/ arrêt de la simulation en temps réels ;
- Vérification de l'état du port série ;
- Suppression des obstacles virtuels/réels de l'environnement du robot.

Tous ces contrôles servent évidemment à configurer les différentes simulations. Lorsqu'une de celle-ci est amorcée, on peut voir le robot suivre le chemin tracé, et celui ci se recalcule à chaque fois que le robot se déplace, permettant ainsi une adaptation dynamique du robot dans sa trajectoire.

Il est également possible de configurer le port série grâce à une boite de dialogue (voir figure 1.2) qui affiche les paramètres, tel que mentionné dans la section Communication série.

1.11 Assemblage de tous les modules

Nous avons mit la communication série, l'affichage de l'interface ainsi que l'algorithme qui trouve la vitesse des roues dans des processus(thread) séparés. De plus, nous avons mit des sémaphores pour les variables qui sont partagés par les processus afin que nous n'ayons pas de problèmes.

Comme vous avez pu le constater, ce groupe à construit l'interface et les algorithmes en supposant que les autres parties du projet marcheraient plus ou moins. En effet, l'interface permet de mettre des obstacles virtuels au cas où la détection d'obstacle ne marcherait pas. Les grosseurs des carreaux de la matrice peuvent varier selon la précision du positionnement. Une vérification du positionnement est faite par l'interface pour s'assurer que la position du robot est réaliste. L'orientation du robot est vérifiée à chaque fois qu'il change de carreau

dans la matrice. Le trajet optimal est recalculé périodiquement pour contrer les influences extérieures qui pourrait faire dévier le robot. Ce n'est pas par manque de confiance auprès des autres groupes que nous avons agit ainsi, mais pour facilité l'intégration des différentes partie.

1.12 Résultats

Malheureusement, les résultats obtenus sont loin d'être ceux escomptés. Le robot à de grandes difficultés à suivre le chemin optimum. Cela s'explique par le fait que le délai entre l'envoie des vitesses et l'exécution de la commande par le robot est trop long. Il faut environ deux secondes pour le robot pour trouver sa position. Durant ce temps, il avance de beaucoup et lorsqu'on lui envoi une commande, elle est pour ainsi dire, désuète. Donc, le robot tourne soit trop loin, soit trop longtemps et il tourne donc autour de l'obstacle. Cependant, il fini souvent par atteindre son but, ce qui est le principal.

De plus, nous avions de la difficulté à faire tourner le robot sur lui-même. On voulait une fonction qui fait tourner le robot d'environ 180 degrés mais le robot n'était vraiment pas assez précis (les encodeurs ne marchaient pas bien) et il ne tournait jamais du même angle.

Le temps est le plus grand facteur qui a joué contre nous. Notre algorithme du chemin optimum fonctionne très bien et le positionnement aussi. Avoir eu plus de temps, nous aurions pu améliorer de beaucoup la trajectoire suivit par le robot. Nous aurions pu aussi inclure l'odométrie pour contrer l'effet du délai dans le calcul de la position par le robot.

1.13 Conclusion

Nous pouvons dire que nous avons rempli notre mandat même si nous n'avons pas atteint notre objectif qui était de conduire le robot à un point **en suivant le chemin optimum**. Les résultats sont quelques peu décevant, mais l'expérience acquise durant ce projet compense grandement cette lacune. Il faut toutefois préciser que le robot se rend quand même au point de rendez-vous, ce qui était une exigence du projet.

Pour ce qui est des autres exigences, tels que la détection des collisions et la détection des obstacles, les résultats sont très appréciables. Le robot détecte bien les obstacles et lors d'une collision, il passe en marche arrière, recule d'une courte distance et peut ainsi continuer son chemin en évitant l'obstacle détecté. L'affichage de la tension des batteries n'a pas été réalisé, mais l'interface dispose d'un mécanisme prévu à cet effet.

Le projet en est donc encore à l'étape du prototype. Plusieurs possibilités sont envisagées pour remédier aux divers problèmes mentionnés précédemment, mais l'échéancier ne nous permet pas de continuer le développement.

Néanmoins, le logiciel d'application est disponible dans la section téléchargement, et est totalement opérationnel. Adressez vos commentaires, questions ou autres, à un des deux responsables de la section Interface usager :

Chapitre II

Contrôle du robot

Chapitre 2 Chapitre II : Contrôle du robot

2.1 Introduction

Cette partie du développement consiste à relever des données de position , les transmettre à un ordinateur hôte afin d'exécuter une commande motrice calculé par celui-ci . Afin de ne pas surcharger les exécutions du robot , la majorité des traitements sont fait sur l'ordinateur . Vous trouverez ,dans ce compte-rendu ,le schéma de fonctionnement du robot ainsi que le cheminement parcouru pour atteindre nos objectifs.

2.2 Algorithme implanter au robot

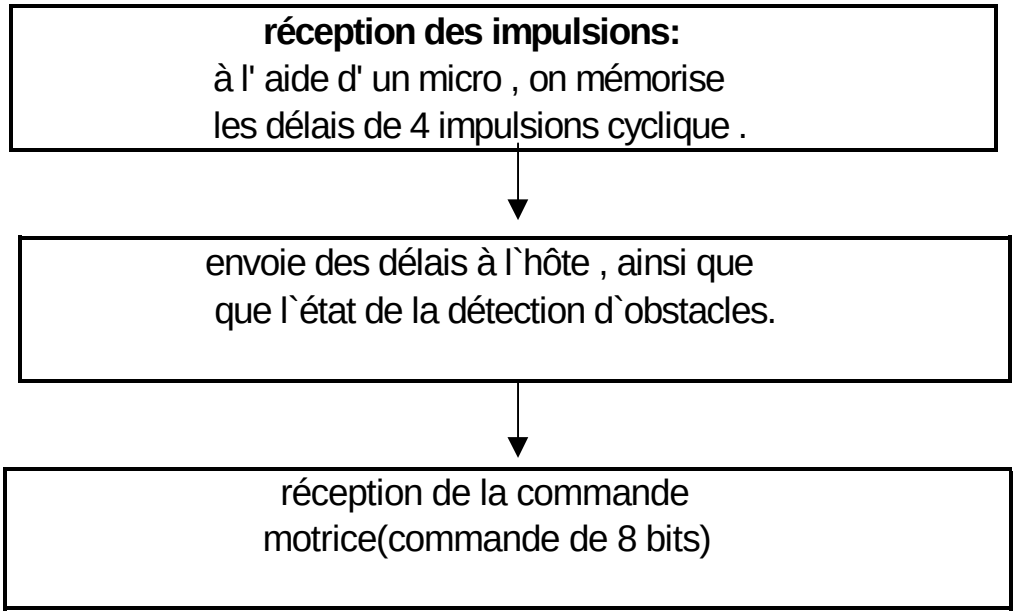

Figure 2.1 : plan du progamme

2.3 Initialisation

L'initialisation contient principalement la déclaration des variables utilisées pour les commandes et la mise en fonction de l'input-capture2 servant pour la détection acoustique(voir section suivante). De plus , il est important de désactiver le `pcode` utilisé par interactive C. Cette désactivation permet d'avoir libre accès au port série du microcontrôleur et ainsi pouvoir établir un lien avec l'ordinateur hôte .

2.4 Réception des impulsions(détection acoustique)

2.4.1 Générateur d'impulsions

Afin de positionner le robot , on utilise un cycle de 4 impulsions dont les pulses sont décalées de 150 millisecondes .À raison d'un cycle par seconde , nous avons établie une séquence facile à synchroniser et permettant d'éliminer l'effet des échos . Le générateur produit des impulsions à 10 kHz facilement détectable avec un micro bon marché .

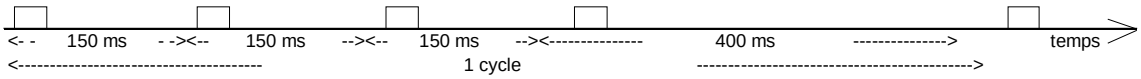

figure 2 : schéma des impulsions vs le temps

2.4.2 Micro de réception

Le micro utilisé est filtré par un filtre passe bande autour de 10 kHz , fréquence évitant de détecter les bruits ambiant . Nous avons ajusté nos gains afin de capter des signaux relativement fort par rapport aux échos pour une meilleur fiabilité .

2.4.3 principe de la réception

Pour savoir notre position , nous devons calculer les délais entre chacune de nos impulsions . Pour ce faire , on utilise la fonction input-capture du 68HC11 permettant de détecter des pulses et de savoir le moment exacte de leurs arrivés. En effet, la particularité de cette fonction est de mémoriser le temps de son registre compteur. Ainsi, lorsqu'une pulse est détecter, son temps est automatiquement insérer à l'adresse du input-capture. Le registre compteur est de 16 bits avec une vitesse d'horloge de 2 Mhz, ce qui représente un débordement à toute les 32.7 ms.

Au commencement , nous comptions le nombre débordement mais nous nous sommes rendu compte qu'on surchargeait le contrôleur, ce qui avait pour conséquence de nous donner une précision de 1200 coups d'horloge : environ 0.6 ms d'incertitude .

Puisque nous savons que l'on a environ 4-5 débordement entre chaque pulse , nous pouvons facilement utilisé la donnée la plus probable . Cette procédure nous a permis de réduire l'incertitude à 200 coups d'horloge (0.1ms) .

Pour synchroniser notre système , nous utilisons des fonctions définis dans interactive C . Il existe des fonctions permettant de calculer un temps en millisecondes . Il suffit simplement de remettre à zéro un compteur et de faire appel à la fonction mseconds() , fonction retournant un float indiquant un temps en ms . Ainsi , on identifie notre première pulse en vérifiant si nous avons eu un délai plus grand que 300 ms entre deux pulses . Ensuite , on fait une série de input-capture pour mémoriser le temps entre chaque pulse . On insère une attente de 100ms entre chaque détection pour éliminer les réflexions immédiate . Après cette attente , on efface le drapeau du input capture pour recevoir la pulse suivante et ainsi de suite.

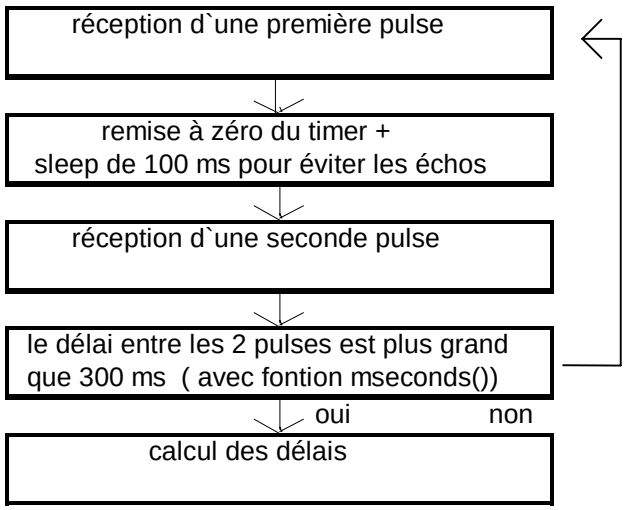

figure 2.3 : algorithme de synchronisation

2.5 envoie des données à l'ordinateur hôte

2.5.1 Préalable

Lors de l'initialisation du processus , on désactive le pcode utilisé par interactive C afin d'avoir le contrôle sur le port série . Nous ne changeons pas la configuration du port, c'est-à-dire qu'on communique à 9600 bauds , un stop bit et un start bit.

2.5.2 Communication

Après avoir reçu les quatre impulsions, on transmet les données retenues par le input-capture et on envoie l'état des pare-chocs. Ainsi, l'hôte peut calculer les délais entre chaque pulse et peut détecter s'il y a des obstacles devant le robot. On n'utilise pas de protocole spéciale et on ne vérifie pas si la donnée a bien été reçu, car on se contente d'une procédure transmission/réception simple parce qu'on suppose que la transmission est sans faille. Ce qui est important de remarquer, c'est que le registre compteur du 68hc11 est de type signé, ainsi le compteur contient une valeur comprise entre 0 à 32 768 et passe ensuite de -32 767 à -1 . C'est alors qu'il se produit le débordement. Nous avons perdu du temps sur la compréhension du compteur et sur la façon de traiter la donnée. En effet, après une multitude de tests , nous avons découvert que le contrôleur ne pouvait pas traiter rapidement nos calculs . C'est qu'il fallait faire des calculs sur 32 bits, ce qui représente une tâche trop longue pour le robot. Nous avons réglé le problème en laissant calculer les délais à l'ordinateur. Ainsi, on a simplement quatre états de compteur(16 bits) à envoyer à l'hôte ; ce qui réduit la quantité d'information à transmettre. Il aurait fallu transmettre trois délais de 32 bits comparativement à quatre états de compteur 16 bits présentement utilisé.

2.6 réception de la commande

2.6.1 fonctionnement

La commande reçue au robot est codée sur huit bits. Nous avons un bit pour la vérification de la batterie, trois bits pour les commandes de rotation et deux bits de vitesse des roues pour chacune des roues.

batterie	rotation	rotation	rotation	vitesseG	vitesseG	vitesse D	vitesseD
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

Les problèmes que nous avons eu avec les encodeurs des roues faisant en sorte que nous n'avions aucune précision dans la rotation. Ainsi, nous n'utilisons pas beaucoup ces commandes, mais ajustons plutôt la vitesse des roues pour effectuer une rotation. C'est plus lent mais nous avons préféré cette méthode pour une meilleur performance à ce stade ci du projet. Il serait question de changer les encodeurs de roues pour la prochaine session pour avoir une meilleure capacité à savoir la distance parcourue par le robot.

2.7 Conclusion

N'ayant pas beaucoup d'informations sur le robot, il a été difficile d'avancer rapidement pour la communication série. Ainsi, une fois cette étape franchit, nous avons rapidement progresser pour avoir ce résultat finale. Avoir réussit plus tôt la communication série , nous aurions probablement un meilleur système puisque nous avons détecté quelques unes des erreurs qui se produisait lors des tests pratiques, notamment au niveau de la vitesse de traitement des données qui est plus lent que prévus. Ce problème amène des modifications majeurs au module.

Chapitre III

Localisation sonore

Chapitre 3 Localisation sonore

3.1 Le générateur d'impulsions

Le générateur d'impulsions contrôle quatre haut-parleurs amplifiés. Il doit générer 4 bips (un par haut-parleur) distancés de 0.15 seconde chacun à chaque cycle. Le délai entre les bips d'un cycle est fixé à 0.15 seconde, ce qui est un compromis entre le temps nécessaire pour annuler les échos et le besoin de prendre la mesure de positionnement aussi souvent possible. Un délai plus long (0.55 seconde) sépare le 4^{e} bip d'un cycle et le 1er bip du cycle suivant pour que le robot puisse se synchroniser, c'est à dire savoir quel bip du cycle il correspond.

Voici un schéma temporel des sorties du générateur:

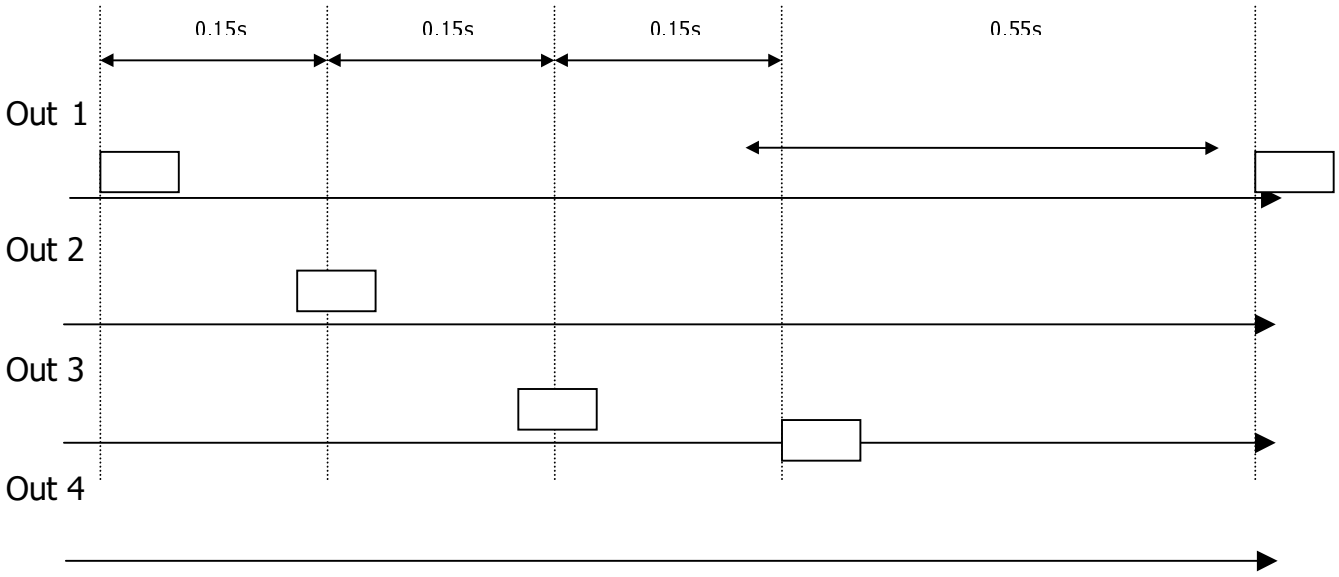

Figure 3.1 : Schéma temporel

Où chaque boîte désigne une série de 50 impulsions carrées de fréquence égale à 10kHz et de rapport cyclique égal à 1. Un microcontrôleur PIC16C54A de Microchip a été utilisé pour générer ces 4 signaux. Un simple filtre RC passe haut a été ajouté à chaque sortie utile du PIC de manière à obtenir un son plus semblables à des impulsions de sinusoïdes 10kHz. Les constantes de temps ont été ajustées en fonction de l'effet obtenu à la sortie des haut-parleurs amplifiés.

Voici un schéma du circuit de génération d'impulsions:

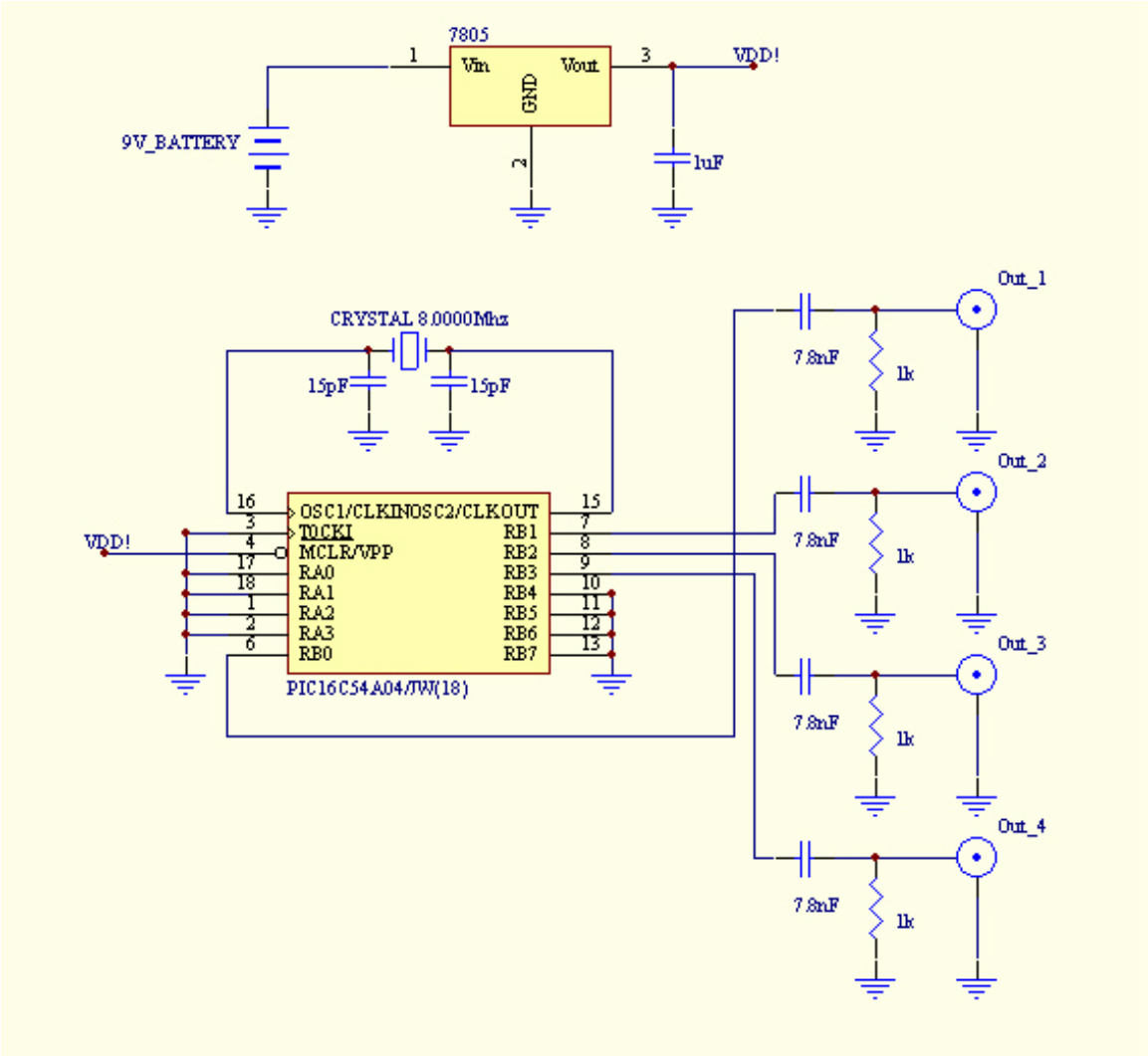

Figure 3.2 : Schéma du générateur de blip

3.2 Module réception acoustique

Le module de réception acoustique à comme fonction de détecter les impulsions produites par le générateur d'impulsions, de les traiter et transmettre l'informations au microcontrôleur La figure 3.3 présente les différentes composantes du module :

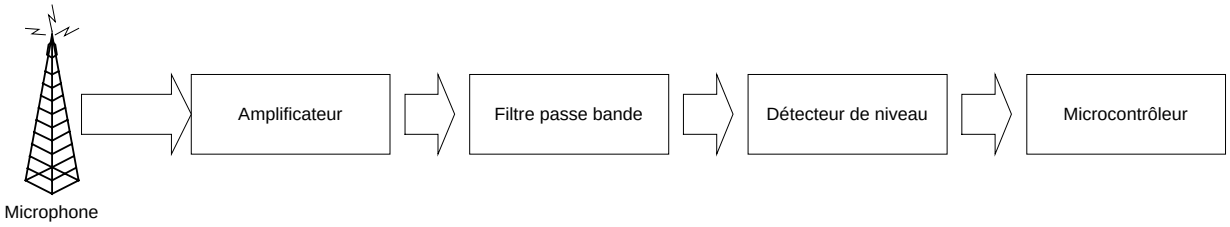

Figure 3.3 : Module de réception acoustique

Il est composée d'un microphone qui sert à la réception du signal. Étant donné que le signal reçu est très faible, un pré amplificateur à été utilisé avant de filtrer. Le filtre est de type passe bande étant donné que les impulsions sont généré à une fréquence précise. Un détecteur de niveau est utilisé pour capter l'information résultantes.

3.2.1 Microphone

Le microphone utilisé à été acheté car celui fourni sur le robot n'était pas très bon. La réponse en fréquence la plus intéressante du microphone est d'environ 10kHz. C'est donc à cette fréquence que le générateur d'impulsions émet ses pulses. Afin d'améliorer la réception du micro et de diminuer les faux écho, une forme conique à été ajouté au bout de celui-ci.

3.2.2 Amplificateur

L'amplificateur à comme fonction d'amplifier le signal provenant du microphone. Le circuit intégré LM386 à été comme pré amplificateur. C'est un circuit très utilisé pour les applications dans le domaine de l'audio. Étant donné que le signal provenant du microphone était très faible, le gain choisi pour l'amplificateur est de 100. La figure suivante montre la réponse en fréquence et la distorsion. Nous pouvons constater que le LM386 possède un gain stable à 10kHz. Par contre, le distorsion n'est pas minimale à cette fréquence mais elle est très acceptable.

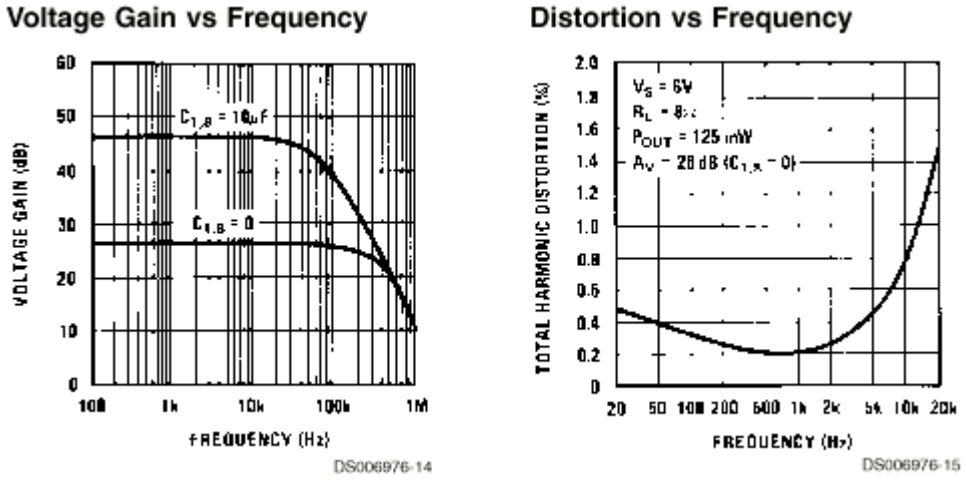

Figure 3.4 : réponse en fréquence du LM386

3.2.3 Filtre passe bande

Le filtre passe bande sert à éliminer les fréquence qui son en dehors de la plage de fréquence voulue. Le filtre utilisé est un MF8 de la compagnie National. La figure 3.5 montre le schéma interne du MF8.

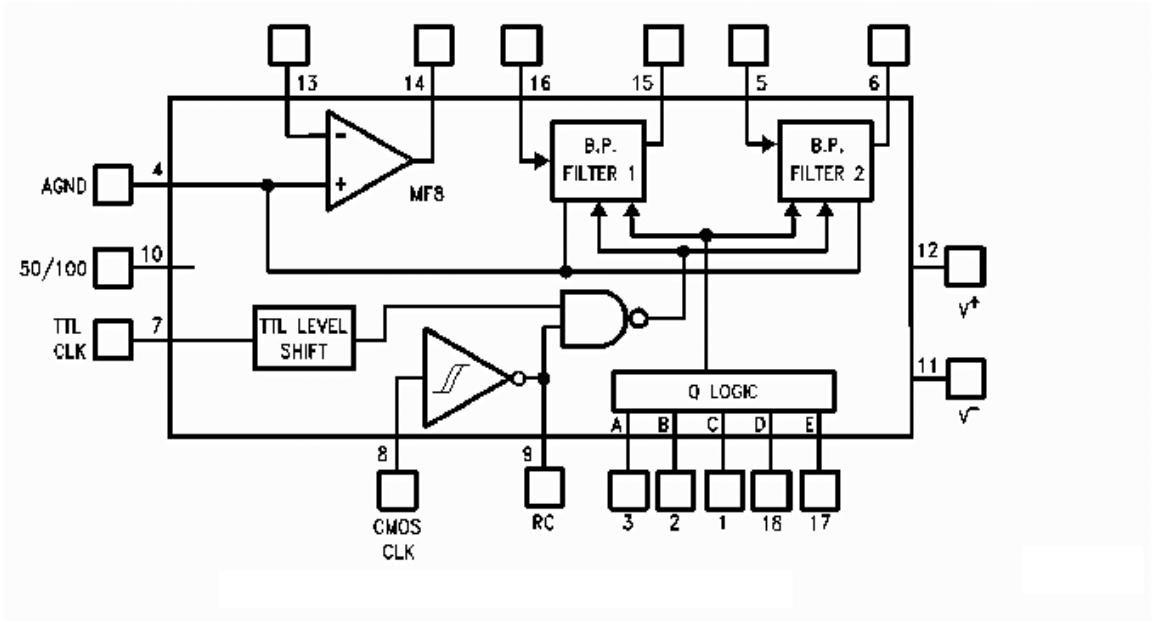

Figure 3.5 : Bloc simplifie du MF8

Ce circuit intégré est composée de 2 étages comprenant chacune un filtre passe bande du second ordre. De plus, il contient un pré ampli que nous avons utilisé aussi mais avec un petit gain de l'ordre de 10. Les entrées « Q logic » permettent de sélectionner le ratio de la fréquence externe avec celui de la fréquence voulu. Ce circuit à été choisi car sont coût est très faible et il offre des performances intéressante. Comme mentionné plus haut, la fréquence des impulsions est de 10kHz. Les étapes de conception du filtre sont les suivantes :

1) Déterminer la bande passante du filtre

La figure 3.6, fournit par le manufacturier, permet de déterminer certain paramètres soit les fréquences de coupures(fc1 et fc2), la bande d'arrêt

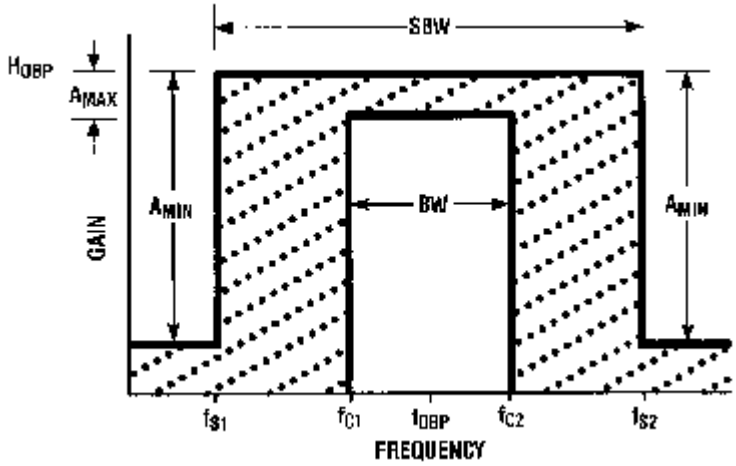

Figure 3.6 : Bande passante du filtre

du filtre(fs1 et fs2) ainsi que la variation du gain dans la bande passante(Amax) et l'atténuation minimal à la bande d'arrêt(Amin). Les paramètres choisi sont les suivant :

Fc1 = 9.5kHz
Fc2 = 10.5kHz
Fs1 = 9kHz
Fs2 = 11kHz
Amax = 1dB
Amin = 20dB

2) Choisir le type de filtre et leur caractéristique

Le choix s'est arrêté sur un filtre de Chebychev. Pour ce filtre, il faur déterminer le « ripple ». Nous avons choisi un « ripple » de 0.1dB et le tableau suivant nous fourni des informations sur des constantes pour le calcul des résistances.

CHEBYSHEV RIPPLE 0.1 dB

Order	K_0	K_2	K_3	K_4	K_5	K_6	K_Q
4	1.6983	2.9512					0.8430
6	1.3183	1.2137	4.5125				1.5473
8	0.7986	0.5782	1.8809	2.0343			2.2176

Figure 3.7 : Constantes de calculs des résistances

3) Choisir l'ordre du filtre

La figure 3.8 permet de déterminer l'ordre du filtre soit de 4. Il suffit de tracer une ligne de A_{max} à A_{min} et prendre la courbe le plus près du rapport SEW/BW.

4) Calculs des résistances

Le calcul se fait comme ceci :

On prend Rf = 100kΩ

K0 = R0 / Rf = 1.6983

R0 = 170kΩ

K2 = R2 / Rf = 2.9512

R2 = 295kΩ

Q = Kq * F0 / BW = 8.43

D'après la table du manufacturier, les codes à l'entrée sont les suivants :

ABCDE = 1110

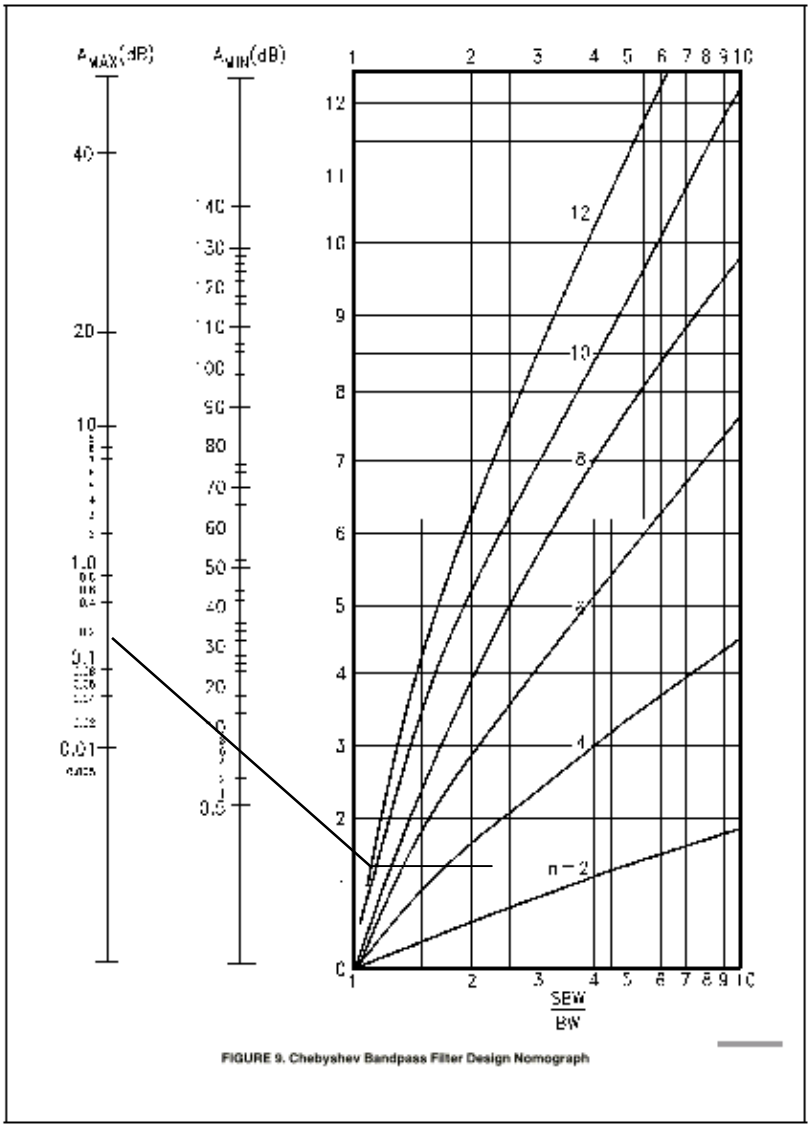

Figure 3.8 : Ordre du filtre

Afin de valider le comportement du filtre, nous avons utilisé un générateur d'onde et les résultats se sont avéré très concluant.

3.2.4 Détecteur de niveau

Le détecteur de niveau est un comparateur. Celui-ci compare le signal d'entrée avec une tension de référence de 5 volts. À sa sortie, il reste seulement une impulsion à 10kHz

Chapitre IV

Détection d’obstacles

Chapitre 4 Détection d'obstacle

4.1 But

Ce département à pour but de faire la conception d'un système de détection d'objets à l'aide d'un système ultrason et/ou avec des senseurs infrarouge. Dans le cas du système à ultrason, celui-ci sera en mesure de fournir la distance approximative d'un objet situé près de lui. Cette information sera transmise du robot à l'ordinateur pour fin de traitement. Le système infrarouge permettra de détecter la présence d'un objet dans un rayon de 30cm. De plus, il sera en mesure de fournir la position approximative de l'objet devant le robot.

Détection d'objets à l'aide d'un module Ultrason

4.2 Matériels

Le système est composée d'un module de la compagnie POLAROID. La figure 4.1 montre le module et ses différentes composantes

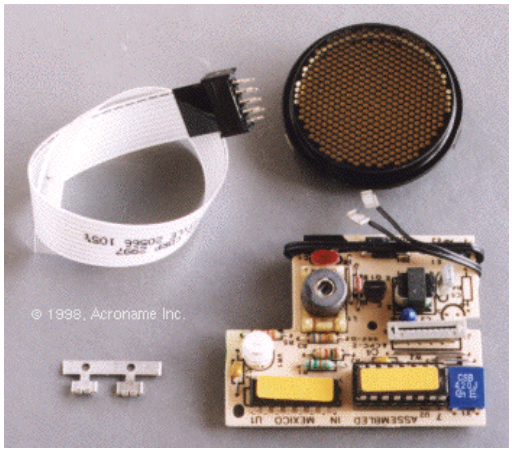

Figure 4.1 : Composantes du Polaroid 6500

Le module est composée d'un système électronique permettant de contrôler les signaux de transmission, d'un câble de connexion et d'un transducer. Le module ultrasonique transmet une onde sonore et mesure le temps que prend l'onde pour revenir. Le temps de réflexion est proportionnelle à la distance de l'objet à la source. Dans cette conception, l'onde sonore est transmise et reçu sur le même transducer. Pour éliminer le faux écho, un temps mort interne est nécessaire. Le contrôle s'effectue à l'aide de 2 signaux. Ces signaux sont les suivants :

INIT : Initialisation du ranging module
BINH : Signal permettant la reception de multiple echo

Un signal de réception permet de déterminer la fin de la séquence. Ce signal est le suivant :

ECHO : Signal indiquant la fin de la séquence

La figure 4.2 montre le fonctionnement détaillé de la séquence des signaux:

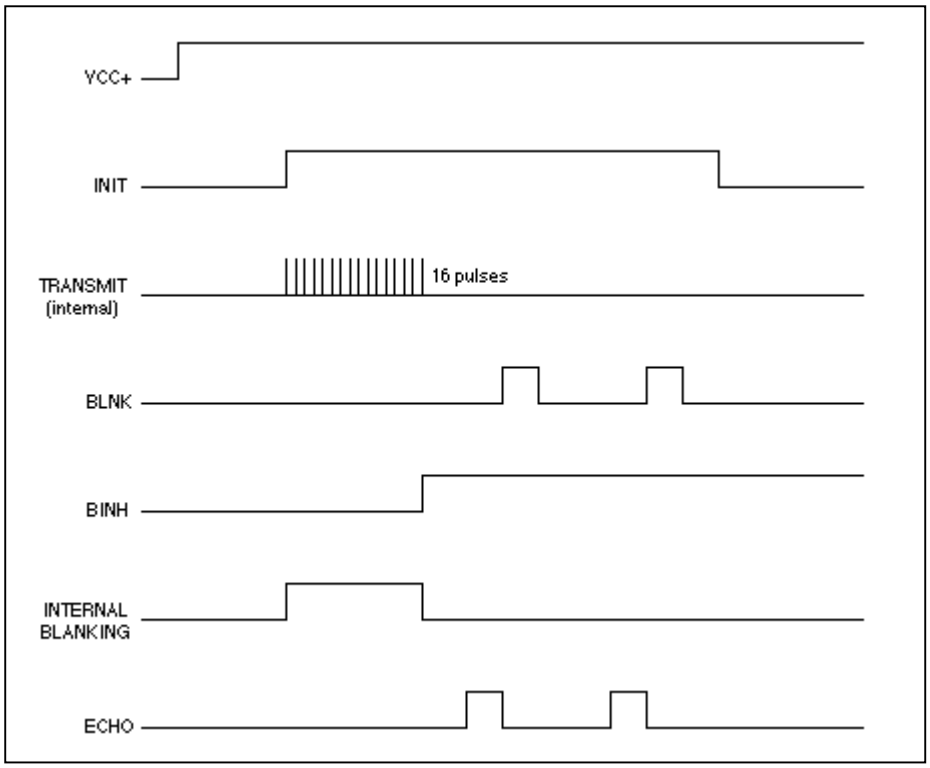

Figure 4.2 : Schéma temporel des signaux

Tout d'abord, le signal INIT est activé. En même temps, un compteur est démarré dans le microcontrôleur pour calculer le temps écoulé entre l'initialisation et la réception de l'écho. Après, le module transmet 16 impulsions à 49.4 KHz. Tant que BINH n'est pas activé, l'ECHO n'est pas pris en compte par le système. De plus, on peut constater que le signal "INTERNAL BLANKING", qui dure 2.38ms, implique que le système ne peut détecter un objet à moins de 13 pouces ceci dans le but d'éviter que le module interprète les impulsions comme un signal d'écho. Pour trouver la distance, il suffit d'appliquer la formule suivante :

Distance(cm) = TimeEcho * 7 / 200 + 22

4.3 Considération de design

Comme lors de la transmission le module requiert une tension de 400V pointe à pointe et un courant de 2A pendant la période de transmission. Il est nécessaire d'ajouter un condensateur de stockage entre les lignes d'alimentations du module. Un condensateur de valeur plus grande que 470 microfarads est recommandé. De plus, une résistance de 4.7 KΩ est utilisée pour limiter le courant à l'entrée du microcontrôleur. La figure suivante (4.3) montrent le schéma de branchement du condensateur:

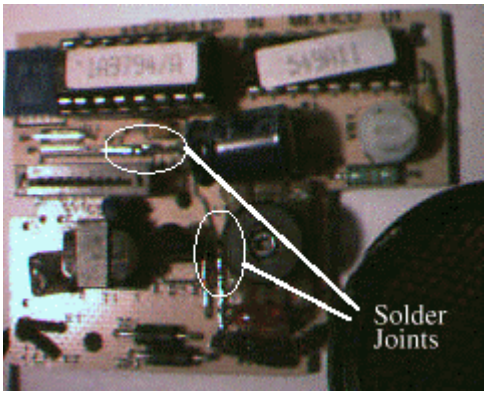

Figure 4.3 : Position du condensteur

4.4 Interface avec le robot

Ce module est branché avec le microcontrôleur du robot. On utilise 2 bits du port D pour les signaux INIT et BINH. Un bit du port A est utilisé comme entrée du signal ECHO. La figure 4.4 montre le schéma de branchement :

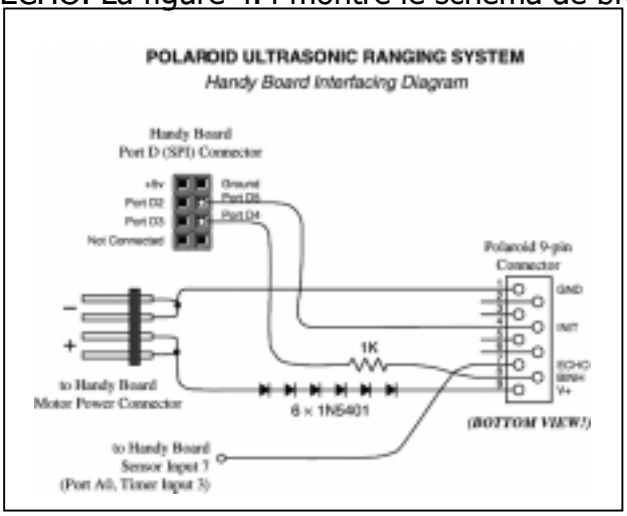

Figure 4.4 :Schéma de branchement

Les 6 diodes sont utilisées seulement pour réduire la tension d'alimentation.

4.5 Tests

Des tests ont été effectués mais par contre ceux-ci n'ont pas été très concluant. Comme nous utilisons le port A0 du microcontrôleur et que celui-ci est déjà utilisé par les senseurs infrarouges, un conflit possible existe. Donc il nous sera impossible d'utiliser les deux en même temps. Étant donné que les senseurs infrarouges fonctionnent correctement, ceci est une autre alternative de détection. Par contre, ceux-ci offre une portée limite de 30 cm, d'un autre côté, ils permettent de détecter la direction de l'obstacle, soit à gauche ou à droite. Étant donné que les test finaux ont été négatifs, nous avons opté pour seconde alternative d'effectuer la reconnaissance d'objet à l'aide des senseurs infrarouges.

Certaines difficultés ont été rencontrées lors de l'intégration avec le robot.

- Peu de port disponible sur le microcontrôleur
- Possibilité de multiple écho, ce qui complique la détection de la distance réelle puisque l'onde peut rebondir sur plusieurs obstacles avant d'être captée.

4.6 Détection d'objets à l'aide de senseurs infrarouges

Les senseurs qui ont été utilisés sont déjà en place sur le robot. Le système est composé de 2 parties :

- 1 transmetteur infrarouge
- 2 receveurs infrarouge

Des fonctions fourni par IC permettent d'émettre et de recevoir le signal. Il est possible de brancher plus de 4 senseurs en utilisant les positions 4 à 7 du port d'entrée digital.

4.7 Considération de design

Il est possible de choisir entre 2 fréquence pour l'émission et la réception soit 100Hz et 125Hz. Cette option pourrait permettre par exemple la communication entre 2 robots. Le module de réception emploi un phase locked loop pour détecter le signal modulé d'une certaine fréquence. Aussi, lorsque le module des senseurs fonctionnent, ils prennent entre 20% et 30% de temps processeur du 68HC11. Il est donc important d'appeler cette routine le moins souvent possible afin de laisser du temps pour les autres processus.

4.8 Interface avec le robot

L'interface avec le robot se fait à l'aide de la librairie des IR. Cette librairie permet de contrôler la fréquence de transmission et la détection des infrarouges à 2 fréquences. Le programme génère une onde carré interne à la fréquence de réception désiré(PLL). Plusieurs fonctions sont disponibles. En voici les principales :

Transmission :

- ir_transmit_on : Active la tranmission des IR
- ir_transmit_off : Désactive la transmission des IR
- set_ir_transmit_period : Détermine la période de transmission
- set_ir_transmit_frequency : Détermine la fréquence de transmission

Réception :

- ir_receive_on : Active la réception des IR
- ir_receive_off : Désactive la réception des IR
- set_ir_receive_frequency : Détermine la fréquence de réception
- ir_counts : Retourne le nombre d'onde carré consécutive

4.9 Tests

Plusieurs tests ont été effectués. Initialement, nous avons testé les IR seul afin de s'assurer de leur fonctionnement. De plus, ceci à permis de valider les sous routines de transmission et de réception. Il s'est avéré que les IR détectent bien les objets sur une distance de 30cm ou moins. De plus, il est possible de savoir si l'objet se trouve à gauche, à droite ou en face du robot. Par contre, nous avons observé que la surface et l'angle qu'a le robot par rapport à l'objet à de l'influence sur le résultat. Parfois, l'angle est trop grand et aucun signal n'est détecté. Ils ont donc été choisi pour faire la détection d'objets à courte distance.

Aucune difficultés n'a été rencontrées en ce qui concerne la détection d'objets. Par contre, nous avons essayé d'implanter une méthode qui nous permettrait de déterminer la distance de l'objet à l'aide des IR. Or, la fonction ir_counts permet de savoir le nombre d'impulsion depuis la transmission, permettant ainsi de calculer la distance étant donné que nous connaissons la fréquence d'émission et la vitesse de propagation dans l'air. Malheureusement pour une raison inconnu, la fonction ir_counts retourne des valeurs non compatible avec les résultats attendus. Nous avons donc abandonné cette option.

Chapitre V

Communication sans fil

Chapitre 5 Communication sans fil

5.1 Objectif

Nous voulons remplacer le lien câblé sériel entre le PC et le robot par un lien sans fils. La communication entre les deux parties doit s'opérer de la même façon qu'avec le câble par le PC et le robot. Elle doit donc être indépendante du type de canal utilisé pour transmettre les informations.

5.2 Transmission à 900 MHz

Ce type de transmission exige, pour chaque signal échangé entre le PC et le robot, un module de transmission et un module de réception. Dans le lien série câblé, deux signaux sont échangés entre le PC et le robot, soit TX et RX. Deux modules d'émission et deux modules de transmission sont nécessaires pour le projet. Pour empêcher que les liens interfèrent, ils opèrent à une fréquence porteuse différente pour chacun. Le lien du PC au robot utilise une fréquence porteuse de 900MHz, et l'autre lien une fréquence porteuse de 1000MHz. Une modulation d'amplitude est utilisée. La bande de fréquence de 900 MHz a été retenue pour quatre raisons :
L'antenne nécessaire pour transmettre le maximum de puissance est de dimension raisonnable. En effet,

$$c = \lambda f$$

$$\frac{\lambda}{2} = \frac{c}{2f} = \frac{3*10^8}{2*900*10^6} = 8{,}33\,\text{cm}.$$

Les composants pour les modules se trouvent facilement chez le manufacturier Motorola. Les transmissions RF par radio amateur sont permises sur cette bande de fréquence. Le défi de réaliser une transmission à cette fréquence est intéressant.

Le schéma fonctionnel de la figure 1 illustre les parties spécifiques du module de transmission, alors que celui de la figure 2 illustre le module de réception.

5.2.1 Module de transmission

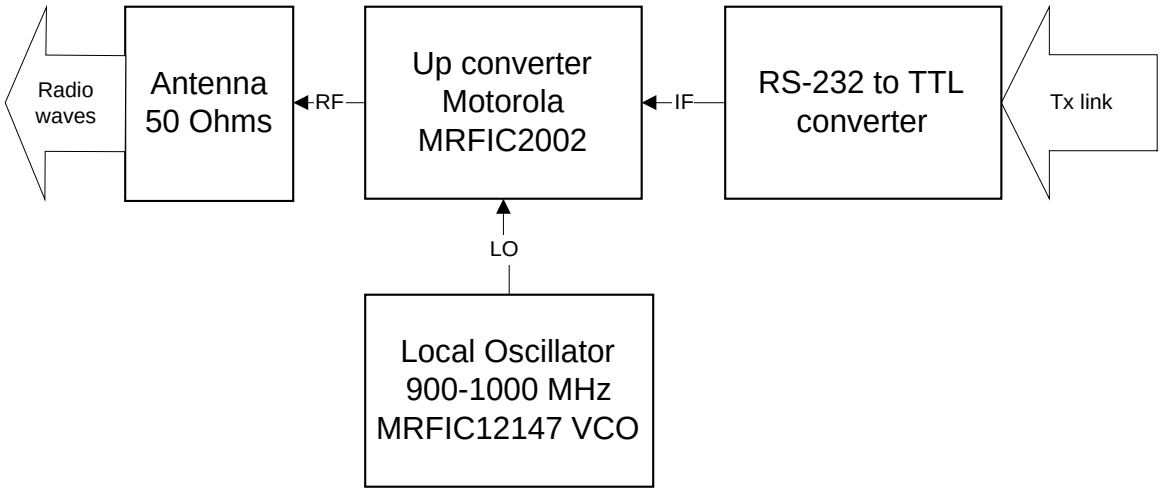

Figure 5.1 - Module de tranmission RF à 900 MHz

Le signal TX est convertit du RS-232 en un niveau TTL pour l'adapter à l'entrée du mixer. Un MAX220 de la compagnie MAXIM est utilisé comme convertisseur. La pièce principale du module de transmission est le "up converter". C'est un mixer qui multiplie les signaux IF et LO. La pièce MRFIC2002 de Motorola fut retenue à cette fin car :

1. La plage de fréquence possible pour le signal IF varie du DC à 250 MHz, permettant de passer une large partie du spectre de notre signal TTL en forme d'onde carrée de 9600 Bds.
2. Le courant d'alimentation est faible, $I_{CC} \approx 5.5$ mA typiquement, économisant la batterie.
3. La tension d'alimentation est $V_{CC} = 5.0$ V, les batteries du robot étant en mesure de le fournir. Cela permet aussi de n'avoir qu'une seule source d'alimentation avec les autres éléments du module de transmission et de réception fonctionnant tous sur 5.0 V.
4. La puissance du signal LO requise est de −10 dBm typiquement, ce qui est compatible avec la puissance fournie par notre LO.
5. Une plaquette de montage avec des traces adaptées en longueur et largeur à une fréquence de 900 MHz pour le signal LO et RF est disponible du manufacturier, facilitant grandement les problèmes de réflexion et de transmission de puissance maximale.

Les fichiers PDF du mixer et de la plaquette de montage indiquent les détails des composants. Comme le mixer accepte une large plage de fréquence pour le signal IF, aucun filtrage n'est fait pour enlever les hautes fréquences pouvant être contenues dans le signal TTL avant de le mixer avec le LO. De cette façon, une large part du contenu spectral du signal TTL est transmis et capté à la

réception. Le filtrage du bruit est fait à la réception. La puissance maximale pour le signal IF et LO est $P_O = P_{LO} = 10$ dBm à l'entrée du mixer.
L'oscillateur local utilisé est un oscillateur contrôlé en tension (VCO). C'est le MC12147 de Motorola qui fut retenu car :

1. Il est conçu pour servir d'oscillateur local.
2. Son courant d'alimentation est faible, $I_{CC} \approx 13.0$ mA.
3. Sa tension d'alimentation est $V_{CC} = 5.0$ V.
4. Sa plage d'opération varie de 100 à 1300 MHz, permettant de réaliser les deux fréquences de transmission.
5. La puissance du signal en sortie varie entre −8 dBm et −2 dBm, dans les normes du up converter et du down mixer.
6. Il sert pour le module de réception aussi.

Le fichier PDF donne les différentes caractéristiques du VCO. C'est un circuit LC dit "tank" qui détermine :

1. La fréquence d'opération selon

$$f_O = \frac{1}{2\pi\sqrt{LC}}.$$

2. La sensibilité de l'ajustement de la fréquence.
3. Le courant d'alimentation tiré de la source.

Les valeurs de L et C sont calculées en tenant compte des capacités et inductances parasites de la plaquette de montage selon

$$C_i = \frac{C_1 * C_V}{C_1 + C_V} + C_P$$

$$C = \frac{C_i * C_b}{C_i + C_b}$$

$$L = L_P + L_T$$

où
C_P est le condensateur parasite de C_1
L_P est l'inductance parasite de L_T
L_T est l'inductance placée sur la plaquette
C_1 est le condensateur de couplage
C_b est le condensateur pour découpler l'alimentation
C_V est le condensateur de la diode.

La figure 2 du fichier PDF du VCO présente un schéma typique d'utilisation du VCO sur lequel les condensateurs et inductances mentionnées sont représentés.

5.2.2 Module de réception

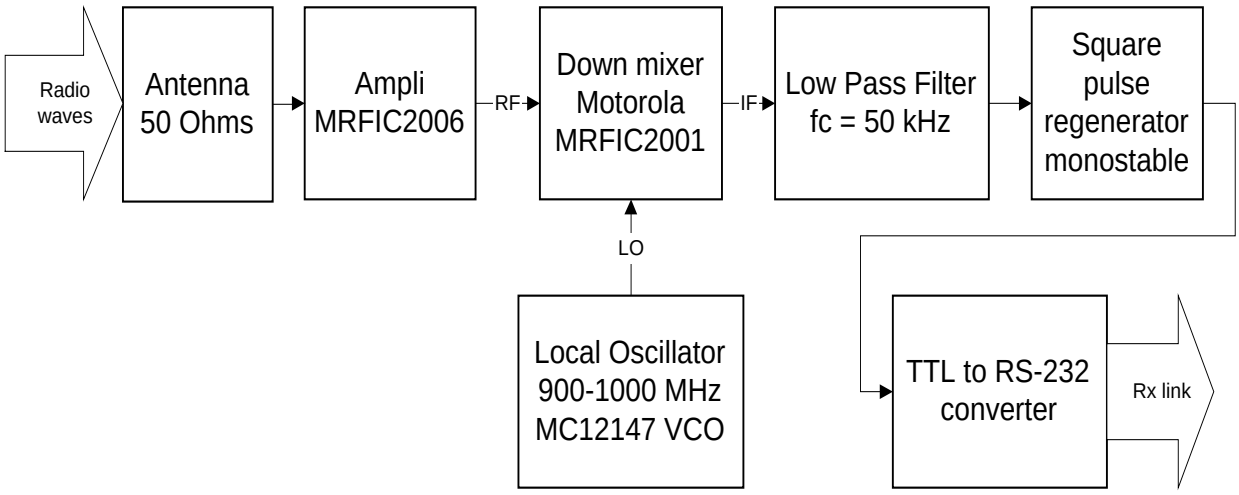

Figure 5.2- Module de réception RF à 900 MHz

Le "down mixer" est la pièce centrale du module de réception. Il multiplie le signal RF reçu de l'antenne par l'oscillateur local. Le signal modulant en amplitude la porteuse opérant à la fréquence de l'oscillateur local est ainsi ramener en bande de base à la sortie IF. Pour améliorer le signal IF, un filtre passe-bande avec une fréquence d'opération centrale de l'ordre de l'oscillateur local et avec une bande de passage de l'ordre du 100 kHz pourrait être placé après l'antenne, mais il est très difficile à réaliser. Cela permettrait de ne récupérer que la porteuse et le signal modulant. Un ampli placé avant le mixer régénère le signal à un niveau acceptable pour le down mixer. Un filtre passe-bas à la sortie IF permet de récupérer que les principaux harmoniques de notre onde carrée. Avec un monostable, l'impulsion carrée TTL est régénérée. Finalement, un convertisseur TTL-RS-232 renvoie le signal dans l'entrée RX du PC ou du robot.

Pour l'ampli, c'est le MRFIC2006 de Motorola qui sert à cette fin car :

1. La pièce est compatible avec le down mixer, fournissant une puissance de sortie d'environ 23 dB à 900 MHz.
2. Sa tension d'alimentation est de 5.0 V.
3. Il est conçu pour opérer dans une plage de 800 MHz à 1 GHz, couvrant les deux fréquences de transmission.
4. Une plaquette de montage est disponible.

C'est la pièce qui consomme le plus de puissance de la batterie, avec un courant d'alimentation de I_{CC} = 46 mA.

C'est le MRFIC2001 de Motorola qui réalise le down converter car :

1. Il opère dans la plage de fréquence de 800 MHz à 1 GHz, couvrant les deux fréquences de transmission utilisées.
2. La plage de fréquence possible pour la sortie IF varie du DC à 250 MHz, permettant de passer une large partie du spectre de notre signal TTL en forme d'onde carrée de 9600 Bds. Il sera plus facile de déclencher le monostable.
3. Sa tension d'alimentation est de V_{CC} = 5.0 V.
4. La puissance requise pour le LO est de P_O = -10 dBm, ce qui est compatible avec l'oscillateur local choisis.
5. Une plaquette de montage est disponible.

5.2.3 Difficultés envisagées

Les plaquettes de montage des modules sont essentielles pour leur réalisation car :

1. Les pièces sont surface mount.
2. Il y aura plusieurs pertes de puissance dues aux réflexions des lignes non adaptées.
3. L'ajustement de l'oscillateur local sera pratiquement irréalisable tenant compte des capacités et inductances parasites qui ne cesseront de varier sans support stable pour les pièces.

D'autres difficultés sont à prévoir aussi concernant les mixers. Le LO doit être très précis pour transmettre sur la bonne fréquence porteuse et démoduler exactement la même. Ce LO exige un excellent circuit tank qui tient compte des effets parasites de la plaquette pour obtenir la fréquence voulue. Il y a donc de l'ajustement très fin à réaliser pour obtenir les bonnes fréquences d'émission et de réception.

5.3 Transmission à 50 MHz

5.3.1 Description

Cette transmission exige très peu de composants pour le module de transmission. Avec un transistor bipolaire contrôlé par les données envoyées au module de transmission, on permet le passage du signal émis par un oscillateur à 50 MHz. Avec un ampli vidéo, ce signal est amplifié. À la réception, un ampli vidéo avec une bande passante dans les 50 MHz permet de récupérer le signal transmis. Un filtre passe-bande avec fréquence de résonance de 50 MHz permet alors de récupérer notre signal modulant par une détection d'enveloppe.

5.3.2 Difficultés envisagées

Des problèmes de bruit sont à prévoir dus au transistor et à réflexions des lignes. Certaines difficultés sont à prévoir pour réaliser un filtre avec une fréquence de résonance aussi élevée.

5.4 Module de transmission RF300TX et de réception RF300RX

5.4.1 Description

Ces modules sont fabriqués pour des applications allant du contrôle de systèmes d'alarme de voiture au carillon de porte, en passant par des applications de contrôle de machineries ou des projets étudiants. Deux parties principales composent chaque module : une partie de codage/décodage et une partie de transmission RF/réception RF, pour le transmetteur et le récepteur respectivement.

Le transmetteur RF300TX fonctionne comme suit. En appuyant sur l'un des deux boutons du transmetteur, l'alimentation est connectée sur une puce (Holtek HT680 3^{18} Series of Encoders) qui envoie aussitôt un message de 18 bits à la partie de transmission RF à un taux d'environ 1200 bauds. Les détails de la transmission sont expliqués dans un document pdf nommé 3_18E.pdf sur le site de la compagnie Holtek (www.holtek.com.tw). La sortie de l'encodeur est alors connectée à la base d'un transistor bipolaire. Un circuit oscillant à 318 MHz est relié au collecteur de ce même transistor. La fréquence d'oscillation du circuit peut être changée par le condensateur variable. Le schéma ci-contre ne reproduit pas fidèlement le circuit imprimé du module : la ligne qui relie la résistance au collecteur fait en réalité le tour des condensateurs. Une inductance est créée de cette façon qui permet l'oscillation du circuit. Lors de l'envoi d'un 1 logique à la base du transistor, le signal oscillant passe et l'antenne, connectée au V_{CC}, émets le signal modulé (voir figure 3). Une LED connectée à l'émetteur permet de voir le passage des bits de niveau logique 1.

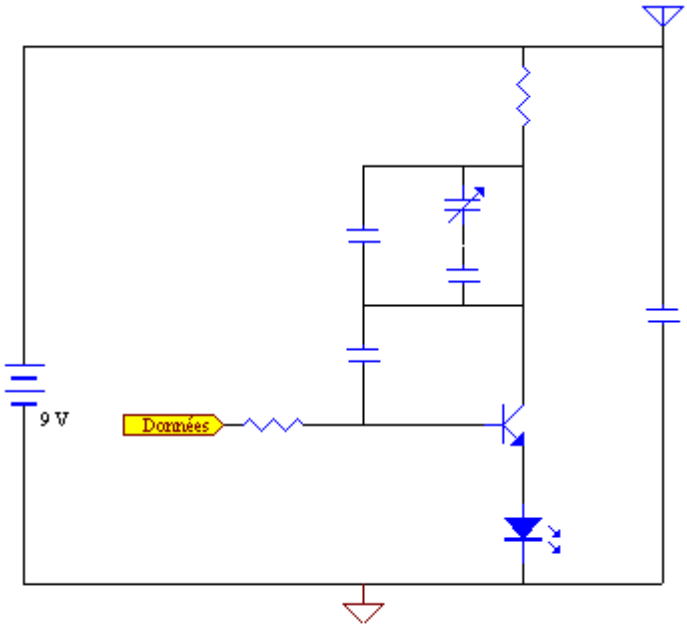

Figure 5.3 -Module de transmission RF du RF300TX

Le récepteur RF300RX n'a pas été étudié en détails car un document fournit avec celui-ci indiquait les endroits où aller chercher les signaux sur le module pour réaliser notre transmission. Un document est disponible de la compagnie Holtek expliquant le fonctionnement de leur décodeur, soit la puce HT682. C'est le document 3_18d.pdf.

5.4.2 Module de transmission

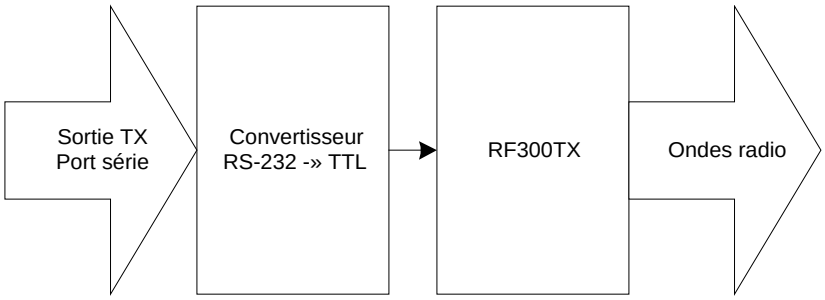

Figure 5.4 -Module de transmission RF basé sur le RF300TX

Le cœur du module de transmission est le petit émetteur RF300TX. À partir de la sortie TX du port série, un convertisseur RS-232 à TTL nous permet d'envoyer directement notre signal à la partie de transmission RF du module RF300TX. Une batterie de 12 V fournit avec le module procure l'alimentation du module RF300TX et une autre source de 5 V alimente le convertisseur. Ce détail est discuté plus loin dans la section Difficultés et solutions. Les fréquences d'opération des modules de transmission sont ajustées à la main, en variant la valeur du condensateur.

5.4.3 Module de réception

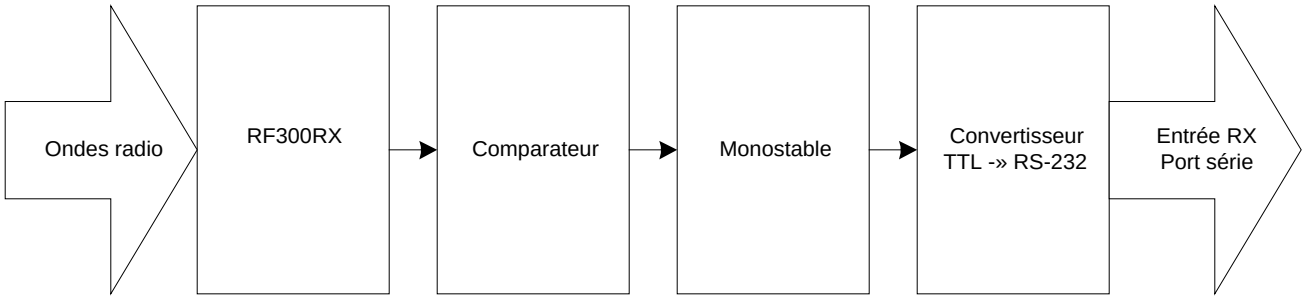

Figure 5.5 -Module de réception RF basé sur le RF300RX

Le module de réception RF300RX permet de récupérer le signal modulant de la porteuse. La fréquence d'opération est ajustée par un condensateur variable pour obtenir le maximum de signal reçu par le récepteur. Un comparateur permet de détecter les bits envoyés à travers le bruit qui se retrouve dans le signal à la sortie du récepteur. Le monostable permet de régénérer les bits reçus avant d'être envoyés à un convertisseur TTL à RS-232. Le module RF300RX du robot est alimenté en 10 V, alors que celui du PC est alimenté en 12 V. C'est les mêmes alimentations pour le comparateur. Le monostable est alimenté en 5 V pour les deux modules de réception, ainsi que les convertisseurs.

5.4.4 Difficultés envisagées et solutions proposées

L'antenne est la raison pour laquelle le module de transmission RF300TX est alimenté indépendamment du convertisseur. Elle est reliée sur la borne positive de l'alimentation. Si la même alimentation est utilisée pour le module de transmission et de réception pour le PC ou le robot, ils risquent de communiquer entre eux par leur alimentation. Afin d'éviter ce problème, la batterie fournit avec le module de transmission est conservée et permet ainsi une alimentation séparée pour le module de transmission et de réception. Pour ne pas surcharger cette batterie, une autre source d'alimentation est utilisée alors pour le convertisseur.

Afin d'économiser la batterie du robot, nous alimentons son module de réception par les batteries supplémentaires qui alimentent le micro.
La fréquence d'opération des modules de réception ne peut être ajustée pour leur permettre de ne capter qu'un seul transmetteur. Les récepteurs captent le signal provenant des deux modules malgré plusieurs efforts pour La communication sera half-duplex et non full-duplex.

Conclusion

Bref, bien que certaines parties comme la communication sans fil n'a pas plus être intégrée, mais nous avons quand même atteint une grande partie de nos objectif ont été atteint comme le positionnement, le contrôle du robot et la détection d'obstacle. Malgré notre défaite dans un des modules, ce cours nous a été très profitable d'un point vue technique, mais aussi d'un point vue gestion de projet. Nous avons pu combiné nos connaissances acquise dans différents domaines pour atteindre le but fixé.