Université Laval

Département de génie électrique et de génie informatique

GEL-16146 Systèmes Electroniques Numériques

Session Automne 1998

Rapport Final

Projet UBI

Présenté à Monsieur Denis Poussart

Présenté par l'équipe A, Charlie Inc.

Paul Bélanger	95 029 016
David Bordeleau	95 038 907
Isabelle Chabot	95 032 093
Vincent Gagnon	
Dominic Gauthier	95 013 764
Jasmin Giroux-Maltais	
Éric Ménard	95 095 783
Jean-François Mercier	95 021 853
David Panneton	95 058 289
François Paquin	96 039 931

Le 18 décembre 1998

Table des matières

Table des figures

1. Introduction

En septembre dernier, Monsieur Denis Poussart, professeur du cours GEL- 16146 Systèmes Électroniques Numériques à l'Université Laval, a confié à l'équipe A Charlie Inc. le mandat de réaliser un système permettant à un micro-robot mobile de :

- se localiser dans un environnement de laboratoire et
- d'utiliser cette information pour piloter correctement une trajectoire de navigation déterminée
- au moyen d'un poste de commande évolué qui communique avec lui par une liaison sans fil.

Après avoir procédé à des travaux continus au cours de la session automne 1998, Charlie Inc. est fière de vous présenter les résultats obtenus. Vous trouverez donc plusieurs chapitres traitant des différents aspects du projet, soit la localisation, le positionnement, l'interface usager, la communication RF et la caractéristique particulière.

Tout en donnant l'heure juste sur les hypothèses, les essais, les erreurs et les réalisations de l'équipe, ce rapport présente aussi les améliorations possibles pour une version 2 de notre système.

Il est important de noter que le système traité dans ce rapport est fonctionnel et peu coûteux. Grâce à un choix judicieux des composantes du projet, nous avons pu faire la conception de circuits simples qui permettraient une mise en marché à prix abordable.

2. Localisation

Cette section traite des modules liés aux fonctions de localisation du robot par rapport à son environnement.

2.1 Introduction

L'objectif de l'équipe *localisation* est de concevoir un générateur de blips, une carte d'acquisition du signal provenant des haut-parleurs et de fournir les algorithmes de calcul à l'équipe *positionnement*.

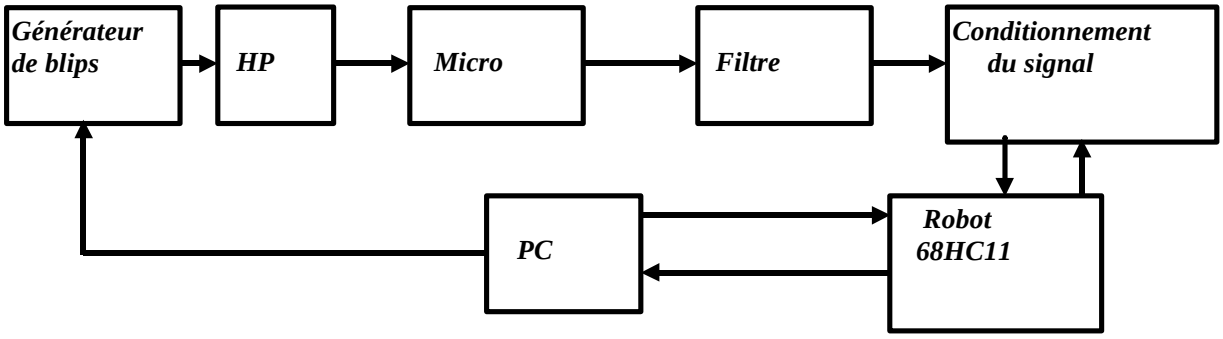


Figure 1 : Schéma bloc de la partie localisation

2.2 Problématique

De quelle façon allons-nous indiquer à l'ordinateur que le micro a capté le signal provenant des haut-parleurs ?

- Le microphone capte le signal.
- Le signal capté est préamplifié et filtré.
- Le signal filtré est conditionné pour donner une onde carrée 0-5V.
- Ce signal est envoyé au 68HC11 pour signifier un blip reçu.
- Le 68HC11 emmagasine les valeurs et les transmet à l'ordinateur via le port série.
- L'ordinateur utilise l'algorithme de localisation pour traiter les données envoyées par le robot.

2.3 Générateur de blips

Le générateur de blips est un module électronique qui, par une commande venant du port parallèle de l'ordinateur, permet l'envoi d'impulsion sonore (d'environ 10kHz) sur quatre caisses de son. Les impulsions sonores suivent une séquence très spécifique de durée précise. Cette séquence consiste à l'envoi d'ondes sonores, durant 2 msec chacune, sur la caisse #1 à la caisse #4 espacée d'un temps mort de 0.5 seconde. Voici un schéma expliquant la séquence.

Vous pourrez voir, à la fin de ce texte, le schéma électrique du générateur version finale. Cependant, avant d'être arrivé à cette version, le générateur de blips a connu plusieurs modifications. Chacune des versions est munie d'un microcontrôleur de la compagnie Microchip (pic16C74A). La seul différence entre chacune des versions est la façon de contrôler les caisses de son.

2.3.1 Générateur de blips version 1

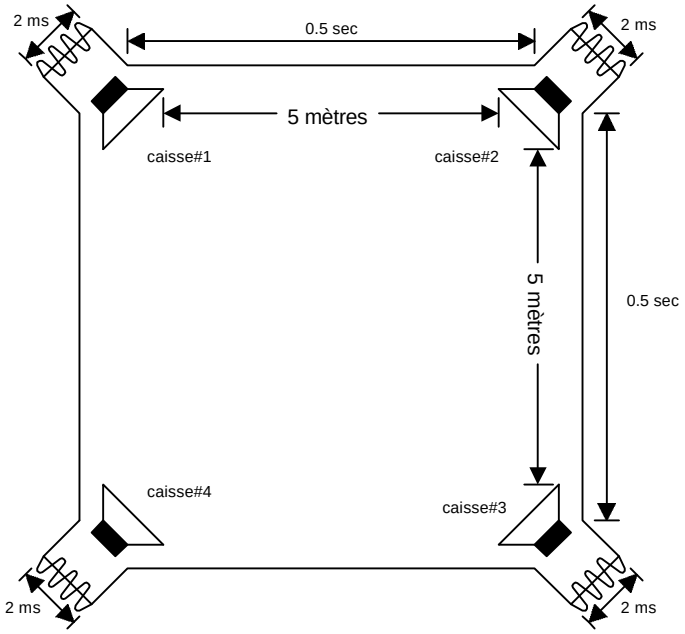


Figure 2 : Le générateur de blips sonores

Dans cette version du circuit, le microcontrôleur active des transistors bipolaires (2N2222A) en guise d'interrupteurs, laissant ainsi passer une onde d'environ 10kHz entre son collecteur et son émetteur.

2.3.1.1 Problèmes rencontrés

Dans ce type de design, le problème est que l'on retrouve un courant continu qui s'écoule entre la base et l'émetteur du transistor, forçant l'ajout d'un condensateur afin d'éliminer ce courant et ainsi empêcher d'endommager le haut parleur. Un autre problème rencontré avec ce type de design est que le transistor, une fois désactivé, laisse passer continuellement une onde d'environ 10kHz de faible amplitude. Cette onde, audible, est inacceptable car elle pourrait éventuellement déstabiliser notre module de localisation.

2.3.2 Générateur de blips version 2

Pour éliminer ce problème, les transistors bipolaires ont été remplacés pas quatre interrupteurs analogiques de type CMOS (14066B). Ce circuit n'a pas permis d'éliminer l'écoulement sonore mais il a permis de le diminuer à une valeur respectable.

2.3.2.1 Problème rencontré

Le problème avec cet interrupteur analogique est qu'il est limité à recevoir, au maximum, un signal égale à sa tension d'alimentation divisé par 2 ($Vcc/2 \approx 2.5V_{pp}$) ce qui n'est pas assez fort pour permettre au micro de le détecter. Il est à noter que ce type d'interrupteur ne permet pas le passage d'un signal négatif, donc en réalité nous sommes forcé d'ajouter un « offset » à notre signal diminuant ainsi son amplitude à $1.25V_{pp}$.

2.3.3 Générateur de blips version 3

Cette version est très simple. Elle consiste à créer, par nous même, l'onde à la fréquence que l'on désire, en faisant osciller chacune des quatre pins du microcontrôleur branchées directement aux amplificateurs des caisses de son.

Cette version est celle qui est utilisée dans la version finale du projet et qui est représenté dans le schéma suivant.

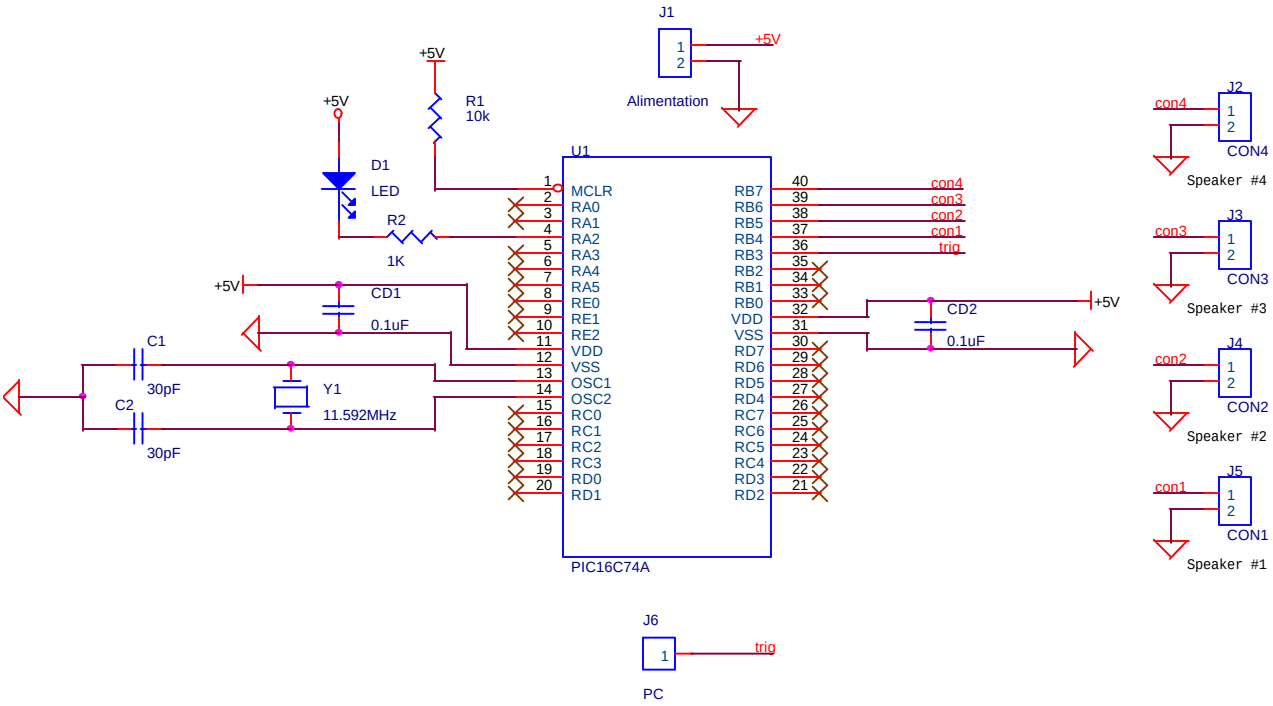


Figure 3 : Dernière version du générateur de blips

2.4 Microphone

Le microphone utilisé est un modèle à condensateur, acheté chez un détaillant. Pour fonctionner, le micro doit être alimenté avec une tension de +5V. Son fonctionnement est fort simple : une variation de la pression acoustique fait bouger le mince film d'aluminium qui constitue une des deux plaques du condensateur. Cette variation de l'espacement des plaques engendre une variation de tension. Ainsi, une pression acoustique est convertie en signal audio.

2.5 Préamplificateur

Le signal capté par le micro est envoyé dans un circuit intégré (Analog Devices SSM2166) agissant comme préamplificateur.

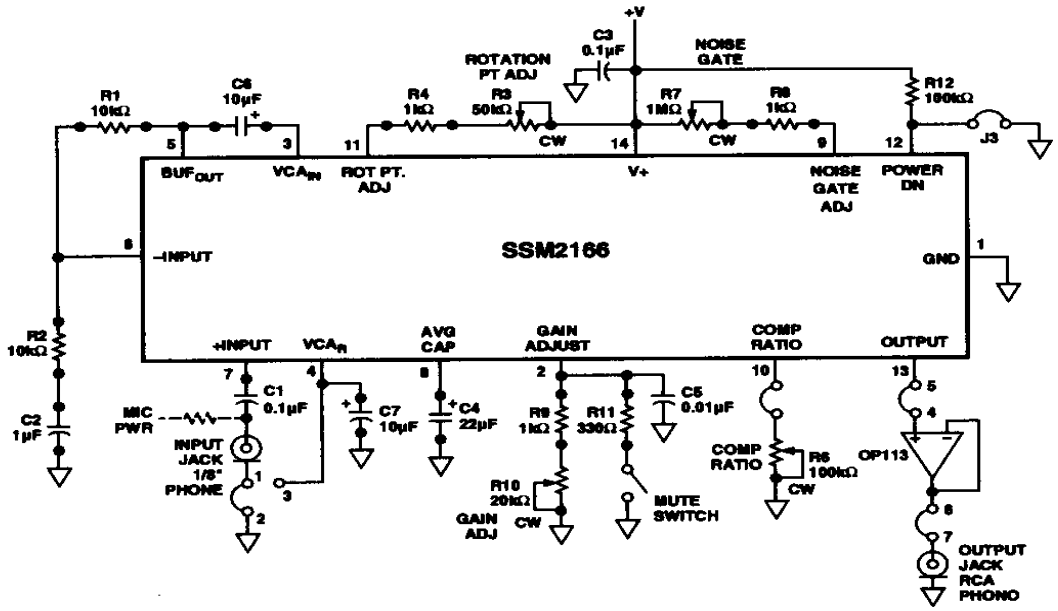


Figure 4 : préamplificateur SSM2166[1]

Ce circuit intégré, bien que très complet de par les nombreux ajustements permis fut vite abandonné. Les dits ajustements étaient plus souvent des sources de problèmes que des solutions.

Une deuxième approche a été d'utiliser un amplificateur opérationnel monté en inverseur. Ce circuit a été réalisé à l'aide d'un TLO82 de Texas Instruments.

[1] Analog Devices. *SSM2166 etc.*

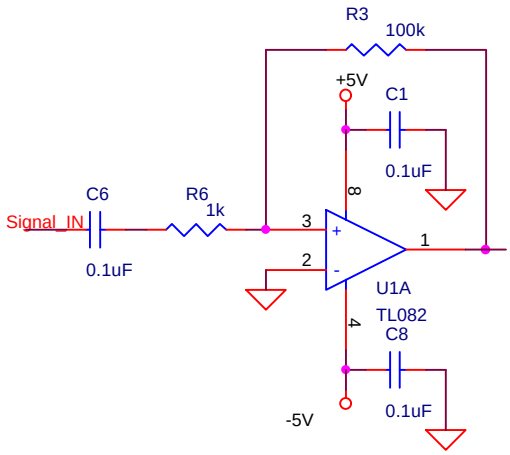


Figure 5 : Préamplificateur du microphone

Le condensateur C6 a pour fonction de bloquer le signal DC.

2.6 Filtre

Par la suite, le signal préamplifié est filtré. Cette étape vise à éliminer les bruits ambiants pour ne garder que la composante spectrale correspondant à la fréquence du clic.

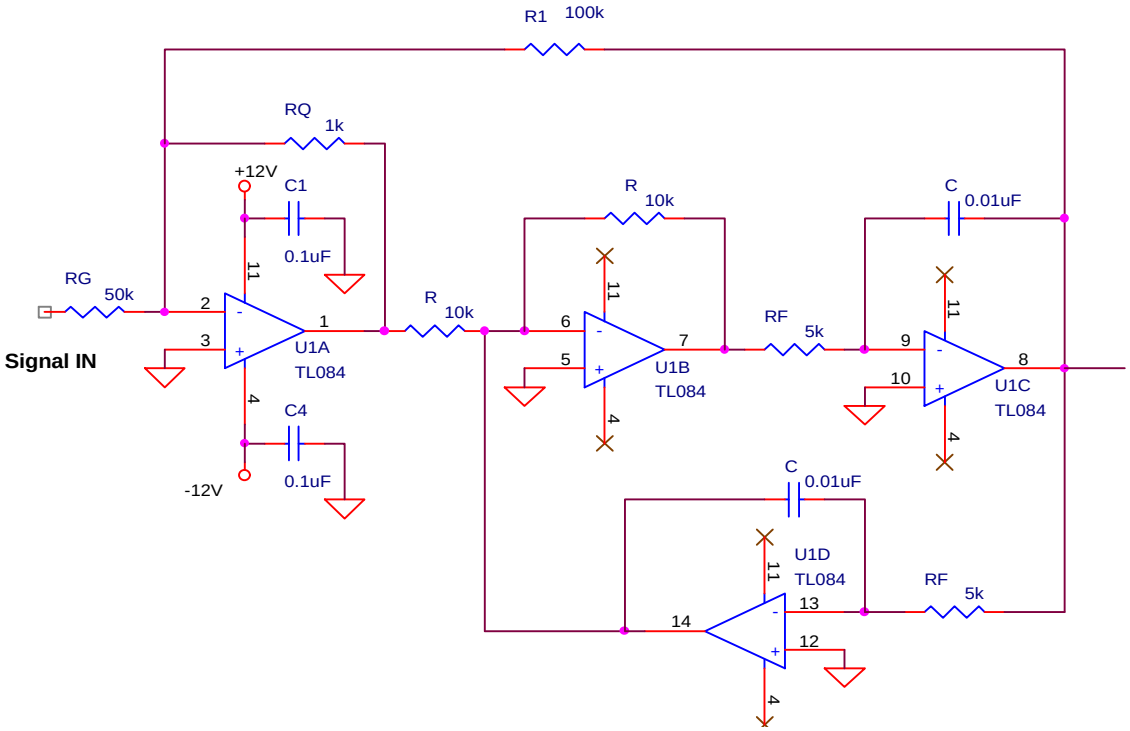


Figure 6 : Schéma d'un étage du filtre

Le filtre est réalisé avec des TLO84 (Texas Instruments). Nous avons cascadé 2 filtres identiques pour obtenir une coupure plus précise à la fréquence désirée. Les résistances employées pour le filtre ont une précision relative de 1%.

Sa fonction de transfert est : $$\frac{Vout}{Vin} = \frac{\frac{-s}{CRf}\frac{Rq}{Rg}}{s^2 + \frac{sRq}{CRfR1} + \frac{1}{C^2Rf^2}}$$

Tous les paramètres du filtre (f_0, Q et G) peuvent être ajustés indépendamment de la façon suivante : $f_0 = 1 / 2\pi R_F C$ $Q = R_1 / R_Q$ $G = R_1 / R_G$

Les valeurs de C et de R sont de 0.01 µF et 10 kΩ respectivement.

Les premiers tests ont été réalisés pour obtenir une fréquence de coupure autour de 3.5 kHz. Nous avons choisi R_F = 5 kΩ, R_Q = 1 kΩ et R_G = 50 kΩ. La fréquence de coupure observée du filtre est de 3340 Hz.

La réponse en fréquence observée[2] du filtre est la suivante :

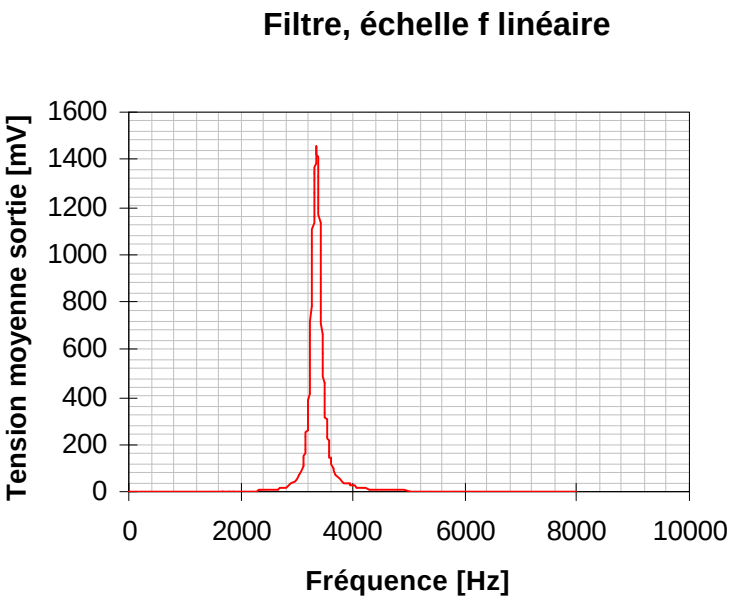


Figure 7 : réponse en fréquence observée pour f0 = 3.4 kHz

Ce test a été effectué en échantillonnant 127 points.

Cependant, à la suite de conseils de la part de personnes ressources, nous avons changé la fréquence du clic à 10 kHz pour que le signal capté par le micro soit moins perturbé par les bruits ambiants. La caractéristique spectrale du filtre a été modifiée pour que f_0 = 10 kHz. La modification est simple : changer la valeur de R_F pour 1.6 kΩ.

De nouveaux tests de réponse en fréquence ont été réalisés[3]. La figure (x) en montre les résultats.

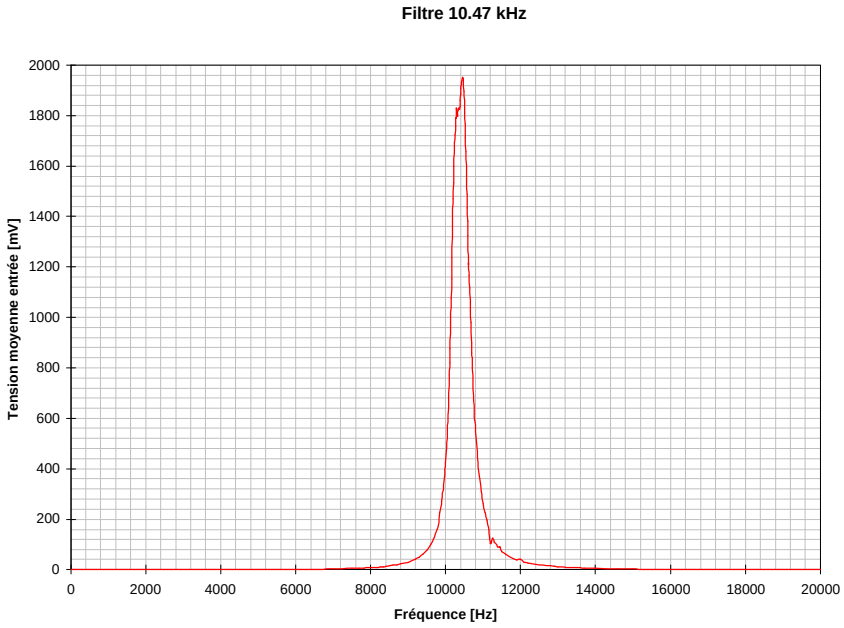


Figure 8 : réponse en fréquence observée pour f0 = 10 kHz

Ce test a été réalisé en échantillonnant 138 points.

2.7 Conditionnement du signal filtré

La sortie du filtre est envoyée dans un amplificateur pour retrouver un niveau de signal non atténué. La sortie de l'amplificateur est envoyée dans un comparateur LM311. Le but est de transformer la réponse sinusoïdale de la sortie de l'amplificateur en une

[2] Tests effectués avec générateur de fonctions HP 33120A et multimètre Beckman DM15XL.

série d'impulsions générées par le comparateur. En ajustant correctement le niveau de tension du comparateur, on obtient ceci :

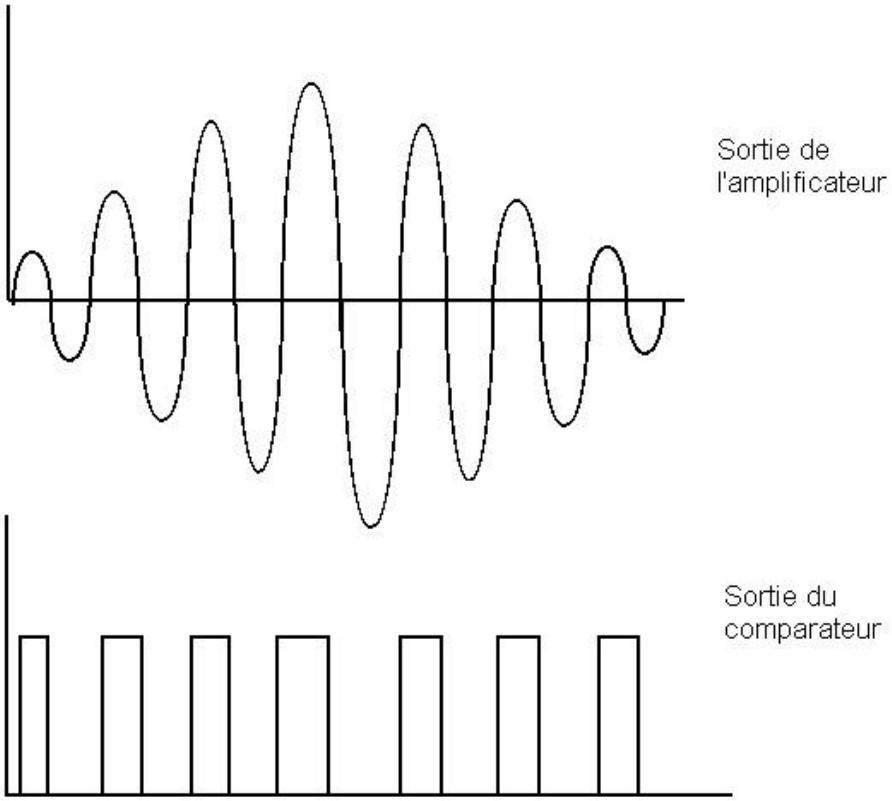


Figure 9 : Sortie de l'amplificateur et du comparateur

Par la suite, la sortie du comparateur devient le signal d'horloge d'une bascule D (Motorola MC14043). L'entrée D de la bascule est reliée au Vcc. De cette façon, dès que la sortie du comparateur devient haute, la bascule reçoit un coup d'horloge qui envoie à la sortie un signal continu de +5V. Ce signal est envoyé à la (XXX) PA1 du port A du 68HC11. Ce signal signifie au microcontrôleur que le blip provenant des haut-parleurs est capté. Le 68HC11 envoie un « 1 » logique à la (XXX) P5D du port D. ce signal sert à faire une remise à zéro de la bascule, opération simplifiée par le fait que cette dernière possède une RAZ asynchrone.

[3] Tests effectués avec générateur de fonctions HP 33120A et multimètre Beckman DM15XL.

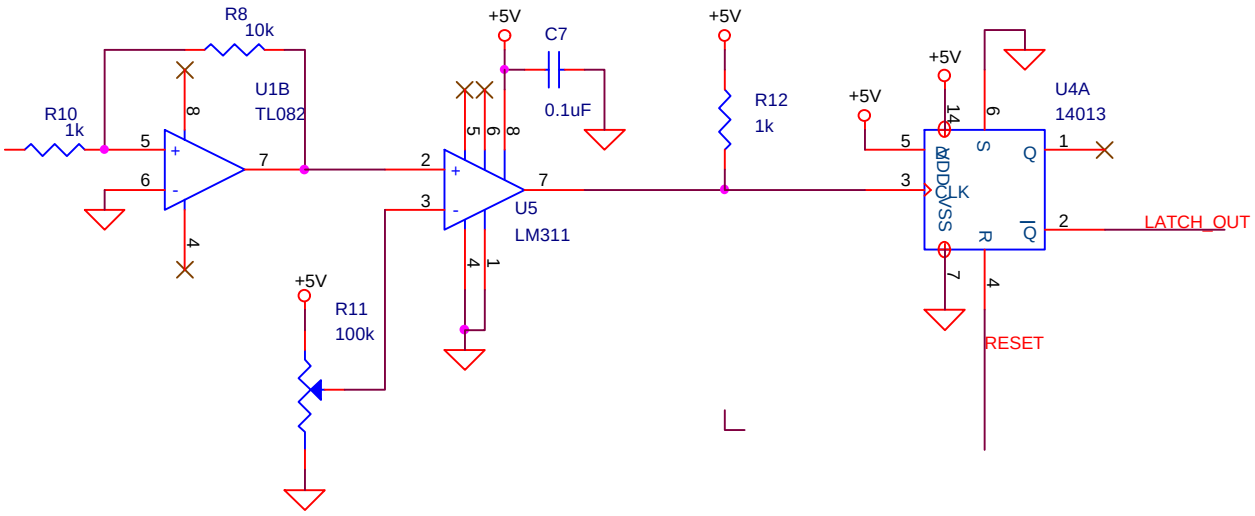


Figure 10 : Schéma de la partie conditionnement

2.8 Calcul du positionnement

Voici d'abord un schéma simplifié de la surface de « jeu » :

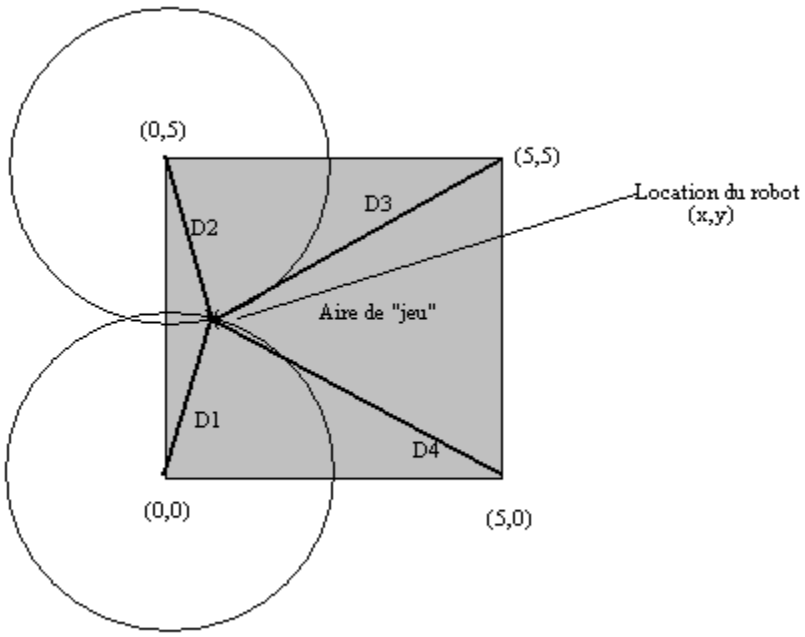


Figure 11 : Schéma de la surface de calcul

La première solution envisagée était d'envoyer 5 blips à 500ms d'intervalle et de mesurer le temps entre chaque blip. Cette méthode a comme avantage de n'introduire aucun délai de transmission entre le PC et le robot, car on n'a qu'à dire au robot d'être prêt à écouter. Les seules sources d'erreur possibles peuvent venir des circuits de filtrage, des compteurs du robot ou du générateur de blips. De plus, un délai constant venant des compteurs du robot, par exemple, ne changerait pas les mesures puisqu'on mesure les différences de temps entre les blips.

Avec 5 blips, on mesure donc 4 temps; théoriquement, on aurait assez d'information avec 2 temps, mais 4 temps nous auraient permis de calculer plusieurs positions et de faire une moyenne pour minimiser les erreurs.

2.8.1 Calculs

Définissons t1 comme le temps entre le haut-parleur (0,0) et (0,5), t2 entre (0,5) et (5,5), t3 entre (5,5) et (5,0) et t4 entre (5,0) et (0,0).

On aurait donc :

$$t1 = 0.500s + (D2 - D1)/Vs$$
$$t2 = 0.500s + (D3 - D2)/Vs$$
$$t3 = 0.500s + (D4 - D3)/Vs$$
$$t4 = 0.500s + (D1 - D4)/Vs$$

avec

$$D1 = \sqrt{x^2 + y^2 + h^2}$$
$$D2 = \sqrt{x^2 + (l - y)^2 + h^2}$$
$$D3 = \sqrt{(l - x)^2 + (l - y)^2 + h^2}$$
$$D1 = \sqrt{(l - x)^2 + y^2 + h^2}$$
Vs = Vitesse du son
h = Différence entre la hauteur des caisses et celle du micro
l = largeur de l'aire de " jeu"

Toutefois, nous n'avons pu implanter cette méthode car, en tentant d'isoler x ou y, on se retrouve avec des polynômes d'ordre 4 en x et y, très difficiles à résoudre même numériquement.

Il aurait aussi été possible d'utiliser cette méthode de mesure, mais sans calculs : il aurait fallu prendre plusieurs mesures sur le terrain (ou au pire calculer plusieurs temps théoriques), placer les résultats dans une matrice et faire de l'interpolation. Toutefois, ceci nécessitait que tous les circuits de positionnement soient fonctionnels assez tôt pour effectuer ces mesures et les calculs devant les suivre. De plus, l'interpolation n'aurait pas pu être linéaire, ce qui aurait causé d'autres problèmes (calculs d'interpolation aussi compliqués que les premiers calculs).

Finalement, nous avons choisi la méthode suivante : mesurer directement le temps de la caisse au robot, pour 4 blips (1 par caisse). Le robot envoie un « GO! » au robot et au générateur de blips. De cette façon, le robot sait à quel moment le générateur envoie son premier blip et se met à compter. Il store ensuite les 4 mesures de temps pour chaque blip et les retourne au PC. Celui-ci rappelle la routine une seconde fois pour avoir finalement 8 temps (2 pour t1, 2 pour t2, 2 pour t3, 2 pour t4). Pour avoir les vrais temps, il faut bien sûr soustraire 0,5s de t2, 1s de t3 et 1,5s de t4 puisque le générateur envoie les blips à toutes les 500ms.

Ensuite, le PC calcule la différence entre chaque paire de temps (en supposant que les paires de temps ayant les plus petites différences entre eux sont ceux ayant la plus faible probabilité d'erreur). Finalement, on choisit deux paires, en tenant aussi compte que les deux temps doivent provenir de caisses voisines (voir explication des calculs plus loin). Ces deux temps sont finalement envoyés à la routine de calcul.

2.8.2 Routine de calcul choisie

L'information qui résulte de la mesure des temps donne une valeur proportionnelle à la distance entre chaque caisse et le robot. Si on trace une sphère (voir schéma du début de cette section) ou un cercle ici pour simplifier, centrée sur la caisse et ayant comme rayon la distance correspondante, on peut voir qu'il y a deux points d'intersection, dont un seul est à l'intérieur de l'aire de « jeu » si les deux caisses sont voisines. Il suffit donc d'utiliser les équations des sphères pour trouver x et y.

Définissons t1 comme le temps de propagation du son entre la caisse à (0,0) et le robot, t2 de même pour (0,5), t3 pour (5,5) et t4 pour (5,0)

On a donc

$$1)\ t1 \bullet Vs = D1 = \sqrt{x^2 + y^2 + h^2}$$
$$2)\ t2 \bullet Vs = D2 = \sqrt{x^2 + (l-y)^2 + h^2}$$
$$3)\ t3 \bullet Vs = D3 = \sqrt{(l-x)^2 + (l-y)^2 + h^2}$$
$$4)\ t4 \bullet Vs = D1 = \sqrt{(l-x)^2 + y^2 + h^2}$$

Vs = Vitesse du son

h = Différence entre la hauteur des caisses et celle du micro

l = largeur de l'aire de " jeu"

Donc, si la paire de temps choisie est t1 et t2, on combine les équations 1 et 2), de même pour les autres paires. Voici les solutions pour x et y pour chaque paire :

Paire 1) et 2)

$$y = (l^2 - (t_2{}^2 - t_1{}^2)v_s{}^2) / 2l$$
$$x = \sqrt{t_1{}^2 / v_s{}^2 - h^2 - y^2}$$

Paire 2) et 3)

$$x = (l^2 - (t_3{}^2 - t_2{}^2)v_s{}^2) / 2l$$
$$y = l - \sqrt{t_2{}^2 / v_s{}^2 - h^2 - x^2}$$

Paire 3) et 4)

$$y = (l^2 - (t_3{}^2 - t_4{}^2)v_s{}^2) / 2l$$

Paire 4) et 1)

$$x = l - \sqrt{t_4{}^2 / v_s{}^2 - h^2 - y^2}$$

$$x = (l^2 - (t_4{}^2 - t_1{}^2)v_s{}^2) / 2l$$

$$y = \sqrt{t_1{}^2 / v_s{}^2 - h^2 - x^2}$$

2.9 Conclusion

Faute de temps et, restons honnêtes, d'expérience, certains aspects de cette partie n'ont pu être testé à fond. Entre autres, les tests de réponse en fréquence du circuit global, une fois assemblé et soudé sur la plaquette de montage n'a pas été réalisé. Nous aurions pu tester la réponse à une onde carrée pour s'assurer de la stabilité du système et caractériser « l'overshoot ».

Aussi, nous aurions pu tester la précision du circuit et en particulier du filtre aux changements de température même si, en réalité, le circuit sera toujours en opération à la température ambiante.

If something can go wrong...

Enfin, un des derniers correctifs à apporter serait de se fixer un échéancier plus réaliste. Pour des raisons parfois hors de notre contrôle, nous n'avons pu procéder aux tests avec la communication qu'au début décembre. Ces tests étaient prévus pour... la mi-novembre ! Mais, et c'est peut-être ici que l'expérience (ou le manque, c'est selon) entre en jeu : tous ces petits aléas sont des choses à ne pas négliger. Si quelque chose peut mal fonctionner (et ca se produit souvent), ca va mal fonctionner. Tout de même, le circuit fonctionne, c'est le principal !

3. Module Positionnement

Puisque le robot Rug Warrior possède son propre micro-contrôleur, il nous semblait tout naturel qu'il possède son propre programme de contrôle. Mais encore, fallait-il le concevoir…

3.1 Introduction

Bien que le robot possède un groupe moteur et différents senseurs, il n'en demeure pas moins qu'il ne peut se déplacer seul et que, même s'il possède une connexion avec un ordinateur, ce dernier ne peut contrôler les moteurs du robot. Dans cet optique, un module de positionnement devait être conçu, à même le micro-contrôleur du robot, afin de pouvoir actionner les composantes du robot.

Tout d'abord, des tests préliminaires ont été effectués sur le robot dans le but de comprendre son fonctionnement et celui du compilateur Interactive C. Ensuite, un protocole de communication a été développé, en coopération avec les membres du module interface usager, dans le but de faciliter la communication entre le robot et l'ordinateur. Les premières fonctions de base ont été codées et une batterie de tests a suivi, dans le but de s'assurer d'avoir une base solide pour le développement de fonctionnalités plus sophistiquées. Enfin, c'est le code pour la partie microphone qui a été développé afin de pouvoir l'intégrer au projet.

En consultant le calendrier primaire sur le site de la compagnie, il est possible de constater que la cédule des tâches à réaliser a été assez bien suivie. En effet, jusqu'au 19 octobre, l'implémentation des fonctions de base a été réalisée ainsi que les premiers tests. Ensuite, les semaines du 2 et du 9 novembre ont servies à tester l'interface usager et le protocole de communication. Les dernières semaines ont servies à créer

les fonctions pour le module du microphone, à l'intégrer et le tester. Enfin, la dernière semaine a servi à l'intégration et aux tests des autres modules.

3.2 Tests avec le robot

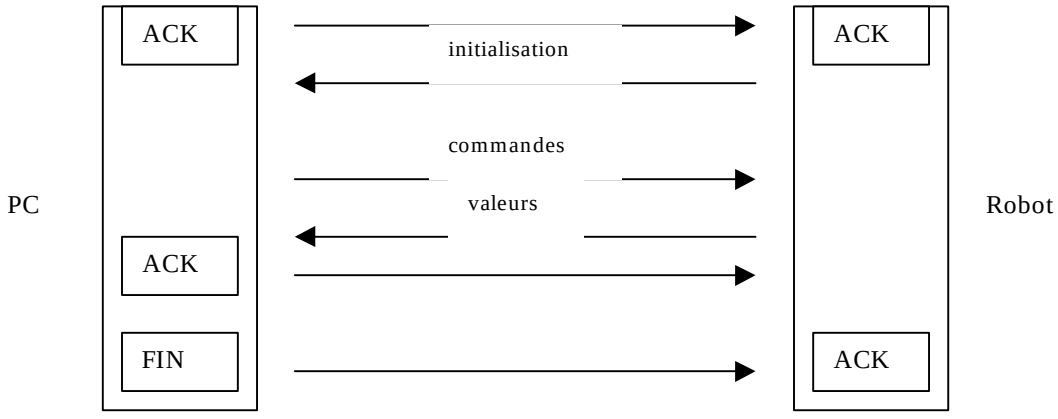
Le seul matériel fourni pour la réalisation du projet était un robot, un compilateur C ainsi que deux livres expliquant brièvement le fonctionnement de ces derniers. Notre premier objectif fut donc de comprendre le fonctionnement du compilateur, de trouver et de comprendre ses principales caractéristiques et enfin, de réussir à le faire fonctionner.

Dans un premier temps, des fichiers exemples, fournis avec le compilateur Interactive C, ont été utilisé afin de s'assurer du bon fonctionnement du compilateur, du robot ainsi que du lien de communication série. À ce moment, nous avons commencé à étudier le contenu des fichiers exemples afin de bien comprendre la façon d'utiliser les différentes caractéristiques du robot. Par la suite, nous avons créé notre premier programme utilisant les caractéristiques de base du robot : TRACK, IR_DETECT(), BUMPER()…

Dans un deuxième temps, la communication série entre un programme roulant sur un ordinateur et un programme roulant sur le robot a été testé. Ceci dans le but de pouvoir commencer le développement de l'interface usager.

3.3 Protocole de communication

Puisque les déplacements du robot sont contrôlés par un ordinateur, donc par l'interface usager, il fallait créer un mode de communication efficace entre le robot et l'interface.



Ce protocole devait s'assurer que les données étaient bien rendues à destination et que chacune des parties était toujours prête à recevoir de l'information. Un schéma du protocole de communication est illustré sur la figure 3.1 ci-haut.

La méthode développée est à la fois très simple et très efficace. En effet, le programme sur l'ordinateur hôte est d'abord amorcé et signale, via son port de communication qu'il est prêt en envoyant un signal ACK. Le programme sur le robot est ensuite exécuté et renvoie à l'ordinateur hôte un signal ACK lorsqu'il reçoit le ACK de l'ordinateur. Puisque, comme mentionné précédemment, c'est l'ordinateur qui est maître de la communication, c'est lui qui va envoyer des commandes au robot. Celui-ci répond à chaque fois par un ACK et effectue ensuite la commande reçue. Lorsque le robot doit retourner de l'information à l'ordinateur (suite à une demande de ce dernier), il envoie une première valeur et attend un ACK de l'ordinateur. Lorsqu'il reçoit le signal, il envoie la deuxième donnée et ainsi de suite jusqu'à la fin. Lorsque l'utilisation du robot est terminée, un signal fin est envoyé depuis l'interface et la communication est rompue par le robot.

3.4 Premières fonctions de base

Lorsque le fonctionnement du robot fut bien compris et lorsque le protocole de communication fut clairement défini, le codage des fonctions de base commença.

En effet, les deux premières fonctions à avoir été développées furent celles permettant d'écrire sur le port de communication du robot et celles permettant de lire sur ce même port. Ensuite, la fonction GOTO fut développée. Cette dernière illustre bien le mode de fonctionnement de notre protocole de communication. L'ordinateur envoie la commande GOTO. Le robot entre dans sa routine GOTO et attend de recevoir une valeur représentant une vitesse. Lorsqu'il la reçoit, il transmet un ACK et attend de recevoir une valeur représentant un angle de rotation. Encore une fois, il envoie un ACK et attend de recevoir une dernière valeur représentant une distance à parcourir. Il envoie

un dernier ACK et calcule les bons paramètres à transmettre à la fonction TRACK afin d'accomplir une bonne rotation et un bon déplacement.

C'est à partir de ce moment que nous avons commencé à rencontrer nos premiers problèmes. Premièrement, seulement 8 bits peuvent être transmis à la fois ce qui représente des entiers de 0 à 255. Puisque les angles vont de 0 à 360°, il fallait trouver un moyen de contourner cet obstacle. Nous avons décidé de résoudre le problème à l'aide de l'ordinateur. C'est celui-ci qui allait détecter les cas problèmes et envoyer plusieurs commandes de rotation pour réussir à effectuer un angle supérieur à 255°. Le même obstacle fut rencontré pour les distances de déplacement. Bien que l'interface usager pouvait transmettre, en décimètres, des distances pouvant aller jusqu'à 25,5 mètres, le robot ne pouvait franchir des distances supérieures à 1,38 mètre. Ceci était dû au fait que la commande TRACK utilise un nombre de clics comme distance de déplacement et ce nombre ne pouvait dépasser 255, ce qui représente une distance de 1,38 mètre. Contrairement à l'autre, ce problème fut résolu à même le robot, ce dernier effectuant plusieurs commandes TRACK. Enfin, puisque la commande de déplacement n'utilise que des entiers, elle perd de la précision lorsqu'elle tronque ou lorsqu'elle arrondie. Sur un petit angle, ceci crée une grande imprécision qu'il faut corriger lors des déplacements subséquents.

3.5 Tests avec l'interface

Une fois les premières fonctions codées, nous avons travaillé sur le calibrage du robot et sur la précision de nos conversions et de nos calculs. Dans le but d'obtenir les résultats les plus fiables possibles, des tests sur le terrain ont été effectués. Ces derniers nous ont permis d'ajuster certains paramètres de calculs afin de compenser les erreurs de précisions du robot. Un bel exemple d'imprécision qui a dû être corrigé est celui concernant la commande TRACK. À la fin d'un déplacement, la commande arrête d'envoyer de l'information aux moteurs mais ceux-ci continuent de « glisser » pendant un certain temps avant de s'immobiliser complètement. De plus, l'erreur

provoquée par ce « glissement » variait avec la distance à parcourir et la vitesse de déplacement. Il a donc fallu créer une méthode de correction d'erreur à même la commande TRACK.

Ces tests nous ont également permis de tester notre protocole. Par souci de rapidité, certaines confirmations (commande ACK) ont été éliminées mais dans l'ensemble, les tests sur le protocole ont très bien fonctionné. À partir de ce moment, il a été possible de faire déplacer le robot à partir de l'interface usager.

3.6 Intégration du microphone

Cette partie a amené son lot de problèmes. En effet, le design initial de la carte contenant le microphone avait été fait pour envoyer, lorsqu'un signal était détecté, un 1 sur le port C du microcontrôleur. Ce dernier produisait alors un 1 sur le port C (la sortie suivante) afin de faire une remise à zéro. Après quelques tests, nous nous sommes rendu compte que le port C était utilisé par l'afficheur à cristaux liquides et qu'il fallait donc utiliser un autre port. Nous avons modifié notre design afin d'utiliser l' «input capture » disponible sur le port A. Le code de la fonction Lire_Clic a également été modifié afin d'utiliser le port A. Mais pour produire notre sortie afin de faire une remise à zéro sur une bascule, il fallait trouver un autre port disponible. Le seul encore disponible était le port D mais il ne possédait pas de connecteur. Il a donc fallu faire ajouter le connecteur pour pouvoir enfin utiliser le port D. Lorsque tout ceci fut terminé, il nous a été enfin possible de tester notre code pour le micro et de tester ce dernier également.

En dernier lieu, il nous a été possible de tester notre routine de capture de clics, de transmission de données et de déplacement avec tous les composantes finement intégrées.

3.7 Conclusion

Il y a une caractéristique du robot qui n'a toujours pas été explorée et qui pourrait grandement être utile pour le module positionnement. Cette caractéristique est la disponibilité, à même le robot, de nombreux senseurs (infrarouge, de contact…) Ces derniers pourraient être utilisés pour la détection de collision et dans le choix de parcours. Cette avenue sera sans aucun doute explorée s'il reste un peu de temps à la fin, lorsque l'intégration des différents modules sera terminée.

4. Module Interface Usager

Ce module constitue l'environnement graphique pour l'utilisateur de notre projet.

4.1 Introduction

On oublie souvent l'importance d'une interface usager conviviale, claire et facile d'utilisation. Sans l'environnement graphique, les instructions à donner au robot sont très complexes. Le but de ce module est donc de faciliter l'interaction entre l'utilisateur et le robot. Pour ce faire, nous avons utilisé l'environnement de développement Microsoft Visual Studio et codé une application .exe en visual c++ pouvant être utilisée sous Windows 95.

4.2 Principe de fonctionnement

Avant d'analyser en détails les caractéristiques de l'interface, il convient d'étudier les liens avec les autres parties du projet. La figure 4.1 présente, sous forme d'un diagramme-bloc, les interactions des autres modules avec l'interface :

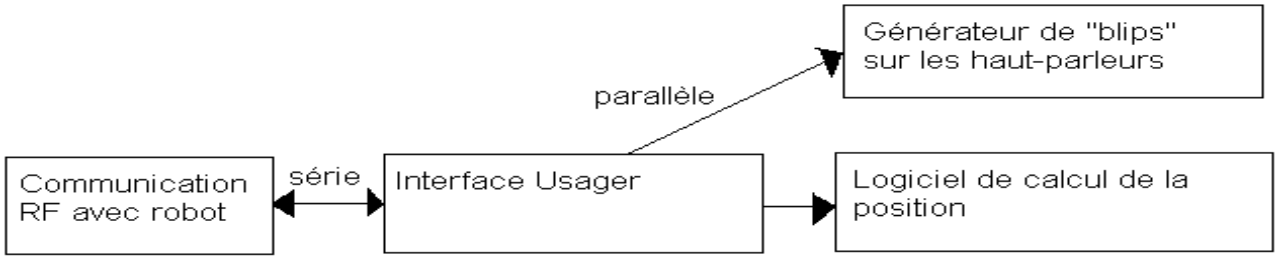


Figure 13 : Schéma Bloc de l'interface

Comme on peut le constater sur le diagramme, l'interface est au centre de toute l'intelligence de notre système. Nous avons choisi de considérer le PC comme le maître et le robot comme l'esclave. De cette manière, c'est toujours le PC qui fait les premiers pas quand une fonction doit être appelée. La communication RF (traitée au point 5 de ce rapport) utilise le port série du PC. Le générateur de "blips" sur les haut-parleurs

(traité au point 2) utilise le port parallèle. Finalement, le calcul de la position du robot se fait à l'aide d'une fonction écrite en C++ (aussi au point 2).

4.3 Caractéristiques techniques

Cette section contient toutes les informations techniques concernant les diverses parties de l'interface et des routines qui la soutiennent.

4.3.1 Aspect visuel

Faire la conception de l'interface demande un œil critique et artistique. C'est pourquoi l'interface a évolué de versions en versions vers un environnement agréable à utiliser, clair, coloré et innovateur tout en gardant l'image de base de la compagnie si bien pensée lors de la conception du site web. Voici donc notre interface :

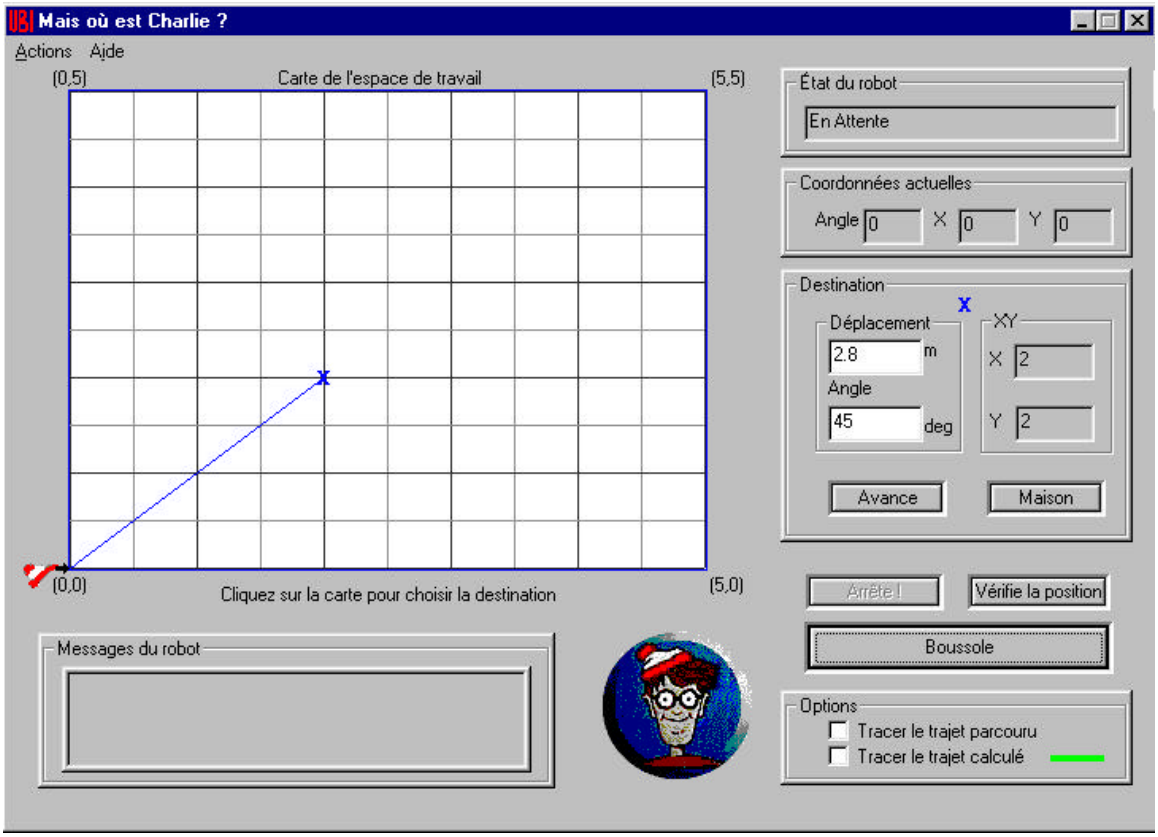


Figure 14 : Interface usager

4.3.2 Les boutons

L'interface est divisée en 5 parties principales : la carte, les messages du robot, l'état et l'emplacement du robot, la destination et les options. Cette partie traitera du résultat de la sélection de chacun des boutons et la section suivante concernera les fonctions en visual C++ qui permettent ces actions.

La barre menu située en haut de l'interface est standard, permettant à l'usager de quitter l'interface, de remettre tous les champs à zéro, de consulter l'aide et d'en savoir plus sur les auteurs.

La carte de l'espace de travail occupe le premier plan au sein de l'interface puisqu'elle est l'instrument principal de contrôle du robot. Voici ce dont elle a l'air:

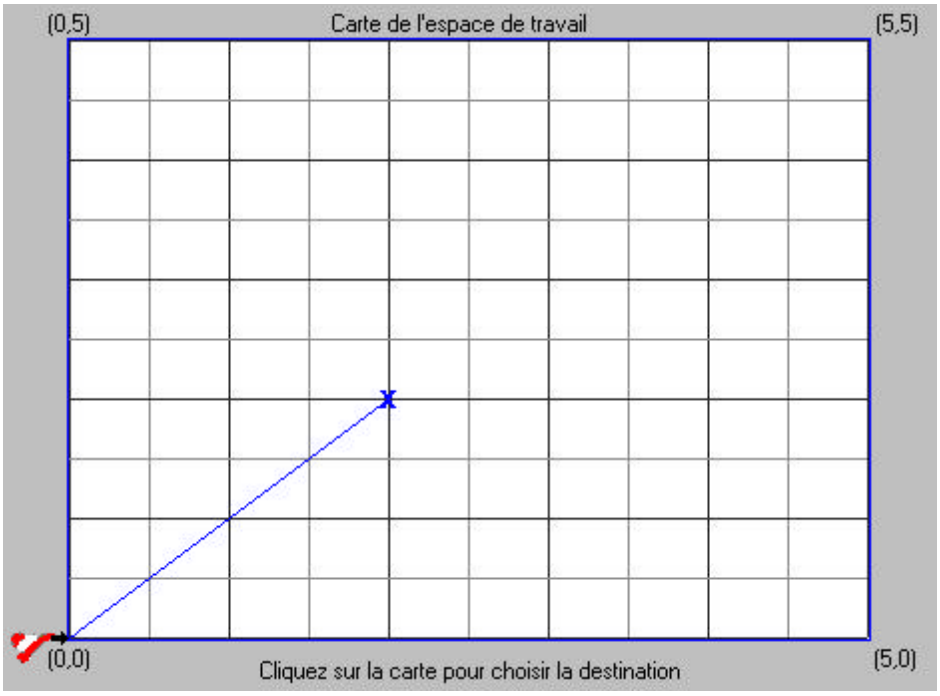


Figure 15 : Carte de l'espace de travail

On peut remarquer 3 caractéristiques principales sur cette carte: la tuque, le "X" bleu avec son trait et les lignes qui la divisent. La carte a été dessinée avec les mêmes

séparations aux .5 m que sur le site d'essai du projet. Cet espace représente une superficie de 5 m^2. La flèche noire qui complète la tuque de Charlie point vers l'emplacement exact du robot au moment présent. Dans ce cas, le robot est à la position (0,0). L'ensemble tuque-flèche se déplace sur la carte une fois la destination atteinte par le robot sur le sol. Le "X" bleu apparaît lorsqu'on clique avec le bouton de gauche sur la carte. A ce moment, un trait bleu apparaît pour signaler le trajet que le robot parcourra si cette destination est choisie.

Le bloc des messages du robot permet à l'utilisateur de savoir quelles valeurs ont été transmises au robot et de savoir dans quelle routine il est. C'est surtout un outil de conception pour la vérification des fonctions.

Le bloc de l'état et des coordonnées du robot est très utile. Il permet de savoir dans quel état se trouve le robot: en attente, position atteinte, obstacle rencontré et ses coordonnées en X et Y ainsi que la direction vers où il pointe (angle).Voici une image de ce bloc:

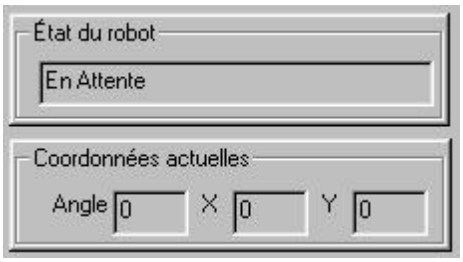


Figure 16 : Bloc de l'état

Le bloc de Destination est sans doute le plus utilisé. Il permet de choisir la destination voulue et agit en parfaite complicité avec la carte. Comme expliqué plus haut, on peut choisir la destination du robot en cliquant directement sur la carte. A ce moment, les coordonnées en X et en Y sélectionnées avec la souris sont affichées à droite du bloc Destination. Au même moment, un déplacement et un angle correspondant à cette destination sont aussi calculés. Si la destination ne convient pas tout à fait, il est possible de l'ajuster en modifiant directement les champs de l'angle et du déplacement.

Une fois la destination choisie, il ne reste plus qu'à utiliser le bouton "Avance" pour appeler les fonctions de calcul des déplacements. Le bouton "Maison" sert à renvoyer le robot à sa maison qui est la position (0,0).

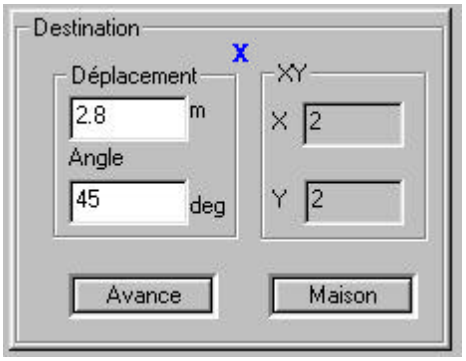


Figure 17 : Bloc de destination

Le dernier bloc est celui des options. Une fois le robot parti vers sa destination, un trait gras vert est utilisé pour dessiner sa trajectoire sur la carte. Le bouton "Arrête!" a été ajouté en cas d'urgence mais comme le robot n'exécute qu'une seule commande à la fois, il n'est pas possible de le faire arrêter d'urgence. Le bouton "Vérifie la position" appelle les fonctions de génération de "blips" et de calcul de la position à l'aide des temps récoltés.

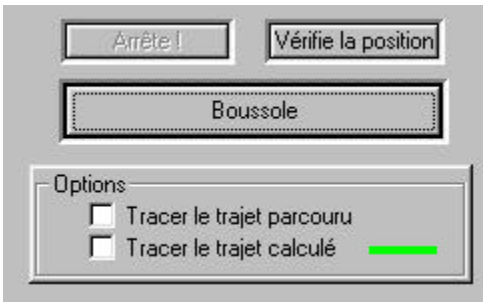


Figure 18 : Bloc d'options

L'utilisateur peu familier avec notre interface peut se questionner sur la façon dont nous traitons les angles. Pour faciliter l'usage de l'interface, nous avons créé une boussole à angles fixes. De cette façon, peu importe vers où pointe le robot, l'utilisateur utilise des

angles définis pour choisir sa destination. Le calcul de l'angle réel que le robot aura à faire est fait par la routine de calcul du déplacement et de l'angle. Voici ce qui apparaît lorsqu'on sélectionne le bouton "Boussole" :

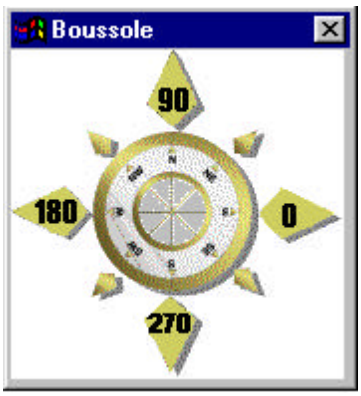


Figure 19 : Boussole

4.3.3 Les fonctions

Vous trouverez dans cette section l'essentiel du code servant au fonctionnement des boutons. Plusieurs routines sont de type programmation Windows (carte, gestion de la souris, affichage de bitmaps) d'autres sont de type c++ (calcul de la destination, calcul de la postions réelle)

4.3.3.1 Carte de l'espace de travail et Destination

Tel que discuté précédemment, la carte permet à l'utilisateur de déterminer et de visualiser les déplacements du robot.

Sélectionner le déplacement

Afin de permettre à l'utilisateur de déterminer les déplacements du robot, la fonction *CMenuDlg::OnLButtonDown(UINT nFlags, CPoint point)* a été définie. Cette fonction est exécutée automatique par la fenêtre Windows à chaque fois que le bouton gauche de

la souris est appuyé. Son deuxième paramètre lui permet de savoir quel point de la fenêtre a été cliqué. On peut alors immédiatement modifier la variable *m_Objectif* qui tient en mémoire les coordonnées du point de l'objectif du robot (affichage du X bleu de destination).

De plus, en appliquant un simple changement de coordonnées (coordonnées de la fenêtre aux coordonnées de la carte 5x5) on peut facilement modifier les valeurs d'angle, déplacement, position en X et position en Y affichées dans le bloc de destination sur l'interface. Voici un exemple du calcul de changement de coordonnées permettant de changer la variable affichée dans le champs « Destination en X » (variable appelée *m_coord_horiz*) :

```
m_coord_horiz = (float)((float) (point.x-m_TopLeft.x)/m_Width)*5;
```

On voit que la position en X du coin supérieur gauche de la carte (*m_TopLeft.x*) est d'abord soustraite de la coordonnée en x du point cliqué (*point.x*) pour changer le référentiel. On divise ensuite le résultat par la largeur réelle en pixels de la carte (*m_Width*) avant de multiplier le tout par 5 pour se mettre en coordonnées de la carte 5x5. On « caste » en virgule flottante (float) à deux reprises dans la formule pour garder la précision dans les calculs.

Une méthode plus précise est aussi offerte aux utilisateurs pour modifier la destination du robot (X bleu). En effet, l'utilisateur peut modifier directement les valeurs de déplacement et d'angle affichées dans le bloc de destination sur l'interface. Aussitôt que ces champs sont modifiés, la fonction *CMenuDlg::OnChangeEDITDeplacement()* ou la fonction *CMenuDlg::OnChangeEDITAngle()* est appelée. Ces fonctions font alors de simples calculs de changements de coordonnées pour permettre la modification de la position du X bleu sur la carte.

Visualiser le déplacement

Comme on vient de le voir, aussitôt que l'utilisateur déterminer le déplacement qu'il veut faire effectuer au robot, le X bleu se positionne immédiatement sur la carte. Ce X ainsi que le segment de droite qui le relie au robot (tuque rouge) indique clairement le déplacement théorique du robot.

Lorsque la commande avance sera lancée (via le bouton « Avance » de l'interface), les commandes de déplacement seront transmises au robot. Dès que le robot se met en marche, la tuque sur la carte se déplace aussi en suivant le chemin théorique. Lorsque le robot arrive à bon port, le calcul de positionnement permet d'afficher la position finale réelle de la tuque. À ce moment, la tuque se positionne à sa position réelle et une ligne noire illustre la différence entre la position théorique et la position réelle.

Pour réaliser le tout, deux fonctions sont appelées au courant du processus de fonction de avance. D'une part, la fonction (qui est en fait un thread) FCT_Animation(LPVoid lpParam) se charge d'animer la tuque le long de sa trajectoire théorique. Pour se faire, on calcule et on affiche le déplacement de la tuque à chaque dixième de seconde en tenant compte de la vitesse et de la destination théorique du robot. D'autre part, la fonction Affiche_Position_Theorique() est exécutée après que le robot ait déterminer sa position réelle. Cette fonction se charge alors de repositionner la tuque à son emplacement réel et d'indiquer via une ligne noire, la distance entre la destination théorique et la destination réelle.

4.3.3.2 La Communication

Pour éviter d'avoir des erreurs de communication et des pertes d'information nous avons décidé d'utiliser le standard de communication *Transmit-Acknowledge*. Son fonctionnement est très simple, à chaque code envoyé le transmetteur attend de recevoir un *Acknowledge* du récepteur avant de transmettre le code suivant. Cette méthode de synchronisation de la communication permet d'être certain que chaque

code transmis est reçu et traité. Il est alors pratiquement impossible que des codes se perdent.

4.3.3.3 Le bouton Avance

Ce bouton est le plus complexe. Une fois cliqué, il appelle le thread avance. Les boîtes du groupe "Destination" sont alors enregistrées. Le thread envoie un code au robot lui indiquant que les données suivantes seront les caractéristiques d'un déplacement. Le logiciel envoie ensuite la distance à parcourir, l'angle et la vitesse (par défaut, 50). Les fonctions de déplacement sur le robot sont alors appelées et celui-ci part vers sa destination. Une fois rendu, il envoie un code de position atteinte ou déplacement arrêté à cause d'un obstacle ou d'un autre problème. Ces résultats du déplacement sont affichés dans les boîtes "État du robot" et "Messages du robot".

Ensuite, la fonction calcul_position est appelée. Celle-ci envoie un code au générateur de blips pour qu'il commence ses blips. Au même moment, le robot se met à calculer les temps entre les blips qu'il reçoit. Le robot retourne ces temps à la fonction qui appelle ensuite l'algorithme de calcul de la position. Celui ci utilise les temps entre les blips pour déduire la position du robot. Une fois cette position calculée, le robot est déplacé à son emplacement réel sur la carte de l'espace de travail et les cases du bloc "Destination" sont modifiées.

4.3.3.4 Le bouton Vérifie la position

Cette fonction est en fait la même routine de calcul de la position que pour le bouton avance, c'est-à-dire que le générateur de blips est appelé et la position calculée à l'aide des temps du robot.

4.3.3.5 Les fichiers Aide et À Propos

Dans la barre menu, sous le titre Aide, on retrouve les fichiers Aide et À Propos. Ces 2 textes proposent des renseignements sur le fonctionnement du logiciel de Charlie Inc. et la composition de son équipe.

4.4 Améliorations futures

Bien que l'interface usager de Charlie soit de grande qualité, nous croyons qu'il serait possible de lui apporter quelques améliorations. En voici les grandes lignes :

Amélioration	**Utilité**
Une boite d'options	Pour modifier les variables utilisées dans le calcul de la position. Ces variables sont la hauteur des haut-parleurs, la grandeur de l'espace de travail et un ajustement sur l'erreur de positionnement.
Modifier la vitesse	L'usager pourrait choisir la vitesse de déplacement du robot.
Améliorer la gestion des obstacles	Pour informer l'usager où se situent les obstacles rencontrés, les placer sur la carte de l'interface usager et ensuite contrôler le robot pour qu'il les évite et continue sa route vers sa destination originale.
Une flèche indiquant la direction	Si, en plus de la boussole, une flèche indiquait la direction où pointe le robot l'usager aurait un point de repère de plus lorsqu'il dirige le robot.
Ajouter un code de *Checksum*	Le *Checksum* (vérifie la somme binaire des codes envoyées) permettrait d'éviter certains problèmes de communication.
Sauvegarde du trajet parcouru par le robot	L'usager pourrait sauvegarder les trajets et ensuite comparer les trajets de plusieurs tests. Surtout utile si les obstacles ne changent pas de place.
Lignes indiquant le chemin théorique et pratique du robot	L'usager pourrait comparer le trajet demandé au trajet effectué. Il pourrait s'ajuster plus facilement aux contraintes de l'espace de travail, en ajustant les options dans la boite d'options.

5 Module de la communication RF

5.1 Introduction

Un des buts du projet était de concevoir un lien sans fil entre le robot et l'ordinateur. Ce lien se fait par onde hertzienne et est conçu de manière à ne pas modifier les protocoles de communication. De cette manière il est toujours possible de remplacer ce module par un fil. Cette partie regroupe donc l'ensemble des expérimentations et des résultats réalisés lors de ce projet.

5.2 Information sur la communication série

La communication série entre l'ordinateur et le robot s'effectue à une vitesse de 9600 bauds/s ce qui correspond à une fréquence de transmission de 9.6 KHz. La tension de sortie standard pour la communication série est de -12 volts à +12 volts. Une tension de -12 volts correspond à un zéro logique. La communication série entre le robot et le PC est « Full-duplex », ce qui signifie que l'ordinateur peut transmettre et recevoir des données en même temps.

5.3 La communication RF

Étant donné que le système est « Full-duplex », il est nécessaire d'utiliser deux modules de communications par élément. C'est-à-dire un module de transmission et de réception pour l'ordinateur et un autre module de transmission et de réception pour le robot. De plus, chacune des paires de modules (transmission et réception) doit être à des fréquences différentes pour éliminer l'interférence possible entre les deux. Ces fréquences sont 20 MHz et 25 MHz pour permettre un filtrage plus efficace. Ces fréquences sont dans la bande des fréquences RF (Radio Frequency). Elles permettent d'utiliser des antennes de hauteurs raisonnables sans les inconvénients des très hautes fréquences. Ce point sera développé plus loin.

5.4 Interférence sur le board en plastique

5.4.1 Définition du problème

Le premier problème majeur rencontré dans cette partie a été l'interférence dans les boards en plastique. Ce problème s'applique autant pour le module de transmission que le module de réception. En effet, après avoir interrogé quelques personnes aux sujets des mystérieuses fréquences dans les boards, on a appris qu'à des fréquences dépassant 10 MHz ce type de board devient très capacitif ce qui crée des fréquences non désirées dans tout le board.

5.4.2 Solutions apportées

Ce problème a été réglé en utilisant des plaquettes sur lesquelles nous sommes en mesure de souder directement les composantes. Après quelques tests, les résultats ont démontré que les composantes devaient être obligatoirement placées très près de la plaquette parce que les fils qui dépassaient avaient les mêmes propriétés que des antennes, soit de capter les ondes des alentours, ce qui créait des interférences.

5.5 Module de transmission

Le module de transmission est composé de trois modules sans l'antenne. Le premier module est l'ordinateur (ou le robot) suivi du modulateur AM et d'un amplificateur Vidéo (voir le schéma ci-dessous). Le modulateur AM est décrit dans la section suivante. L'amplificateur Vidéo, quant à lui, provient de la compagnie Analog Devices. Ce circuit intégré permet d'avoir la puissance nécessaire pour l'émission (100 mW) et ainsi avoir la portée de communication suffisante.

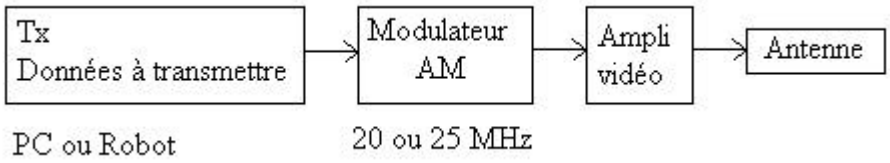


Figure 20 : Schéma de transmission

5.5.1 Modulateur AM

Le principe du modulateur est le suivant. Lorsque la sortie de l'ordinateur (ou du robot) est basse (environ -12 V) le circuit émet une onde de grande amplitude et lorsque la sortie est haute (environ +12 V), le circuit émet seulement une onde de très faible amplitude d'où la modulation en amplitude.

5.5.1.1 Modulation grâce au XR2206

Le XR2206 est un circuit intégré pouvant faire de la modulation AM en fonction d'une entrée TTL. Cette alternative a dû être laissée de côté étant donné que le XR2206 ne pouvait pas moduler à des fréquences supérieures à 1MHz ce qui représentait une grandeur physique d'antenne beaucoup trop élevée pour notre application.

5.5.1.2 Fréquence de modulation

La fréquence de modulation choisie est de 20 ou 25 MHz dépendant du module utilisé. Ces valeurs ont été fixées en tenant compte des difficultés relatives qui existent à des fréquences trop élevées et des paramètres physiques de l'antenne qui seront discutés au point 5.5.4. La modulation de type AM a été privilégiée à la modulation FM puisqu'il est plus simple d'en faire la démodulation.

5.5.1.3 Conversion de la tension d'entrée du transistor PNP

Lors d'un premier design, la sortie TX du PC était connectée directement dans l'entrée du transistor PNP comme représenté sur le schéma ci-dessous. Étant donné que le transistor PNP avait besoin de beaucoup trop de courant, celui-ci écrasait la tension de ±12 V et faisait chauffer la résistance de 50 Ω lorsque la tension était à un niveau logique bas.

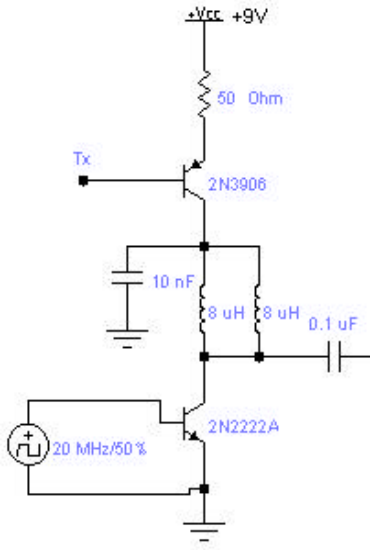


Figure 21 : Schéma du modulateur sans modification à l'entrée TX

5.5.1.4 Ajout d'un amplificateur suiveur et d'un diviseur de tension

Le problème a pu être réglé en utilisant un amplificateur opérationnel TL082 connecté en suiveur pour fournir le courant au transistor PNP sans écraser le signal du PC. Un diviseur de tension a été utilisé de manière à réduire la tension à la base du transistor entre les bornes ±5V. Ce diviseur de tension a été réalisé avec un potentiomètre de façon à pouvoir ajuster le facteur de division selon nos besoins ou selon une variation de la tension sortie d'un PC à l'autre.

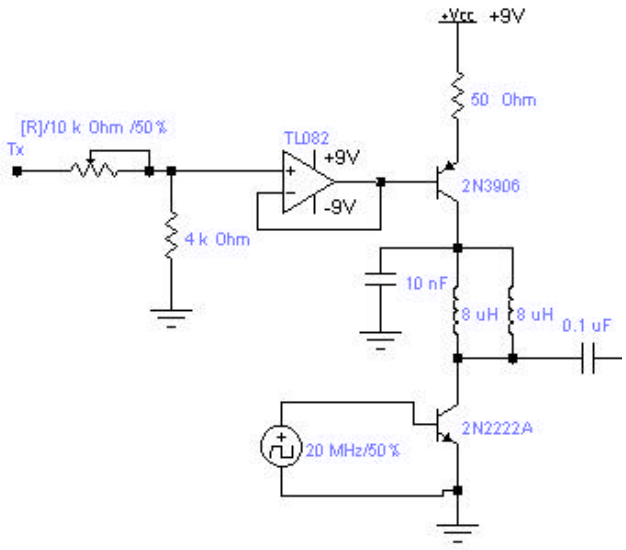


Figure 22 : Schéma du modulateur avec les ajouts aux circuits pour l'entrées Tx

5.5.2 Amplificateur vidéo

Un amplificateur vidéo a été utilisé entre la sortie du modulateur et l'antenne de manière à augmenter la puissance émise par l'antenne. L'amplificateur utilisé a été un AD818, c'est un amplificateur opérationnel possédant une grande largeur de bande passante d'où le nom d'amplificateur vidéo.

5.5.2.1 Problème d'impédance d'entrée de l'ampli vidéo

Le problème que nous avons rencontré avec l'ampli vidéo pour augmenter la puissance de sortie était d'ajuster l'impédance d'entrée de l'amplificateur, car par connexion directe, le signal à la sortie était déformé.

5.5.2.2 Solution au problème d'impédance

Le problème a été facilement résolu en utilisant un potentiomètre placé à l'entrée de l'amplificateur (voir schéma ci-dessous) de manière à ajuster l'impédance du signal et équilibrer les deux impédances de façon à obtenir le signal idéal en sortie.

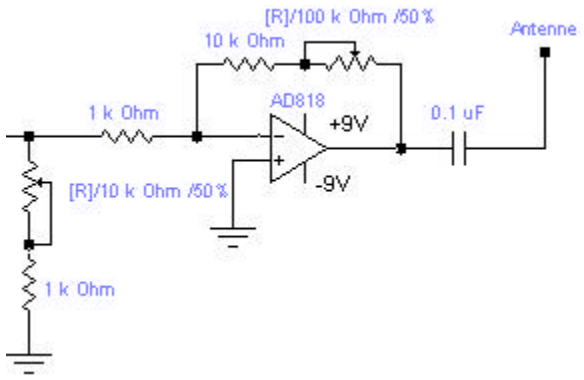


Figure 23 : Schéma représentant l'amplificateur vidéo

5.5.3 Antennes

Le type d'antenne que nous avons utilisé est des antennes de type λ/4 avec image. C'est comme l'antenne de nos automobiles! La longueur de l'antenne sera de 0.75 mètres ce qui devrait nous permettre de bien capter les fréquences que nous avons utilisées. La formule pour déterminer la longueur de l'antenne de manière à capter le signal de façon optimale est:

$\lambda = c / f$

où c = vitesse de la lumière, f = fréquence de la porteuse, λ = longueur de l'onde.

Notez qu'étant donné la proximité de la source d'émission et de réception, il n'est pas essentiel de respecter à la lettre la formule précédente. En effet, même si l'antenne est beaucoup plus petite que la théorie le voudrait, l'antenne va quand même bien capter notre signal.

5.5.4 Signal modulant

Le signal modulant est en fait la porteuse avec laquelle le signal numérique est modulé. Pour les deux modules de transmission, la porteuse est soit 20 MHz ou 25 MHz. Un cristal oscillateur a été utilisé pour produire la porteuse, celui-ci produit une onde TTL carrée 0-5V à une fréquence qui est propre au cristal, 20 ou 25 MHz dans le cas présent. Un condensateur a été utilisé pour éliminer le DC de l'onde de manière à placer celle-ci entre -2.5V et 2.5V.

5.5.4.1 Cristal oscillateur qui surchauffe

Lors des premiers essais, la sortie du cristal était connectée directement dans l'entrée du transistor NPN par le biais d'un condensateur. Résultats, le transistor tirait trop de courant dans le cristal et le cristal chauffait.

5.5.4.2 Insertion d'un op amp en mode suiveur entre le cristal et le NPN

Après quelques essais, les résultats ne furent pas satisfaisants, le signal obtenu à la sortie de l'amplificateur n'était pas très bon. Ceci est dû au fait que le cristal a une impédance de sortie trop faible et l'ampli réussi à lui tirer trop de courant ce qui atténue la sortie du cristal à un niveau trop faible pour être bien amplifié.

5.5.4.3 Conception d'un signal modulant de grande puissance

Ce nouveau circuit est conçu de façon à répondre à une forte demande en courant de la part du transistor NPN sans modifier sa sortie. Ce circuit est réalisé à l'aide d'un circuit intégré 8284 de Intel qui sert de contrôleur d'horloge et de compteurs 4096 qui permettent de diviser l'horloge obtenue avec le 8284 de manière à obtenir la fréquence d'horloge désirée. Malheureusement, cette approche n'a pas été testée dû au manque de temps.

5.5.4.4 Signal obtenu à la sortie

Pour effectuer les tests sur le module de transmission un générateur a été utilisé au lieu du cristal. Les résultats obtenus ont été surprenants, le niveau logique 1 était modulé avec une amplitude faible et le zéro avec une amplitude élevée. Par contre, après étude du signal nous nous sommes aperçus qu'il était modulé avec une porteuse de 1 MHz, cependant cette porteuse était modulée à 20 MHz.

5.5.4.5 Hypothèse

Après avoir étudié notre circuit, nous avons émis l'hypothèse que le transistor NPN n'est pas en mesure de répondre à 20 MHz. Cependant, lorsque l'on diminuait le générateur sous 1 MHz nous perdions complètement le signal. Étant donné que nous ne connaissions pas la cause exacte de ce problème et dû au manque de temps, nous avons donc laissé cette hypothèse en suspend.

5.5.5 - Circuit complet pour la transmission

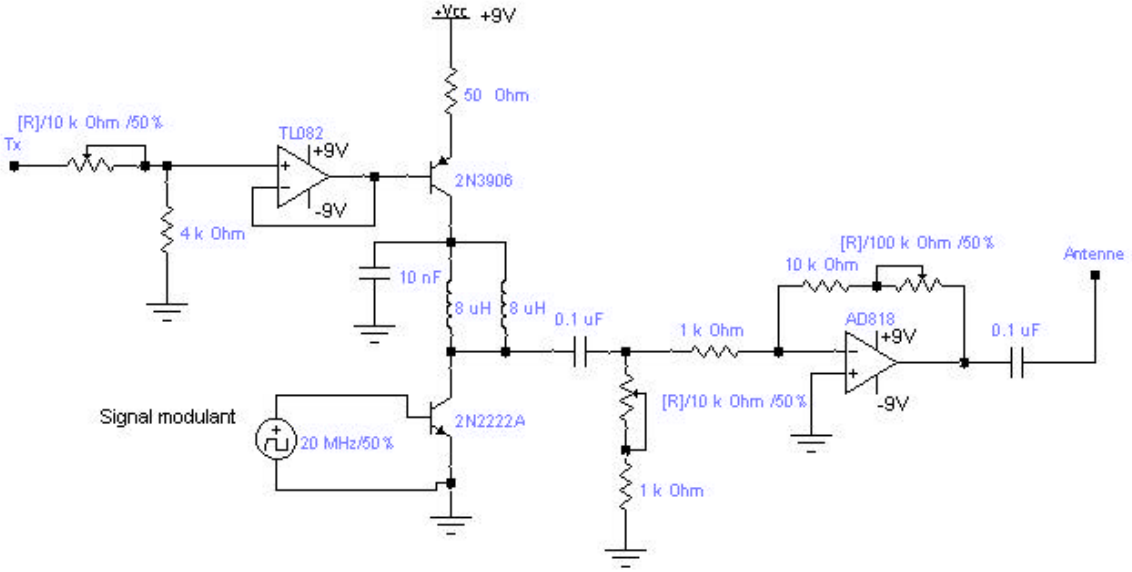
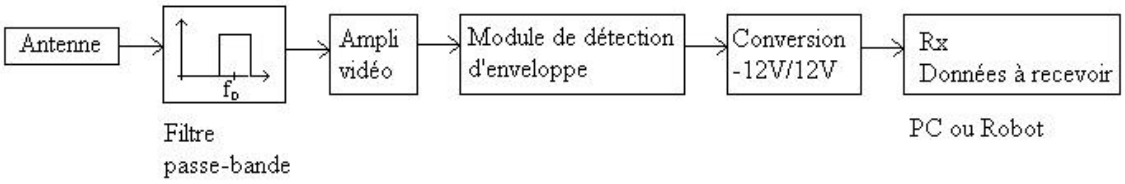


Figure 24 : Circuit complet de transmission

5.6 Module de réception

Le module de réception (voir schéma) est composé de cinq modules sans l'antenne. Le premier module est le filtre passe-bande (voir schéma) suivi d'un amplificateur vidéo, du module de détection d'enveloppe, d'une conversion à –12 +12 volts et finalement de l'ordinateur (ou du robot). Le module Conversion dépendra des résultats obtenus lors de la transmission.



5.6.1 Le filtre passe-bande

Figure 25 : Schéma de réception

Lorsque l'onde s'est propagée dans l'espace, elle est captée par l'antenne. Toutefois, puisque les ondes hertziennes sont très populaires il faut filtrer les ondes indésirables. Nous avons donc décidé d'utiliser un filtre passe-bande pour garder seulement la fréquence utile de transmission soit 20 MHz et 25 MHz.

5.6.1.1 Filtre de type RLC

Le premier filtre réalisé a été un simple filtre RLC classique (voir figure ci-dessous). Les calculs de la fréquence de coupure W_0 de ce filtre sont faciles à réaliser. De plus, sa réalisation pratique est simple. Nous avons décidé de simuler le filtre à l'aide du logiciel « Electronic Workbench » comme cela on s'assurait de la validité de nos

calculs. Les résultats obtenus furent très positifs: une bande passante de 1.6 MHz était obtenue avec les valeurs d'inductance et de condensateur : L = 2 µH et C = 31.6 pF.

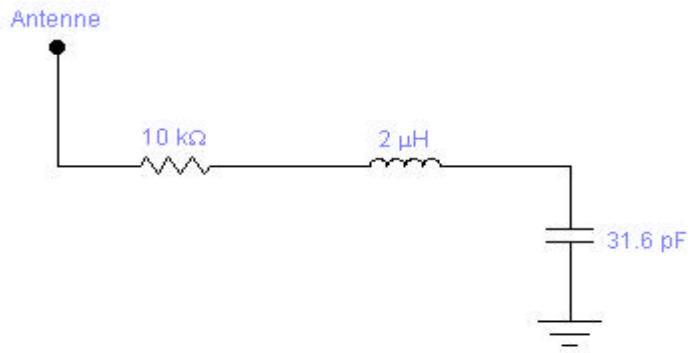


Figure 26 : Filtre passe-bande actif RLC

Cependant, les résultats pratiques obtenus avec ce filtre n'ont pas été convaincants et nous ne pouvions pas l'utiliser puisqu'il comportait une bande passante beaucoup trop grande puisqu'il laissait passer la fréquence de 25 MHz lorsqu'il était configuré pour avoir une fréquence de coupure de 20 MHz. Comme quoi la pratique est bien différente de la théorie ! Notre hypothèse à ce sujet, l'interférence des ondes ambiantes à 20 MHz causait des oscillations parasites.

5.6.1.2 Filtre actif RC

Le deuxième filtre que nous avons utilisé est celui réalisé lors de notre cours d'électronique II (voir schéma ci-dessous). Cependant, il fallait utiliser des amplificateurs Vidéo. Les calculs nécessaires pour déterminer les valeurs des résistances et condensateurs sont les suivant :

$F_0 = 1/(2*pi*R*C) = 20$ MHz $\Rightarrow$ R = 8 KΩ et C = 1 pF

$Q = W_0 / W_H - W_L = 20$ puisque $W_H - W_L = 1$

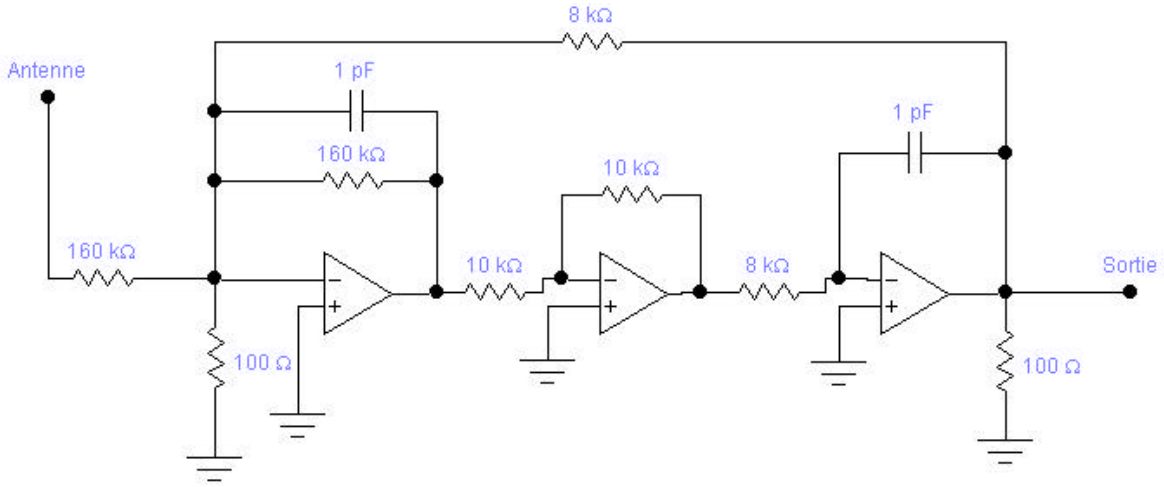


Figure 27 : Filtre actif RC

Rb = Q*R = 160 KΩ, Ra = Rb = 160 KΩ, R1 = R2 = 10 KΩ

Même si nous avons ajouté deux résistances de 100 Ω pour diminuer le courant à l'entrée d'un amplificateur vidéo comme le fabricant l'indique. Le circuit ne fonctionnait pas. Encore une fois nous étions victimes des interférences dues aux hautes fréquences.

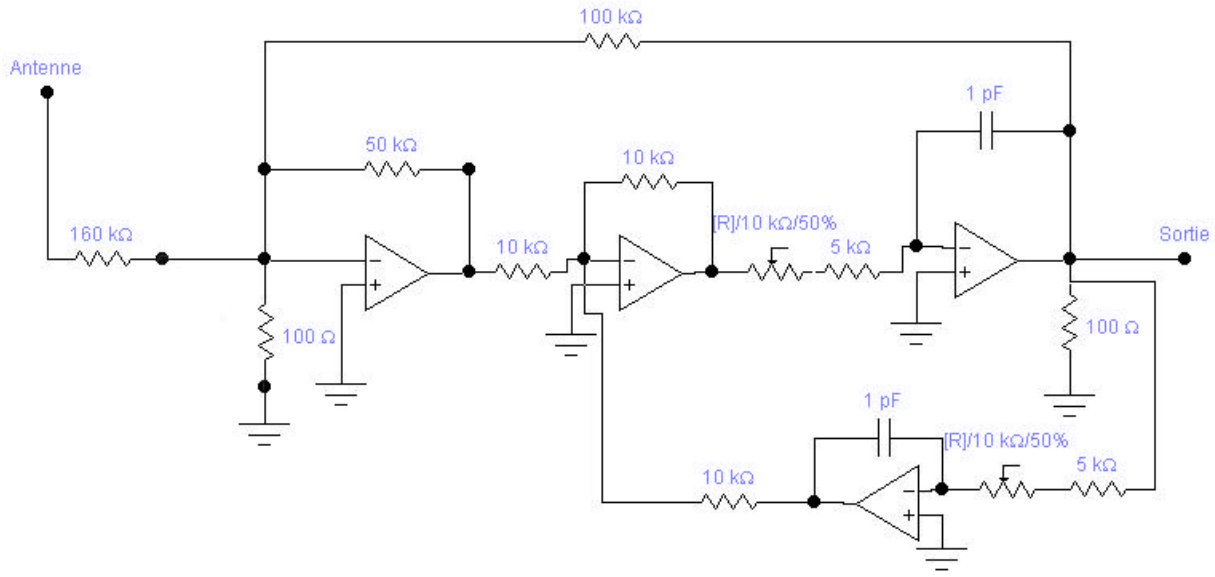
5.6.1.3 Filtre actif RC avec bande passante et Gain ajustable

Le calcul des composantes de ce filtre est similaire au précédent :

F_0= 1/(2*pi*R*C) = 20 MHz $\Rightarrow$ R = 8 KΩ et C = 1 pF

Q = R_1/R_Q = 20, Gain = R_1/R_G = 1, R_G = 50 KΩ, R_Q = 5 KΩ

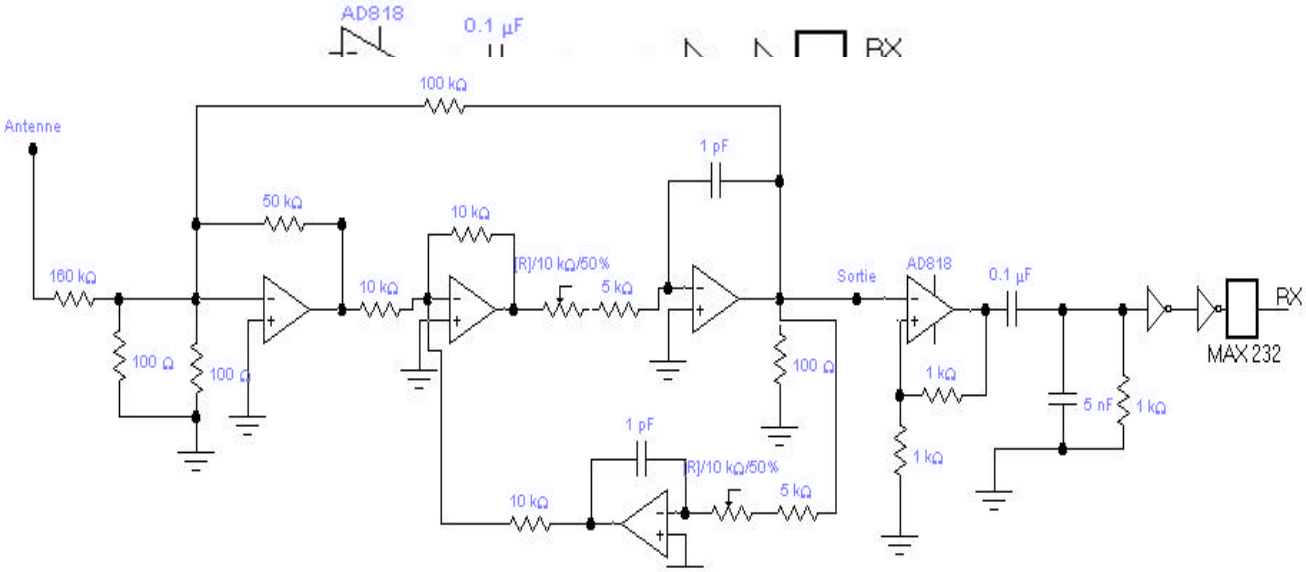
Figure 28 : Schéma du filtre passe-bande avec gain et bande-passante ajustable



Un potentiomètre de 10 KΩ a été ajouté à la résistance R_F pour permettre l'ajustement de la fréquence de coupure F_0. Ce circuit est un autre modèle que nous voulions utiliser, mais nous n'avons pas pu faire le montage et les tests dû au manque de temps.

5.6.2 L'amplification du signal et la détection d'enveloppe

Le signal filtré par le filtre passe-bande est ensuite amplifié par l'amplificateur vidéo et filtré à nouveau pour réaliser la détection de l'enveloppe du signal AM (voir schéma ci-dessous). Ensuite, le signal détecté est converti en TTL par les deux inverseurs. Il ne suffit plus que de reconvertir le signal en +12 Volts –12 volts, les tensions du port série de l'ordinateur ou du robot. Cette étape de la conversion n'a pas pu être testée puisque nous n'avons pas réussi à faire la modulation AM.



5.7 Conclusion

Pour conclure, nous avons réussi à surmonter plusieurs obstacles. Cependant, malgré plusieurs tests et designs, nous n'avons pas réussi à atteindre le but qui était de concevoir un lien sans fil entre le robot et l'ordinateur. Nous en sommes donc arrivé à la conclusion: l'application des concepts théoriques à la pratique ne se fait pas facilement et les hautes fréquences sont un domaine difficile d'accès puisque les propriétés des composantes changent énormément dans de telles conditions.

6. Module Caméra

Ce module constitue la caractéristique distinctive de notre projet.

6.1 Introduction

Au nombre des fléaux qui affligent l'humanité en cette fin de vingtième siècle, le fait de ne pas voir où on s'en va figure sûrement en tête de liste! Chez Charlie Inc, nous avons la conviction profonde que pour aller de l'avant, il faut regarder droit devant nous. C'est dans cette optique que s'inscrit l'implantation d'un système vidéo.

6.2 Principe de fonctionnement

Avant d'analyser en détails les caractéristiques du système vidéo, il convient d'en étudier les différentes parties afin de cerner adéquatement le principe de fonctionnement. La figure 6.1 présente, sous forme d'un diagramme bloc, le système vidéo dans son ensemble :

Schéma Bloc du système vidéo

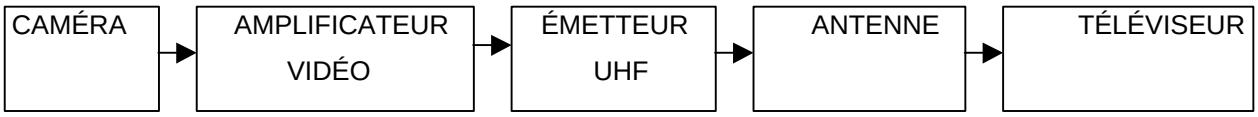


Figure 31 : Schéma bloc du module caméra

Le premier élément du système est évidemment la caméra. Celle-ci est fixée sur le devant du robot. Elle transmet les images à un amplificateur vidéo qui amplifie le signal provenant de la caméra. L'émetteur UHF reçoit le signal de l'amplificateur et le transmet, au moyen de l'antenne, à la fréquence du canal 19. Naturellement, c'est le téléviseur qui reçoit finalement l'image provenant de la caméra.

Grâce à ce système vidéo, l'utilisateur peut visualiser sur l'écran du téléviseur l'environnement se trouvant devant le robot. Il peut ainsi contrôler le robot à partir d'une autre pièce tout en étant informé des obstacles pouvant se trouver devant lui.

6.3 Caractéristiques techniques

Cette section contient toutes les informations techniques concernant les diverses parties du système vidéo.

6.3.1 Caméra

La caméra utilisée dans ce projet est une caméra noir et blanc de la compagnie Marshall Electronics Inc. La caméra est entièrement construite sur un seul circuit intégré CMOS. Nous avons choisi cette caméra en raison de sa petite taille, de son poids négligeable et de sa faible consommation électrique. La figure 6.2 montre notre caméra (à gauche) à côté d'une pièce de un sou.

Photo de notre caméra (à gauche)

Figure 32 : Photo de la caméra

La photo ci-haut présente le modèle VX-0071 (à gauche) et le modèle VX-0072 qui ne se distingue du précédent que par son boîtier. La caméra fonctionne avec une alimentation électrique de 6 à 12 volts. Par commodité, nous avons utilisé une pile alcaline de 9 volts. On obtient alors une autonomie de 24 heures, donc la caméra est

relativement économique en énergie. Nous pourrions toutefois augmenter l'autonomie à 35 heures en utilisant une pile au lithium. Le signal disponible à la sortie du circuit de la caméra est un signal vidéo codé en NTSC (système utilisé partout en Amérique du Nord) et peut se connecter directement à une entrée vidéo 75 ohms. La définition de l'image est respectable, c'est à dire 320 lignes horizontales et 240 lignes verticales. L'intensité lumineuse nécessaire pour obtenir une image nette est de 0.9 lux, ce qui est très bien puisque notre caméra peut transmettre des images dans presque n'importe quelles conditions lumineuses. En effet, 1 lux correspond approximativement à l'intensité lumineuse d'une chandelle. Avec un éclairage minimum de 0.9 lux, nous pouvons donc filmer avec très peu de lumière. Le "focus" est ajustable en dévissant ou en vissant la lentille.

6.3.2 Amplificateur Vidéo

L'amplificateur vidéo constitue le deuxième bloc du système de transmission vidéo. Il est nécessaire puisque le signal disponible à la sortie de la caméra n'est pas assez fort pour être envoyé directement à l'émetteur. La figure 6.3 présente le schéma électrique de l'amplificateur.

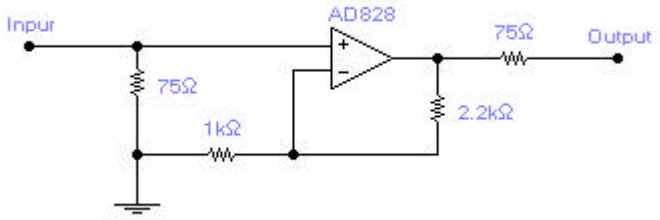


Figure 33 : Schéma électrique de l'amplificateur vidéo

Le cœur du circuit est un amplificateur opérationnel AD828 de la compagnie Analog Devices. C'est l'amplificateur idéal pour réaliser ce type de circuit puisqu'il permet d'obtenir un gain suffisant tout en conservant une largeur de bande supérieure à ce

qu'on obtiendrait avec un autre type d'amplificateur opérationnel, disons un LM741 par exemple.

L'impédance d'entrée du circuit est de 75 ohms afin de répondre aux exigences de la caméra. Il en va de même pour la sortie qui doit se brancher dans l'émetteur UHF. Le AD828 est alimenté avec des tensions de +5 et –5 volts.

6.3.3 Émetteur UHF

Cette troisième partie du système vidéo permet d'envoyer le signal de la caméra, amplifié par l'amplificateur vidéo, jusqu'à la télévision par l'intermédiaire de l'antenne. L'émetteur utilisé est un modèle tout fait d'avance, un modèle commercial, développé par P.A.E.J Enterprises. Les caractéristiques de cet émetteur en ont fait un choix de prédilection pour notre type d'application. En effet, nous cherchions un émetteur de petite taille, de faible poids et surtout économique en énergie. Voici un résumé des caractéristiques de cet émetteur :

- La fréquence de transmission est déterminée par un cristal
- Il émet à la fréquence du canal 19 (en UHF)
- Il possède une portée de 300 pieds
- Il s'alimente avec une pile alcaline de 9 volts
- Son poids est de 8 onces (226.8 grammes)
- La longueur de l'antenne est de 19 pouces (48.26 cm) donc une longueur raisonnable pour être fixée sur le robot.

Bien entendu, comme il s'agit d'un émetteur UHF, le téléviseur qui reçoit le signal doit être équipé d'une antenne UHF. Celle-ci est tout simplement une petite antenne en

forme de cercle qu'on connecte à la prise de câble 75 ohms du téléviseur via un transformateur d'impédance 300 ohms / 75 ohms.

6.4 Étapes de développement, problèmes et solutions

Cette section présente brièvement la démarche que nous avons adoptée pour réaliser le système vidéo. Nous vous ferons part également de certaines embûches rencontrées en cours de réalisation et des solutions qui nous ont permis de les surmonter.

6.4.1 Hypothèses de base

Le système vidéo devait être composé à la base d'une caméra et d'un émetteur. La caméra devait envoyer un signal vidéo directement à l'émetteur, et ce dernier devait l'émettre, à l'aide d'une antenne, vers un téléviseur. Nous avons donc concentré nos efforts sur la recherche d'une caméra et d'un émetteur. Ceux-ci devaient être petits, légers et consommer un minimum d'énergie. Les choix retenus vous ont été présentés dans la section précédente.

6.4.2 Problèmes rencontrés et solutions proposées

Le premier problème rencontré est le suivant : le signal disponible à la sortie de la caméra n'est pas suffisamment fort pour être transmis convenablement par l'émetteur. La solution adoptée fut de l'amplifier au moyen d'un amplificateur vidéo. Le défi qui se présentait alors était de trouver un amplificateur opérationnel possédant une largeur de bande suffisamment grande pour notre signal tout en offrant un gain d'au moins deux. Nous avons opté pour un circuit intégré de la compagnie Analog Devices, le AD828. Les délais de livraison vers le Canada étaient malheureusement considérables.

Le problème suivant était de taille : le circuit ne fonctionnait pas! Une étude exhaustive de la situation a permis de comprendre qu'un « breadboard » peut parfois jouer de vilains tours à hautes fréquences. Une fois le circuit soudé sur une plaquette de montage, un signal est apparu à l'écran du téléviseur. Néanmoins, la joie éprouvée alors fut cruellement réprimée par une réalité frustrante : l'image était incohérente, distorsionnée et trop foncée. Nous avons compris alors l'importance d'adapter convenablement les impédances des différentes composantes d'un système.

6.5 Conclusion

La version finale du système vidéo implémenté sur le robot est celle présentée à la section 6.2. Nous croyons que cette caractéristique distinctive est un atout important de notre projet puisqu'elle permet de contrôler le robot à partir d'une autre pièce tout en étant informé des obstacles pouvant se retrouver devant le robot. De plus, sur un plan académique, les connaissances acquises lors de la réalisation de cette partie du projet n'ont d'égale que l'expérience dont nous nous sommes enrichis en la menant à terme de si brillante façon.

7. Conclusion

À la lumière des résultats obtenus par Charlie Inc., il ressort clairement que le projet a été une réussite. Toutefois, comme dans toute analyse scientifique, des améliorations souhaitables ont été identifiées et, si nous disposions de plus de temps, une version révisée de notre produit pourrait voir le jour.

En plus de permettre l'intégration de connaissances diverses ayant trait aux domaines de l'électronique et des communications, ce projet s'est avéré une excellente expérience de travail en équipe et de gestion de projet. Nous avons appris l'importance d'une bonne communication au sein de l'entreprise et avons compris l'utilité d'un échéancier réaliste. Il a été très intéressant de travailler avec un groupe imposé qui représentait bien les défis auxquels nous ferons face dans nos carrières d'ingénieurs.

De plus, comme la gestion du budget fera partie intégrante de notre travail d’ingénieur, il est important de noter le respect d’un budget de travail minime.